C55x v3.x CPU Algebraic Instruction Set Reference Guide

Literature Number: SWPU068E
June 2009

TEXAS INSTRUMENTS

Preface

Read This First

About This Manual

The C55x™ is a fixed-point digital signal processor (DSP) in the TMS320™ DSP family, and it can use either of two forms of the instruction set: a mnemonic form or an algebraic form. This book is a reference for the algebraic form of the instruction set. It contains information about the instructions used for all types of operations. For information on the mnemonic instruction set, see *C55x v3.x CPU Mnemonic Instruction Set Reference Guide*, SWPU067.

This release is updated with the 3.0 Revision of the TMS320C55x DSP. Information not affected by the revision remains identical to the previous manual. The main new features of this revision are:

❑ Relaxed parallelism restrictions:

  Total size of both instructions may be up to 8 bytes.

  Constant buses (KAB and KDB) are no longer a source of conflict.

❑ New instructions:

  lsmf

  36 dual mac instructions with double coefficient features

  MPY, MAC, and MAS instructions with unsigned coefficients

  lock

Notational Conventions

This book uses the following conventions.

- In syntax descriptions, the instruction is in a **bold typeface**. Portions of a syntax in **bold** must be entered as shown. Here is an example of an instruction syntax:

  lms(Xmem, Ymem, ACx, ACy**)**

  lms is the instruction, and it has four operands: *Xmem*, *Ymem*, *ACx*, and *ACy*. When you use **lms**, the operands should be actual dual data-memory operand values and accumulator values. A comma and a space (optional) must separate the four values.

- Square brackets, [and], identify an optional parameter. If you use an optional parameter, specify the information within the brackets; do not type the brackets themselves.

Related Documentation From Texas Instruments

The following books describe the C55x™ devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

TMS320C55x Technical Overview (SPRU393). This overview is an introduction to the TMS320C55x™ digital signal processor (DSP). The TMS320C55x is the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features of the TMS320C55x.

C55x CPU Reference Guide (literature number SWPU073) describes the architecture, registers, and operation of the CPU for the TMS320C55x™ digital signal processors (DSPs).

C55x CPU Mnemonic Instruction Set Reference Guide (literature number SWPU067) describes the mnemonic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x™ DSPs and explains how to write code that uses special features and instructions of the DSP.

TMS320C55x Optimizing C Compiler User's Guide (literature number SPRU281) describes the TMS320C55x™ C Compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x™ devices.

Trademarks

TMS320, TMS320C54x, TMS320C55x, C54x, and C55x are trademarks of Texas Instruments.

Contents

Figures

Tables

Chapter 1

Terms, Symbols, and Abbreviations

This chapter lists and defines the terms, symbols, and abbreviations used in the TMS320C55x™ DSP algebraic instruction set summary and in the individual instruction descriptions. Also provided are instruction set notes and rules and a list of nonrepeatable instructions.

Topic **Page**

1.1 Instruction Set Terms, Symbols, and Abbreviations

Table 1−1 lists the terms, symbols, and abbreviations used and Table 1−2 lists the operators used in the instruction set summary and in the individual instruction descriptions.

Table 1−1. Instruction Set Terms, Symbols, and Abbreviations

Symbol	Meaning
[ ]	Optional operands
ACB	Bus that brings D-unit registers to A-unit and P-unit operators
ACOVx	Accumulator overflow status bit: ACOV0, ACOV1, ACOV2, ACOV3
ACw, ACx, ACy, ACz	Accumulator: AC0, AC1, AC2, AC3
ARn_mod	Content of selected auxiliary register (ARn) is premodified or postmodified in the address generation unit.
ARx, ARy	Auxiliary register: AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
AU	A unit
Baddr	Register bit address
BitIn	Shifted bit in: Test control flag 2 (TC2) or CARRY status bit
BitOut	Shifted bit out: Test control flag 2 (TC2) or CARRY status bit
BORROW	Logical complement of CARRY status bit
C, Cycles	Execution in cycles. For conditional instructions, x/y field means: x cycle, if the condition is true. y cycle, if the condition is false.
CA	Coefficient address generation unit
CARRY	Value of CARRY status bit
Cmem	Coefficient indirect operand referencing a 16-bit or 32-bit value in data space
cond	Condition based on accumulator value (ACx), auxiliary register (ARx) value, temporary register (Tx) value, test control (TCx) flag, or CARRY status bit. See section 1.2.
CR	Coefficient Read bus
CSR	Computed single-repeat register

Table 1−1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
DA	Data address generation unit
DR	Data Read bus
dst	Destination accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
DU	D unit
DW	Data Write bus
Dx	Data address label coded on x bits (absolute address)
E	Indicates if the instruction contains a parallel enable bit.
kx	Unsigned constant coded on x bits
Kx	Signed constant coded on x bits
Lmem	Long-word single data memory access (32-bit data access). Same legal inputs as Smem.
lx	Program address label coded on x bits (unsigned offset relative to program counter register)
Lx	Program address label coded on x bits (signed offset relative to program counter register)
M40	If the optional M40 keyword is applied to the instruction, the instruction provides the option to locally set M40 to 1 for the execution of the instruction
Operator	Operator(s) used by an instruction.
Pipe, Pipeline	Pipeline phase in which the instruction executes: AD Address D Decode R Read X Execute
Px	Program or data address label coded on x bits (absolute address)
RELOP	Relational operators: == equal to < less than >= greater than or equal to != not equal to

Table 1−1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
rnd	If the optional rnd keyword is applied to the instruction, rounding is performed in the instruction
RPTC	Single-repeat counter register
S, Size	Instruction size in bytes.
SA	Stack address generation unit
saturate	If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated
SHFT	4-bit immediate shift value, 0 to 15
SHIFTW	6-bit immediate shift value, −32 to +31
Smem	Word single data memory access (16-bit data access)
SP	Data stack pointer
src	Source accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
SSP	System stack pointer
STx	Status register: ST0, ST1, ST2, ST3
TAx, TAy	Auxiliary register (ARx) or temporary register (Tx): AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
TCx**,** TCy	Test control flag: TC1, TC2
TRNx	Transition register: TRN0, TRN1
Tx, Ty	Temporary register (Tx): T0, T1, T2, T3
uns	If the optional uns keyword is applied to the input operand, the operand is zero extended
XACdst	Destination extended register: All 23 bits of coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7

Table 1−1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
XACsrc	Source extended register: All 23 bits of coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAdst	Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), and data page pointer (XDP)
XARx	All 23 bits of extended auxiliary register: XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAsrc	Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), and data page pointer (XDP)
xdst	Accumulator: AC0, AC1, AC2, AC3 Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
xsrc	Accumulator: AC0, AC1, AC2, AC3 Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
Xmem, Ymem	Indirect dual data memory access (two data accesses)

Table 1−2. Operators Used in Instruction Set

Symbols	Operators	Evaluation
+ − ~	Unary plus, minus, 1s complement	Right to left
* / %	Multiplication, division, modulo	Left to right
+ −	Addition, subtraction	Left to right
<< >>	Signed left shift, right shift	Left to right
< < < >>>	Logical left shift, logical right shift	Left to right
< <=	Less than, less than or equal to	Left to right
> >=	Greater than, greater than or equal to	Left to right
== !=	Equal to, not equal to	Left to right
&	Bitwise AND	Left to right
|	Bitwise OR	Left to right
^	Bitwise exclusive OR (XOR)	Left to right

Note: Unary +, −, and * have higher precedence than the binary forms.

1.2 Instruction Set Conditional (cond) Fields

Table 1−3 lists the testing conditions available in the cond field of the conditional instructions.

Table 1−3. Instruction Set Conditional (cond) Field

Bit or Register	Condition (cond) Field	For Condition to be True ...
Accumulator	Tests the accumulator (ACx) content against 0. The comparison against 0 depends on M40 status bit: ❑ If M40 = 0, ACx(31–0) is compared to 0. ❑ If M40 = 1, ACx(39–0) is compared to 0.	
	ACx == #0	ACx content is equal to 0
	ACx < #0	ACx content is less than 0
	ACx > #0	ACx content is greater than 0
	ACx != #0	ACx content is not equal to 0
	ACx <= #0	ACx content is less than or equal to 0
	ACx >= #0	ACx content is greater than or equal to 0
Accumulator Overflow Status Bit	Tests the accumulator overflow status bit (ACOVx) against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0. When this condition is used, the corresponding ACOVx is cleared to 0.	
	overflow(ACx)	ACOVx bit is set to 1
	!overflow(ACx)	ACOVx bit is cleared to 0
Auxiliary Register	Tests the auxiliary register (ARx) content against 0.	
	ARx == #0	ARx content is equal to 0
	ARx < #0	ARx content is less than 0
	ARx > #0	ARx content is greater than 0
	ARx != #0	ARx content is not equal to 0
	ARx <= #0	ARx content is less than or equal to 0
	ARx >= #0	ARx content is greater than or equal to 0
CARRY Status Bit	Tests the CARRY status bit against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0.	
	CARRY	CARRY bit is set to 1
	!CARRY	CARRY bit is cleared to 0

Table 1−3. Instruction Set Conditional (cond) Field (Continued)

Bit or Register	Condition (cond) Field	For Condition to be True ...
Temporary Register	Tests the temporary register (Tx) content against 0.	
	Tx == #0	Tx content is equal to 0
	Tx < #0	Tx content is less than 0
	Tx > #0	Tx content is greater than 0
	Tx != #0	Tx content is not equal to 0
	Tx <= #0	Tx content is less than or equal to 0
	Tx >= #0	Tx content is greater than or equal to 0
Test Control Flags	Tests the test control flags (TC1 and TC2) independently against 1; when the optional ! symbol is used before the flag designation, the flag can be tested independently against 0.	
	TCx	TCx flag is set to 1
	!TCx	TCx flag is cleared to 0
	TC1 and TC2 can be combined with an AND (&), OR (|), and XOR (^) logical bit combinations:	
	TC1 & TC2	TC1 AND TC2 is equal to 1
	!TC1 & TC2	$\overline{\text{TC1}}$ AND TC2 is equal to 1
	TC1 & !TC2	TC1 AND $\overline{\text{TC2}}$ is equal to 1
	!TC1 & !TC2	$\overline{\text{TC1}}$ AND $\overline{\text{TC2}}$ is equal to 1
	TC1 | TC2	TC1 OR TC2 is equal to 1
	!TC1 | TC2	$\overline{\text{TC1}}$ OR TC2 is equal to 1
	TC1 | !TC2	TC1 OR $\overline{\text{TC2}}$ is equal to 1
	!TC1 | !TC2	$\overline{\text{TC1}}$ OR $\overline{\text{TC2}}$ is equal to 1
	TC1 ^ TC2	TC1 XOR TC2 is equal to 1
	!TC1 ^ TC2	$\overline{\text{TC1}}$ XOR TC2 is equal to 1
	TC1 ^ !TC2	TC1 XOR $\overline{\text{TC2}}$ is equal to 1
	!TC1 ^ !TC2	$\overline{\text{TC1}}$ XOR $\overline{\text{TC2}}$ is equal to 1

1.3 Affect of Status Bits

1.3.1 Accumulator Overflow Status Bit (ACOVx)

The ACOV[0−3] depends on M40:

- ❑ When M40 = 0, overflow is detected at bit position 31
- ❑ When M40 = 1, overflow is detected at bit position 39

If an overflow is detected, the destination accumulator overflow status bit is set to 1.

1.3.2 C54CM Status Bit

- ❑ When C54CM = 0, the enhanced mode, the CPU supports code originally developed for a TMS320C55x™ DSP.
- ❑ When C54CM = 1, the compatible mode, all the C55x CPU resources remain available; therefore, as you translate code, you can take advantage of the additional features on the C55x DSP to optimize your code. This mode must be set when you are porting code that was originally developed for a TMS320C54x™ DSP.

1.3.3 CARRY Status Bit

- ❑ When M40 = 0, the carry/borrow is detected at bit position 31
- ❑ When M40 = 1, the carry/borrow is detected at bit position 39

When performing a logical shift or signed shift that affects the CARRY status bit and the shift count is zero, the CARRY status bit is cleared to 0.

1.3.4 FRCT Status Bit

- ❑ When FRCT = 0, the fractional mode is OFF and results of multiply operations are not shifted.
- ❑ When FRCT = 1, the fractional mode is ON and results of multiply operations are shifted left by 1 bit to eliminate an extra sign bit.

1.3.5 INTM Status Bit

The INTM bit globally enables or disables the maskable interrupts. This bit has no effect on nonmaskable interrupts (those that cannot be blocked by software).

- ❑ When INTM = 0, all unmasked interrupts are enabled.
- ❑ When INTM = 1, all maskable interrupts are disabled.

1.3.6 M40 Status Bit

- ❑ When M40 = 0:
  - overflow is detected at bit position 31
  - the carry/borrow is detected at bit position 31
  - saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
  - TMS320C54x™ DSP compatibility mode
  - for conditional instructions, the comparison against 0 (zero) is performed on 32 bits, ACx(31−0)
- ❑ When M40 = 1:
  - overflow is detected at bit position 39
  - the carry/borrow is detected at bit position 39
  - saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)
  - for conditional instructions, the comparison against 0 (zero) is performed on 40 bits, ACx(39−0)

1.3.6.1 M40 Status Bit When Sign Shifting

In D-unit shifter:

- ❑ When shifting to the LSBs:
  - when M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted according to the shift quantity:
    - if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
    - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
  - bit 39 is extended according to SXMD
  - the shifted-out bit is extracted at bit position 0
- ❑ When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - if M40 = 0, the shifted-out bit is extracted at bit position 31
  - if M40 = 1, the shifted-out bit is extracted at bit position 39

- ❑ After shifting, unless otherwise noted, when M40 = 0:
  - ■ overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)
  - ■ the carry/borrow is detected at bit position 31
  - ■ if SATD = 1, when an overflow is detected, ACx saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
  - ■ TMS320C54x™ DSP compatibility mode
- ❑ After shifting, unless otherwise noted, when M40 = 1:
  - ■ overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)
  - ■ the carry/borrow is detected at bit position 39
  - ■ if SATD = 1, when an overflow is detected, ACx saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

In A-unit ALU:

- ❑ When shifting to the LSBs, bit 15 is sign extended
- ❑ When shifting to the MSBs, 0 is inserted at bit position 0
- ❑ After shifting, unless otherwise noted:
  - ■ overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)
  - ■ if SATA = 1, when an overflow is detected, register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

1.3.6.2 M40 Status Bit When Logically Shifting

In D-unit shifter:

- ❑ When shifting to the LSBs:
  - ■ if M40 = 0, 0 is inserted at bit position 31 and the guard bits (39−32) of the destination accumulator are cleared
  - ■ if M40 = 1, 0 is inserted at bit position 39
  - ■ the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit

- When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - if M40 = 0, the shifted-out bit is extracted at bit position 31 and stored in the CARRY status bit, and the guard bits (39−32) of the destination accumulator are cleared
  - if M40 = 1, the shifted-out bit is extracted at bit position 39 and stored in the CARRY status bit

In A-unit ALU:

- When shifting to the LSBs:
  - 0 is inserted at bit position 15
  - the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit
- When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - the shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit

1.3.7 RDM Status Bit

When the optional rnd or R keyword is applied to the instruction, then rounding is performed in the D-unit shifter. This is done according to RDM:

- When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) is added to the 40-bit result of the shift result.
- When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit result of the shift result, 8000h (2^{15}) is added:

```
if( 8000h < bit(15−0) < 10000h)
   add 8000h to the 40-bit result of the shift result.
else if( bit(15−0) == 8000h)
   if( bit(16) == 1)
   add 8000h to the 40-bit result of the shift result.
```

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

1.3.8 SATA Status Bit

This status bit controls operations performed in the A unit.

- When SATA = 0, no saturation is performed.
- When SATA = 1 and an overflow is detected, the destination register is saturated to 7FFFh (positive overflow) or 8000h (negative overflow).

1.3.9 SATD Status Bit

This status bit controls operations performed in the D unit.

❑ When SATD = 0, no saturation is performed.

❑ When SATD = 1 and an overflow is detected, the destination register is saturated.

1.3.10 SMUL Status Bit

❑ When SMUL = 0, the saturation mode is OFF.

❑ When SMUL = 1, the saturation mode is ON. When SMUL = 1, FRCT = 1, and SATD = 1, the result of 18000h × 18000h is saturated to 00 7FFF FFFFh (regardless of the value of the M40 bit). This forces the product of the two negative numbers to be a positive number. For multiply-and-accumulate/subtract instructions, the saturation is performed after the multiplication and before the addition/subtraction.

1.3.11 SXMD Status Bit

This status bit controls operations performed in the D unit.

❑ When SXMD = 0, input operands are zero extended.

❑ When SXMD = 1, input operands are sign extended.

1.3.12 Test Control Status Bit (TCx)

The test/control status bits (TC1 or TC2) hold the result of a test performed by the instruction.

1.4 Instruction Set Notes and Rules

1.4.1 Notes

- Algebraic syntax keywords and operand modifiers are case insensitive. You can write:

  ```
  abdst(*AR0, *ar1, AC0, ac1)
  ```

  or

  ```
  aBdST(*ar0, *aR1, aC0, Ac1)
  ```

- Operands for commutative operations (+, *, &, |, ^) can be arranged in any order.

- Expression qualifiers can be specified in any order. For example, these two instructions are equivalent:

  ```
  AC0 = m40(rnd(uns(*AR0) * uns(*AR1)))
  AC0 = rnd(m40(uns(*AR0) * uns(*AR1)))
  ```

- Algebraic instructions must use parenthesis in the exact form shown in the instruction set. For example, this instruction is legal:

  ```
  AC0 = AC0 + (AC1 << T0)
  ```

  while both of these instructions are illegal:

  ```
  AC0 = AC0 + ((AC1 << T0))
  AC0 = AC0 + AC1 << T0
  ```

1.4.2 Rules

- Simple instructions are not allowed to span multiple lines. One exception, single instructions that use the “,” notation to imply parallelism. These instructions may be split up following the “,” notation.

  The following example shows a single instruction (dual multiply) occupying two lines:

  ```
  ACx = m40(rnd(uns(Xmem) * uns(coef(Cmem)))),
  ACy = m40(rnd(uns(Ymem) * uns(coef(Cmem))))
  ```

- User-defined parallelism instructions (using || notation) are allowed to span multiple lines. For example, all of the following instructions are legal:

  ```
  AC0 = AC1  ||  AC2 = AC3

  AC0 = AC1  ||
  AC2 = AC3

  AC0 = AC1
  || AC2 = AC3

  AC0 = AC1
  ||
  AC2 = AC3
  ```

- The block repeat syntax uses braces to delimit the block that is to be repeated:

```
blockrepeat {
       instr
        instr
          :
        instr
     }
```

```
localrepeat {
        instr
        instr
           :
        instr
     }
```

The left opening brace must appear on the same line as the repeat keyword. The right closing brace must appear alone on a line (trailing comments allowed).

Note that a label placed just inside the closing brace of the loop is effectively outside the loop. The following two code sequences are equivalent:

```
localrepeat {
        instr1
        instr2
   Label:
       }
       instr3
```

and

```
localrepeat {
        instr1
        instr2
       }
    Label:
       instr3
```

A label is the address of the first construct following the label that gets assembled into code in the object file. A closing brace does not generate any code and so the label marks the address of the first instruction that generates code, that is, instr3.

In this example, “goto Label” exits the loop, which is somewhat unintuitive:

```
localrepeat {
       goto Label
       instr2
   Label:
       }
       instr3
```

1.4.2.1 Reserved Words

Register names and algebraic syntax keywords are reserved. They may not be used as names of identifiers, labels, etc.

1.4.2.2 Literal and Address Operands

Literals in the algebraic strings are denoted as K or k fields. In the Smem address modes that require an offset, the offset is also a literal (K16 or k3). 8-bit and 16-bit literals are allowed to be linktime-relocatable; for other literals, the value must be known at assembly time.

Addresses are the elements of the algebraic strings denoted by P, L, and l. Further, 16-bit and 24-bit absolute address Smem modes are addresses, as is the dma Smem mode, denoted by the ‘@’ syntax. Addresses may be assembly-time constants or symbolic linktime-known constants or expressions.

Both literals and addresses follow syntax rule 1. For addresses only, rules 2 and 3 also apply.

Rule 1

A valid address or literal is a # followed by one of the following:

❑ a number `(#123)`

❑ an identifier `(#FOO)`

❑ a parenthesized expression `(#(FOO + 2))`

Note that # is not used inside the expression.

Rule 2

When an address is used in a dma, the address does not need to have a leading #, be it a number, a symbol or an expression. These are all legal:

```
@#123
@123
@#foo
@foo
@#(foo+2)
@(foo+2)
```

Rule 3

When used in contexts other than dma (such as branch targets or Smem-absolute address), addresses generally need a leading #. As a convenience, the # may be omitted in front of an identifier. These are all legal:

Branch	Absolute Address
`goto #123`	`*(#123)`
`goto #foo`	`*(#foo)`
`goto foo`	`*(foo)`
`goto #(foo+2)`	`*(#(foo+2))`

These are illegal:

```
goto 123            *(123)
goto (foo+2)        *((foo+2))
```

1.4.2.3 Memory Operands

- Syntax of Smem is the same as that of Lmem or Baddr.
- In the following instruction syntaxes, Smem **cannot** reference to a memory-mapped register (MMR). No instruction can access a byte within a memory-mapped register. If Smem is an MMR in one of the following syntaxes, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

```
dst = uns(high_byte(Smem))
dst = uns(low_byte(Smem))
ACx = low_byte(Smem) << #SHIFTW
ACx = high_byte(Smem) << #SHIFTW
high_byte(Smem) = src
low_byte(Smem) = src
```

- Syntax of Xmem is the same as that of Ymem.
- Syntax of coefficient operands, Cmem:

  `*CDP`

  `*CDP+`

  `*CDP−`

  `*(CDP + T0)`, when C54CM = 0

  `*(CDP + AR0)`, when C54CM = 1

  When an instruction uses a Cmem operand with paralleled instructions, the pointer modification of the Cmem operand must be the same for both instructions of the paralleled pair or the assembler generates an error. For example:

```
AC0 = AC0 + (*AR2+ * coef(*CDP+)),
AC1 = AC1 + (*AR3+ * coef(*CDP+))
```

- ❑ An optional mmr prefix is allowed to be specified for indirect memory operands, for example, `mmr(*AR0)`. This is an assertion by you that this is an access to a memory-mapped register. The assembler checks whether such access is legal in given circumstances.

  The mmr prefix is supported for Xmem, Ymem, indirect Smem, indirect Lmem, and Cmem operands. It is not supported for direct memory operands; it is expected that an explicit mmap() parallel instruction is used in conjunction with direct memory operands to indicate MMR access.

  Note that the mmr prefix is part of the syntax. It is an implementation restriction that mmr cannot exchange positions with other prefixes around the memory operand, such as dbl or uns. If several prefixes are specified, mmr must be the innermost prefix. Thus, `uns(mmr(*AR0))` is legal, but `mmr(uns(*AR0))` is not legal.

- ❑ The following indirect operands **cannot** be used for accesses to I/O space. An instruction using one of these operands requires a 2-byte extension for the constant. This extension would prevent the use of the port() qualifier needed to indicate an I/O-space access.

  ```
  *ARn(#K16)

  *+ARn(#K16)

  *CDP(#K16)

  *+CDP(#K16)
  ```

  Also, the following instructions that include the delay operation cannot be used for accesses to I/O space:

  ```
  delay(Smem)

  ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem],
  delay(Smem)
  ```

  Any illegal access to I/O space will generate a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

1.4.2.4 Operand Modifiers

Operand modifiers look like function calls on operands. Note that uns is an operand modifier and an instruction modifier meaning unsigned. The operand modifier uns is used when the operand is modified on the way to the rest of the operation (multiply-and-accumulate). The instruction modifier uns is used when the whole operation is affected (multiply, register compare, compare and branch).

Modifier	Meaning
dbl	Access a true 32-bit memory operand
dual	Access a 32-bit memory operand for use as two independent 16-bit halves of the given operation
HI	Access upper 16 bits of the accumulator
high_byte	Access the high byte of the memory location
LO	Access lower 16 bits of the accumulator
low_byte	Access the low byte of the memory location
pair	Dual register access
rnd	Round
saturate	Saturate
uns	Unsigned operand

When an instruction uses a Cmem operand with paralleled instructions and the Cmem operand is defined as unsigned (uns), both Cmem operands of the paralleled pair must be defined as unsigned (and reciprocally).

When an instruction uses both Xmem and Ymem operands with paralleled instructions and the Xmem operand is defined as unsigned (uns), Ymem operand must also be defined as unsigned (and reciprocally).

1.4.2.5 Operator Syntax Rules

Instructions that read and write the same operand can also be written in op-assign form. For example:

```
AC0 = AC0 + *AR4
```

can also be written:

```
AC0 += *AR4
```

This form is supported for these operations: +=, −=, &=, |=, ^=

Note that in certain instances use of op-assign notation results in ambiguous algebraic assembly. This happens if the op-assign operator is not delimited by white space, for example:

`*AR0+=#4` is ambiguous, is it `*AR0 += #4` or `*AR0+ = #4` ?

The assembler always parses adjacent += as plus-assign; therefore, this instructions is parsed as `*AR0 += #4`.

`*AR0+=*AR1` is ambiguous, is it `*AR0 += *AR1` or `*AR0+ =*AR1` ?

Once again, the first form, `*AR0 += *AR1`, is used. This is not a valid instruction −− an error is printed.

1.5 Nonrepeatable Instructions

Table 1−4 lists the instructions that cannot be used in a repeatable instruction.

Table 1−4. Nonrepeatable Instructions

Instruction Description	Algebraic Syntax That Cannot Be Repeated

Branch Conditionally	if (cond) goto l4
	if (cond) goto L8
	if (cond) goto L16
	if (cond) goto P24

Branch Unconditionally	goto ACx goto L7 goto L16 goto P24
Branch on Auxiliary Register Not Zero	if (ARn_mod != #0) goto L16

Call Conditionally	if (cond) call L16
	if (cond) call P24

Call Unconditionally	call ACx call L16 call P24
Clear Status Register Bit	bit(STx, k4) = #0
Compare and Branch	compare (uns(src RELOP K8)) goto L8

Execute Conditionally	if (cond) execute(AD_Unit)
	if (cond) execute(D_Unit)

Idle	idle

Load CPU Register from Memory	DP = Smem
	RETA = dbl(Lmem)

Load CPU Register with Immediate Value	DP = k16
Move CPU Register Content to Auxiliary or Temporary Register	TAx = RPTC

Repeat Block of Instructions Unconditionally	localrepeat{}
	blockrepeat{}

Repeat Single Instruction Conditionally	while (cond && (RPTC < k8)) repeat

Table 1−4. Nonrepeatable Instructions (Continued)

Instruction Description	Algebraic Syntax That Cannot Be Repeated

Repeat Single Instruction Unconditionally	repeat(k8)
	repeat(k16)
	repeat(CSR)

Repeat Single Instruction Unconditionally and Decrement CSR	repeat(CSR), CSR –= k4

Repeat Single Instruction Unconditionally and Increment CSR	repeat(CSR), CSR += TAx
	repeat(CSR), CSR += k4

Return Conditionally	if (cond) return
Return Unconditionally	return
Return from Interrupt	return_int
Round Accumulator Content	ACy = rnd(ACx)
Set Status Register Bit	bit(STx, k4) = #1
Software Interrupt	intr(k5)
Software Reset	reset
Software Trap	trap(k5)
Store CPU Register Content to Memory	dbl(Lmem) = RETA

Chapter 2

Parallelism Features and Rules

This chapter describes the parallelism features and rules of the TMS320C55x™ DSP algebraic instruction set.

Topic **Page**

2.1 Parallelism Features

The C55x™ DSP architecture enables you to execute two instructions in parallel within the same cycle of execution. The types of parallelism are:

- Built-in parallelism within a single instruction.

  Some instructions perform two different operations in parallel. A comma is used to separate the two operations. This type of parallelism is also called implied parallelism. For example:

  ```
  AC0 = *AR0 * coef(*CDP),
  AC1 = *AR1 * coef(*CDP)
  ```

  This is a single instruction. The data referenced by AR0 is multiplied by the coefficient referenced by CDP. At the same time, the data referenced by AR1 is multiplied by the same coefficient (CDP).

- User-defined parallelism between two instructions.

  Two instructions may be paralleled by you or the C compiler. The parallel bars, **||**, are used to separate the two instructions to be executed in parallel. For example:

  ```
  AC1 = *AR1- * *AR2+
  || T1 = T1 ^ AR2
  ```

  The first instruction performs a multiplication in the D-unit. The second instruction performs a logical operation in the A-unit ALU.

- Built-in parallelism can be combined with user-defined parallelism. Parenthesis separators can be used to determine boundaries of the two instructions. For example:

  ```
  (AC2 = *AR3+ * AC1,
  T3 = *AR3+)
  || AR1 = #5
  ```

  The first instruction includes implied parallelism. The second instruction is paralleled by you.

2.2 Parallelism Basics

In the parallel pair, all of these constraints must be met:

- ❑ No resource conflicts as detailed in section 2.3.
- ❑ One instruction must have a parallel enable bit or the pair must qualify for soft-dual parallelism as detailed in section 2.4.
- ❑ No memory operand may use an addressing mode that requires a constant that is 16 bits or larger:
  - ■ *abs16(#k16)
  - ■ *(#k23)
  - ■ *port(#k16)
  - ■ *ARn(K16)
  - ■ *+ARn(K16)
  - ■ *CDP(K16)
  - ■ *+CDP(K16)
- ❑ The following instructions cannot be in parallel:
  - ■ `if (cond) goto P24`
  - ■ `if (cond) call P24`
  - ■ `idle`
  - ■ `intr(k5)`
  - ■ `reset`
  - ■ `trap(k5)`
- ❑ Neither instruction in the parallel pair can use any of these instruction or operand modifiers:
  - ■ `circular()`
  - ■ `linear()`
  - ■ `mmap()`
  - ■ `readport()`
  - ■ `writeport()`
- ❑ A particular register or memory location can only be written once per pipeline phase. Violations of this rule take many forms. Loading the same register twice is a simple case. Other cases include:
  - ■ Conflicting address mode modifications (for example, *AR2+ versus *AR2−)
  - ■ Combining a SWAP instruction (modifies all of its registers) with any other instruction that writes one of the same registers

- ❑ Data stack pointer (XSP) or system stack pointer (XSSP) modifications cannot be combined with any of the following instructions:
  - ■ Call Conditionally, (if (cond) call instructions)
  - ■ Call Unconditionally, (call instructions)
  - ■ Push to top of Stack (push instructions)
  - ■ Pop from top of Stack (pop instructions)
  - ■ Return Conditionally, (if (cond) return instructions)
  - ■ Return Unconditionally, (return instructions)
  - ■ Return from Interrupt, (return_int, instructions)
  - ■ trap or intr instructions
- ❑ When both instructions in a parallel pair modify a status bit, the value of that status bit becomes undefined.

2.3 Resource Conflicts

Every instruction uses some set of operators, address generation units, and buses, collectively called resources, while executing. To determine which resources are used by a specific instruction, see Table 4−1. Two instructions in parallel use all the resources of the individual instructions. A resource conflict occurs when two instructions use a combination of resources that is not supported on the C55x device. This section details the resource conflicts.

2.3.1 Operators

You may use each of these operators only once:

- ❑ D Unit ALU
- ❑ D Unit Shift
- ❑ D Unit Swap
- ❑ A Unit Swap
- ❑ A Unit ALU
- ❑ P Unit

For an instruction that uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.

2.3.2 Address Generation Units

You may use no more than the indicated number of data address generation units:

- ❑ 2 Data Address (DA) Generation Units
- ❑ 1 Coefficient Address (CA) Generation Unit
- ❑ 1 Stack Address (SA) Generation Unit

2.3.3 Buses

You may use no more than the indicated number of buses:

- ❑ 2 Data Read (DR) Buses
- ❑ 1 Coefficient Read (CR) Bus
- ❑ 2 Data Write (DW) Buses
- ❑ 1 ACB Bus − brings D-unit registers to A-unit and P-unit operators

2.4 Soft-Dual Parallelism

Instructions that reference memory operands do not have parallel enable bits. Two such instructions may still be combined with a type of parallelism called soft-dual parallelism. The constraints of soft-dual parallelism are:

- ❑ Both memory operands must meet the constraints of the dual AR indirect addressing mode (Xmem and Ymem), as described in section 3.4.2. The operands available for the dual AR indirect addressing mode are:
  - *ARn
  - *ARn+
  - *ARn–
  - *(ARn + AR0)
  - *(ARn + T0)
  - *(ARn – AR0)
  - *(ARn – T0)
  - *ARn(AR0)
  - *ARn(T0)
  - *(ARn + T1)
  - *(ARn – T1)
- ❑ Neither instruction can contain any of the following:
  - Instructions embedding high_byte(Smem) and low_byte(Smem).
    - `dst = uns(high_byte(Smem))`
    - `dst = uns(low_byte(Smem))`
    - `ACx = low_byte(Smem) << #SHIFTW`
    - `ACx = high_byte(Smem) << #SHIFTW`
    - `high_byte(Smem) = src`
    - `low_byte(Smem) = src`

- These instructions that read and write the same memory location:
  - `cbit(Smem, src)`
  - `bit(Smem, src) = #0`
  - `bit(Smem, src) = #1`
  - `TCx = bit(Smem, k4), bit(Smem, k4) = #1`
  - `TCx = bit(Smem, k4), bit(Smem, k4) = #0`
  - `TCx = bit(Smem, k4), cbit(Smem, k4)`

❑ With regard to soft-dual parallelism, the `mar(Smem)` and `XAdst = mar(Smem)` instructions have the same properties as any memory reference instruction.

2.4.1 Soft-Dual Parallelism of MAR Instructions

Although the following modify auxiliary register (MAR) instructions do not reference memory and do not have parallel enable bits, they may be combined together or with any other memory reference instructions (not limited to Xmem/ Ymem) to form soft-dual parallelism.

❑ `mar(TAy + TAx)`
❑ `mar(TAx + k8)`
❑ `mar(TAy = TAx)`
❑ `mar(TAx = k8)`
❑ `mar(TAy – TAx)`
❑ `mar(TAx – k8)`

Note that this is not the full list of MAR instructions; instructions `mar(TAx = D16)` is not included.

2.5 Execute Conditionally Instructions

The parallelization of the execute conditionally, if (cond) execute, instructions does not adhere to the descriptions in this chapter. All of the specific instances of legal parallelism are covered in the execute conditionally descriptions in Chapter 5.

2.6 Other Exceptions

The following are other exceptions not covered elsewhere in this chapter.

❑ An instruction that reads the repeat counter register (RPTC) may not be combined with any single-repeat instruction:

- `repeat()`
- `repeat(CSR)`
- `while (cond) repeat`

Chapter 3

Introduction to Addressing Modes

This chapter provides an introduction to the addressing modes of the TMS320C55x™ DSP.

Topic Page

3.1 Introduction to the Addressing Modes

The TMS320C55x DSP supports three types of addressing modes that enable flexible access to data memory, to memory-mapped registers, to register bits, and to I/O space:

- ❑ The absolute addressing mode allows you to reference a location by supplying all or part of an address as a constant in an instruction.
- ❑ The direct addressing mode allows you to reference a location using an address offset.
- ❑ The indirect addressing mode allows you to reference a location using a pointer.

Each addressing mode provides one or more types of operands. An instruction that supports an addressing-mode operand has one of the following syntax elements listed in Table 3−1.

Table 3−1. Addressing-Mode Operands

Syntax Element(s)	Description
Baddr	When an instruction contains Baddr, that instruction can access one or two bits in an accumulator (AC0–AC3), an auxiliary register (AR0–AR7), or a temporary register (T0–T3). Only the register bit test/set/clear/complement instructions support Baddr. As you write one of these instructions, replace Baddr with a compatible operand.
Cmem	When an instruction contains Cmem, that instruction can access a single word (16 bits) of data from data memory. As you write the instruction, replace Cmem with a compatible operand.
HI(Cmem)/ LO(Cmem)	When an instruction contains HI(Cmem)/LO(Cmem), that instruction can access a long word (32 bits) of data from data memory. As you write the instruction, replace Cmem with a compatible operand.
Lmem	When an instruction contains Lmem, that instruction can access a long word (32 bits) of data from data memory or from a memory-mapped registers. As you write the instruction, replace Lmem with a compatible operand.
Smem	When an instruction contains Smem, that instruction can access a single word (16 bits) of data from data memory, from I/O space, or from a memory-mapped register. As you write the instruction, replace Smem with a compatible operand.
Xmem and Ymem	When an instruction contains Xmem and Ymem, that instruction can perform two simultaneous 16-bit accesses to data memory. As you write the instruction, replace Xmem and Ymem with compatible operands.

3.2 Absolute Addressing Modes

Table 3−2 lists the absolute addressing modes available.

Table 3−2. Absolute Addressing Modes

Addressing Mode	Description
k16 absolute	This mode uses the 7-bit register called DPH (high part of the extended data page register) and a 16-bit unsigned constant to form a 23-bit data-space address. This mode is used to access a memory location or a memory-mapped register.
k23 absolute	This mode enables you to specify a full address as a 23-bit unsigned constant. This mode is used to access a memory location or a memory-mapped register.
I/O absolute	This mode enables you to specify an I/O address as a 16-bit unsigned constant. This mode is used to access a location in I/O space.

3.2.1 k16 Absolute Addressing Mode

The k16 absolute addressing mode uses the operand *abs16(#k16), where k16 is a 16-bit unsigned constant. DPH (the high part of the extended data page register) and k16 are concatenated to form a 23-bit data-space address. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

3.2.2 k23 Absolute Addressing Mode

The k23 absolute addressing mode uses the *(#k23) operand, where k23 is a 23-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 3-byte extension to the instruction (the most-significant bit of this 3-byte extension is discarded). Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

Instructions using the operand *(#k23) to access the memory operand Smem cannot be used in a repeatable instruction. See Table 1−4 for a list of these instructions.

3.2.3 I/O Absolute Addressing Mode

The I/O absolute addressing mode uses the *port(#k16) operand, where k16 is a 16-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction. The `delay()` instruction cannot use this mode.

3.3 Direct Addressing Modes

Table 3−3 lists the direct addressing modes available.

Table 3−3. Direct Addressing Modes

Addressing Mode	Description
DP direct	This mode uses the main data page specified by DPH (high part of the extended data page register) in conjunction with the data page register (DP). This mode is used to access a memory location or a memory-mapped register.
SP direct	This mode uses the main data page specified by SPH (high part of the extended stack pointers) in conjunction with the data stack pointer (SP). This mode is used to access stack values in data memory.
Register-bit direct	This mode uses an offset to specify a bit address. This mode is used to access one register bit or two adjacent register bits.
PDP direct	This mode uses the peripheral data page register (PDP) and an offset to specify an I/O address. This mode is used to access a location in I/O space.

The DP direct and SP direct addressing modes are mutually exclusive. The mode selected depends on the CPL bit in status register ST1_55:

CPL	Addressing Mode Selected
0	DP direct addressing mode
1	SP direct addressing mode

The register-bit and PDP direct addressing modes are independent of the CPL bit.

3.3.1 DP Direct Addressing Mode

When an instruction uses the DP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from DPH that selects one of the 128 main data pages (0 through 127). The 16 LSBs are the sum of two values:

- ❑ The value in the data page register (DP). DP identifies the start address of a 128-word local data page within the main data page. This start address can be any address within the selected main data page.
- ❑ A 7-bit offset (Doffset) calculated by the assembler. The calculation depends on whether you are accessing data memory or a memory-mapped register (using the mmap() qualifier).

The concatenation of DPH and DP is called the extended data page register (XDP). You can load DPH and DP individually, or you can use an instruction that loads XDP.

3.3.2 SP Direct Addressing Mode

When an instruction uses the SP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from SPH. The 16 LSBs are the sum of the SP value and a 7-bit offset that you specify in the instruction. The offset can be a value from 0 to 127. The concatenation of SPH and SP is called the extended data stack pointer (XSP). You can load SPH and SP individually, or you can use an instruction that loads XSP.

On the first main data page, addresses 00 0000h–00 005Fh are reserved for the memory-mapped registers. If any of your data stack is in main data page 0, make sure it uses only addresses 00 0060h–00 FFFFh on that page.

3.3.3 Register-Bit Direct Addressing Mode

In the register-bit direct addressing mode, the offset you supply in the operand, @bitoffset, is an offset from the LSB of the register. For example, if bitoffset is 0, you are addressing the LSB of a register. If bitoffset is 3, you are addressing bit 3 of the register.

Only the register bit test/set/clear/complement instructions support this mode. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).

3.3.4 PDP Direct Addressing Mode

When an instruction uses the PDP direct addressing mode, a 16-bit I/O address is formed. The 9 MSBs are taken from the 9-bit peripheral data page register (PDP) that selects one of the 512 peripheral data pages (0 through 511). Each page has 128 words (0 to 127). You select a particular word by specifying a 7-bit offset (Poffset) in the instruction. For example, to access the first word on a page, use an offset of 0.

You must use a readport() or writeport() instruction qualifier to indicate that you are accessing an I/O-space location rather than a data-memory location. You place the readport() or the writeport() instruction qualifier in parallel with the instruction that performs the I/O-space access.

3.4 Indirect Addressing Modes

Table 3−4 list the indirect addressing modes available. You may use these modes for linear addressing or circular addressing.

Table 3−4. Indirect Addressing Modes

Addressing Mode	Description
AR indirect	This mode uses one of eight auxiliary registers (AR0–AR7) to point to data. The way the CPU uses the auxiliary register to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Dual AR indirect	This mode uses the same address-generation process as the AR indirect addressing mode. This mode is used with instructions that access two or more data-memory locations.
CDP indirect	This mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Coefficient indirect	This mode uses the same address-generation process as the CDP indirect addressing mode. This mode is available to support instructions that can access a coefficient in data memory at the same time they access two other data-memory values using the dual AR indirect addressing mode.

3.4.1 AR Indirect Addressing Mode

The AR indirect addressing mode uses an auxiliary register ARn (n = 0, 1, 2, 3, 4, 5, 6, or 7) to point to data. The way the CPU uses ARn to generate an address depends on the access type:

For An Access To ...	ARn Contains ...
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by ARnH, which is the high part of extended auxiliary register XARn. For accesses to data space, use an instruction that loads XARn; ARn can be individually loaded, but ARnH cannot be loaded.
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/complement instructions support AR indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).
I/O space	A 16-bit I/O address.

The AR indirect addressing-mode operand available depends on the ARMS bit of status register ST2_55:

ARMS	DSP Mode or Control Mode
0	DSP mode. The CPU can use the list of DSP mode operands (Table 3−5), which provide efficient execution of DSP-intensive applications.
1	Control mode. The CPU can use the list of control mode operands (Table 3−6), which enable optimized code size for control system applications.

Table 3−5 (page 3-8) introduces the DSP operands available for the AR indirect addressing mode. Table 3−6 (page 3-12) introduces the control mode operands. When using the tables, keep in mind that:

- ❑ Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- ❑ All additions to and subtractions from the pointers are done modulo 8M. ARnH is updated by the hardware when the pointer modification crosses the main data pages’ boundary.

Table 3−5. DSP Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated: If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn–	ARn is decremented after the address is generated: If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*+ARn	ARn is incremented before the address is generated: If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*–ARn	ARn is decremented before the address is generated: If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)

Table 3−5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer. This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer. This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(T1)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T1 is used as an offset from that base pointer.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)

Table 3−5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: ARn = ARn + T1	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn after the address is generated: ARn = ARn – T1	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + AR0B)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0 (The addition is done with reverse carry propagation) This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time. Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + T0B)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0 (The addition is done with reverse carry propagation) This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time. Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)

Table 3−5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – AR0B)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0 (The subtraction is done with reverse carry propagation) This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time. Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – T0B)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 (The subtraction is done with reverse carry propagation) This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time. Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer. Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr)
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16 Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr)

Table 3−6. Control Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated: If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn–	ARn is decremented after the address is generated: If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem) Memory-mapped register Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)

Table 3−6. Control Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer. This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer. This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer. Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr)

Table 3−6. Control Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16 Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr)
*ARn(short(#k3))	ARn is not modified. ARn is used as a base pointer. The 3-bit unsigned constant (k3) is used as an offset from that base pointer. k3 is in the range 1 to 7.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register bit (Baddr) I/O-space (Smem)

3.4.2 Dual AR Indirect Addressing Mode

The dual AR indirect addressing mode enables you to make two data-memory accesses through the eight auxiliary registers, AR0–AR7. As with single AR indirect accesses to data space, the CPU uses an extended auxiliary register to create each 23-bit address. You can use linear addressing or circular addressing for each of the two accesses.

You may use the dual AR indirect addressing mode for:

❑ Executing an instruction that makes two 16-bit data-memory accesses. In this case, the two data-memory operands are designated in the instruction syntax as Xmem and Ymem. For example:

```
ACx = (Xmem << #16) + (Ymem << #16)
```

❑ Executing two instructions in parallel. In this case, both instructions must each access a single memory value, designated in the instruction syntaxes as Smem or Lmem. For example:

```
dst = Smem
|| dst = src & Smem
```

The operand of the first instruction is treated as an Xmem operand, and the operand of the second instruction is treated as a Ymem operand.

The available dual AR indirect operands are a subset of the AR indirect operands. The ARMS status bit does not affect the set of dual AR indirect operands available.

Note:

The assembler rejects code in which dual operands use the same auxiliary register with two different auxiliary register modifications. You can use the same ARn for both operands, if one of the operands is *ARn or *ARn(T0); neither modifies ARn.

Table 3−7 (page 3-15) introduces the operands available for the dual AR indirect addressing mode. Note that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- All additions to and subtractions from the pointers are done modulo 8M. ARnH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3−7. Dual AR Indirect Operands

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn+	ARn is incremented after the address is generated: If 16-bit operation: ARn = ARn + 1 If 32-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn–	ARn is decremented after the address is generated: If 16-bit operation: ARn = ARn – 1 If 32-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)

Table 3−7. Dual AR Indirect Operands (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer. This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer. This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: ARn = ARn + T1	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn after the address is generated: ARn = ARn – T1	Data-memory (Smem, Lmem, Xmem, Ymem)

3.4.3 CDP Indirect Addressing Mode

The CDP indirect addressing mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on the access type:

For An Access To ...	CDP Contains ...
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by CDPH, the high part of the extended coefficient data pointer (XCDP).
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/complement instructions support CDP indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).
I/O space	A 16-bit I/O address.

Table 3−8 (page 3-17) introduces the operands available for the CDP indirect addressing mode. Note that:

❑ Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP.

❑ All additions to and subtractions from the pointers are done modulo 8M. CDPH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3−8. CDP Indirect Operands

Operand	Pointer Modification	Supported Access Types
*CDP	CDP is not modified.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register-bit (Baddr) I/O-space (Smem)
*CDP+	CDP is incremented after the address is generated: If 16-bit/1-bit operation: CDP = CDP + 1 If 32-bit/2-bit operation: CDP = CDP + 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register-bit (Baddr) I/O-space (Smem)

Table 3−8. CDP Indirect Operands (Continued)

Operand	Pointer Modification	Supported Access Types
*CDP–	CDP is decremented after the address is generated: If 16-bit/1-bit operation: CDP = CDP – 1 If 32-bit/2-bit operation: CDP = CDP – 2	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register-bit (Baddr) I/O-space (Smem)
*CDP(#K16)	CDP is not modified. CDP is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer. Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register-bit (Baddr)
*+CDP(#K16)	The 16-bit signed constant (K16) is added to CDP before the address is generated: CDP = CDP + K16 Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem) Register-bit (Baddr)

3.4.4 Coefficient Indirect Addressing Mode

The coefficient indirect addressing mode uses the same address-generation process as the CDP indirect addressing mode for data-space accesses. The coefficient indirect addressing mode is supported by select memory-to-memory move and memory initialization instructions and by the following arithmetical instructions:

❑ Dual multiply (accumulate/subtract)
❑ Finite impulse response filter
❑ Multiply
❑ Multiply and accumulate
❑ Multiply and subtract

Instructions using the coefficient indirect addressing mode to access data are mainly instructions performing operations with three memory operands per cycle. Two of these operands (Xmem and Ymem) are accessed with the dual AR indirect addressing mode. The third operand (Cmem) is accessed with the coefficient indirect addressing mode. The Cmem operand is carried on the BB bus.

Keep the following facts about the BB bus in mind as you use the coefficient indirect addressing mode:

❑ The BB bus is not connected to external memory. If a Cmem operand is accessed through the BB bus, the operand must be in internal memory.

❑ Although the following instructions access Cmem operands, they do not use the BB bus to fetch the 16-bit or 32-bit Cmem operand.

Instruction Syntax	Description of Cmem Access	Bus Used to Access Cmem
Smem = Cmem	16-bit read from Cmem	DB
Cmem = Smem	16-bit write to Cmem	EB
Lmem = dbl(Cmem)	32-bit read from Cmem	CB for most significant word (MSW) DB for least significant word (LSW)
dbl(Cmem) = Lmem	32-bit write to Cmem	FB for MSW EB for LSW

Consider the following instruction syntax. In one cycle, two multiplications can be performed in parallel. One memory operand (Cmem) is common to both multiplications, while dual AR indirect operands (Xmem and Ymem) are used for the other values in the multiplication.

```
ACx = Xmem * Cmem,
ACy = Ymem * Cmem
```

To access three memory values (as in the above example) in a single cycle, the value referenced by Cmem must be located in a memory bank different from the one containing the Xmem and Ymem values.

Table 3−9 introduces the operands available for the coefficient indirect addressing mode. Note that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP.
- All additions to and subtractions from the pointers are done modulo 8M. CDPH is updated by the hardware when the pointer modification crosses the main data pages' boundary.

Table 3−9. Coefficient Indirect Operands

Operand	Pointer Modification	Supported Access Type
*CDP	CDP is not modified.1	Data-memory
*CDP+	CDP is incremented after the address is generated: If 16-bit operation: CDP = CDP + 1 If 32-bit operation: CDP = CDP + 2	Data-memory
*CDP–	CDP is decremented after the address is generated: If 16-bit operation: CDP = CDP – 1 If 32-bit operation: CDP = CDP – 2	Data-memory
*(CDP + AR0)	The 16-bit signed constant in AR0 is added to CDP after the address is generated: CDP = CDP + AR0 This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory
*(CDP + T0)	The 16-bit signed constant in T0 is added to CDP after the address is generated: CDP = CDP + T0 This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory

3.5 Circular Addressing

Circular addressing can be used with any of the indirect addressing modes. Each of the eight auxiliary registers (AR0–AR7) and the coefficient data pointer (CDP) can be independently configured to be linearly or circularly modified as they act as pointers to data or to register bits, see Table 3−10. This configuration is done with a bit (ARnLC) in status register ST2_55. To choose circular modification, set the bit.

Table 3−10. Circular Addressing Pointers

Pointer	Linear/Circular Configuration Bit	Supplier of Main Data Page	Buffer Start Address Register	Buffer Size Register
AR0	ST2_55(0) = AR0LC	AR0H	BSA01	BK03
AR1	ST2_55(1) = AR1LC	AR1H	BSA01	BK03
AR2	ST2_55(2) = AR2LC	AR2H	BSA23	BK03
AR3	ST2_55(3) = AR3LC	AR3H	BSA23	BK03
AR4	ST2_55(4) = AR4LC	AR4H	BSA45	BK47
AR5	ST2_55(5) = AR5LC	AR5H	BSA45	BK47
AR6	ST2_55(6) = AR6LC	AR6H	BSA67	BK47
AR7	ST2_55(7) = AR7LC	AR7H	BSA67	BK47
CDP	ST2_55(8) = CDPLC	CDPH	BSAC	BKC

Each auxiliary register ARn has its own linear/circular configuration bit in ST2_55:

ARnLC	ARn Is Used For ...
0	Linear addressing
1	Circular addressing

The CDPLC bit in status register ST2_55 configures the DSP to use CDP for linear addressing or circular addressing:

CDPLC	CDP Is Used For ...
0	Linear addressing
1	Circular addressing

You can use the circular addressing instruction qualifier, circular(), if you want every pointer used by the instruction to be modified circularly, just add the circular() qualifier in parallel with the instruction. The circular addressing instruction qualifier overrides the linear/circular configuration in ST2_55.

Chapter 4

Instruction Set Summary

This chapter provides a summary of the TMS320C55x™ DSP algebraic instruction set (Table 4−1). With each instruction, you will find the availability of a parallel enable bit, word count (size), cycle time, what pipeline phase the instruction executes, in what operator unit the instruction executes, how many of each address generation unit is used, and how many of each bus is used.

Table 4−1 does not list all of the resources that may be used by an instruction, it only lists those that may result in a resource conflict, and thus prevent two instructions from being in parallel. If an instruction lists nothing in a particular column, it means that particular resource will never be in conflict for that instruction.

The column heads of Table 4−1 are:

❑ Instruction: In cases where the resource usage of an instruction varies with the kinds of registers, you see the notation <name>-AU for A-unit registers and <name>-DU for D-unit registers. So, dst-AU is a destination that is an A-unit register and src-DU is a source that is a D-unit register. In the few cases where that notation is insufficient, you see the cases listed in the Notes column.

❑ E: Whether that instruction has a parallel enable bit

❑ S: The size of the instruction in bytes

❑ C: Number of cycles required for the instruction

❑ Pipe: The pipeline phase in which the instruction executes:

Name	**Phase**
AD	Address
D	Decode
R	Read
X	Execute

❑ Operator: Which operator(s) are used by this instruction. When an instruction uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.

- Address Generation Unit: How many of each address generation unit is used. The address generation units are:

Name	**Unit**
DA	Data Address Generation Unit
CA	Coefficient Address Generation Unit
SA	Stack Address Generation Unit

- Buses: How many of each bus is used. The buses are:

Name	**Bus**
DR	Data Read
CR	Coefficient Read
DW	Data Write
ACB	Brings D unit registers to A unit and P unit operators

Table 4−1. Algebraic Instruction Set Summary

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
	Absolute Distance (page 5-2)													
	abdst(Xmem, Ymem, ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	.	.	2	.	.	.	
	Absolute Value (page 5-4)													
	dst-AU = |src-AU|	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = |src-DU|	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = |src|	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
	Addition (page 5-7)													
[1]	dst-AU = dst-AU + src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = dst-AU + src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = dst-DU + src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[2]	dst-AU = dst-AU + k4	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-DU = dst-DU + k4	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	
[3]	dst-AU = src-AU + K16	N	4	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU + K16	N	4	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src + K16	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[4]	dst-AU = src-AU + Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = src-DU + Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = src + Smem	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.
[5]	ACy = ACy + (ACx << Tx)	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[6]	ACy = ACy + (ACx << #SHIFTW)	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[7]	ACy = ACx + (K16 << #16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[8]	ACy = ACx + (K16 << #SHFT)	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[9]	ACy = ACx + (Smem << Tx)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[10]	ACy = ACx + (Smem << #16)	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[11]	ACy = ACx + uns(Smem) + CARRY	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[12]	ACy = ACx + uns(Smem)	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[13]	ACy = ACx + (uns(Smem) << #SHIFTW)	N	4	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[14]	ACy = ACx + dbl(Lmem)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[15]	ACx = (Xmem << #16) + (Ymem << #16)	N	3	1	X	DU_ALU	2	.	.	2	.	.	.	
[16]	Smem = Smem + K16	N	4	1	X	DU_ALU	1	.	.	1	.	1	.	
Addition with Absolute Value (page 5-27)														
	ACy = rnd(ACy + |ACx|)	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
Addition with Parallel Store Accumulator Content to Memory (page 5-29)														
	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	N	4	1	X	DU_ALU + DU_SHIFT	2	.	.	2	.	2	.	
Addition or Subtraction Conditionally (page 5-31)														
[1]	ACy = adsc(Smem, ACx, TC1)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[2]	ACy = adsc(Smem, ACx, TC2)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
Addition or Subtraction Conditionally with Shift (page 5-33)														
	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	.
Addition, Subtraction, or Move Accumulator Content Conditionally (page 5-36)														
	ACy = adsc(Smem, ACx, TC1, TC2)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
Bitwise AND (page 5-38)														
[1]	dst-AU = dst-AU & src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = dst-AU & src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = dst-DU & src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[2]	dst-AU = src-AU & k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU & k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src & k8	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[3]	dst-AU = src-AU & k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU & k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src & k16	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[4]	dst-AU = src-AU & Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = src-DU & Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = src & Smem	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.
[5]	ACy = ACy & (ACx <<< #SHIFTW)	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[6]	ACy = ACx & (k16 <<< #16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[7]	ACy = ACx & (k16 <<< #SHFT)	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[8]	Smem = Smem & k16	N	4	1	X	AU_ALU	1	.	.	1	.	1	.	

Bitwise AND Memory with Immediate Value and Compare to Zero (page 5-47)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	TC1 = Smem & k16	N	4	1	X	AU_ALU	1	.	.	1	.	.	.	
[2]	TC2 = Smem & k16	N	4	1	X	AU_ALU	1	.	.	1	.	.	.	

Bitwise OR (page 5-48)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	dst-AU = dst-AU | src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = dst-AU | src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = dst-DU | src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[2]	dst-AU = src-AU | k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU | k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src | k8	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[3]	dst-AU = src-AU | k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU | k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src | k16	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[4]	dst-AU = src-AU | Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = src-DU | Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = src | Smem	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[5]	ACy = ACy | (ACx <<< #SHIFTW)	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[6]	ACy = ACx | (k16 <<< #16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[7]	ACy = ACx | (k16 <<< #SHFT)	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[8]	Smem = Smem | k16	N	4	1	X	AU_ALU	1	.	.	1	.	1	.	

Bitwise Exclusive OR (XOR) (page 5-57)

No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[1]	dst-AU = dst-AU ^ src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = dst-AU ^ src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = dst-DU ^ src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[2]	dst-AU = src-AU ^ k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU ^ k8	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src ^ k8	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[3]	dst-AU = src-AU ^ k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU ^ k16	N	4	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src ^ k16	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[4]	dst-AU = src-AU ^ Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = src-DU ^ Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = src ^ Smem	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.
[5]	ACy = ACy ^ (ACx <<< #SHIFTW)	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[6]	ACy = ACx ^ (k16 <<< #16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[7]	ACy = ACx ^ (k16 <<< #SHFT)	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[8]	Smem = Smem ^ k16	N	4	1	X	AU_ALU	1	.	.	1	.	1	.	

Branch Conditionally (page 5-66)

No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[1]	if (cond) goto l4	N	2	6/5†	R	PU_UNIT	.	.	.	.	.	.	.	
[2]	if (cond) goto L8	Y	3	6/5†	R	PU_UNIT	.	.	.	.	.	.	.	
[3]	if (cond) goto L16	N	4	6/5†	R	PU_UNIT	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[4]	if (cond) goto P24	N	5	5/5†	R	PU_UNIT	.	.	.	.	.	.	.	
† x/y cycles: x cycles = condition true, y cycles = condition false														
Branch Unconditionally (page 5-70)														
[1]	goto ACx	N	2	10	X	PU_UNIT	.	.	.	.	.	.	1	
[2]	goto L7	Y	2	6†	AD	PU_UNIT	.	.	.	.	.	.	.	
[3]	goto L16	Y	3	6†	AD	PU_UNIT	.	.	.	.	.	.	.	
[4]	goto P24	N	4	5	D	PU_UNIT	.	.	.	.	.	.	.	
† These instructions execute in 3 cycles if the addressed instruction is in the instruction buffer unit.														
Branch on Auxiliary Register Not Zero (page 5-74)														
	if (ARn_mod != #0) goto L16	N	4	6/5†	AD	PU_UNIT	1	.	.	.	.	.	.	
† x/y cycles: x cycles = condition true, y cycles = condition false														
Call Conditionally (page 5-77)														
[1]	if (cond) call L16	N	4	6/5†	R	PU_UNIT	1	.	1	.	.	2	.	
[2]	if (cond) call P24	N	5	5/5†	R	PU_UNIT	1	.	1	.	.	2	.	
† x/y cycles: x cycles = condition true, y cycles = condition false														
Call Unconditionally (page 5-83)														
[1]	call ACx	N	2	10	X	PU_UNIT	1	.	1	.	.	2	1	
[2]	call L16	Y	3	6	AD	PU_UNIT	1	.	1	.	.	2	.	
[3]	call P24	N	4	5	D	PU_UNIT	1	.	1	.	.	2	.	
Circular Addressing Qualifier (page 5-87)														
	circular()	N	1	1	AD		.	.	.	.	.	.	.	
Clear Accumulator, Auxiliary, or Temporary Register Bit (page 5-88)														
	bit(src-AU, Baddr) = #0	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	bit(src-DU, Baddr) = #0	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU
2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Clear Memory Bit (page 5-89)														
	bit(Smem, src) = #0	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
Clear Status Register Bit (page 5-90)														
[1]	bit(ST0, k4) = #0	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[2]	bit(ST1, k4) = #0	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[3]	bit(ST2, k4) = #0	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[4]	bit(ST3, k4) = #0	Y	2	1†	X	AU_ALU	.	.	.	.	.	.	.	
† When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.														
Compare Accumulator, Auxiliary, or Temporary Register Content (page 5-93)														
[1]	TC1 = uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	.
	TC1 = uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TC1 = uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[2]	TC2 = uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	.
	TC2 = uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TC2 = uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
Compare Accumulator, Auxiliary, or Temporary Register Content with AND (page 5-95)														
[1]	TCx = TCy & uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	TCx = TCy & uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TCx = TCy & uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[2]	TCx = !TCy & uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	TCx = !TCy & uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TCx = !TCy & uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	
Compare Accumulator, Auxiliary, or Temporary Register Content with OR (page 5-100)														
[1]	TCx = TCy | uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	TCx = TCy | uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TCx = TCy | uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[2]	TCx = !TCy | uns(src-AU RELOP dst-AU)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	TCx = !TCy | uns(src RELOP dst)	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	See Note 2.
	TCx = !TCy | uns(src-DU RELOP dst-DU)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
Compare Accumulator, Auxiliary, or Temporary Register Content Maximum (page 5-105)														
	dst-AU = max(src-AU, dst-AU)	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = max(src-DU, dst-AU)	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = max(src, dst-DU)	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
Compare Accumulator, Auxiliary, or Temporary Register Content Minimum (page 5-108)														
	dst-AU = min(src-AU, dst-AU)	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = min(src-DU, dst-AU)	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = min(src, dst-DU)	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
Compare and Branch (page 5-111)														
	compare (uns(src-AU RELOP K8)) goto L8	N	4	7/6†	X	AU_ALU + PU_UNIT	.	.	.	.	.	.	.	
	compare (uns(src-DU RELOP K8)) goto L8	N	4	7/6†	X	DU_ALU + PU_UNIT	.	.	.	.	.	.	.	
† x/y cycles: x cycles = condition true, y cycles = condition false														
Compare and Select Accumulator Content Maximum (page 5-114)														
[1]	max_diff(ACx, ACy, ACz, ACw)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Compare and Select Accumulator Content Minimum (page 5-120)														
[1]	min_diff(ACx, ACy, ACz, ACw)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[2]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
Compare Memory with Immediate Value (page 5-126)														
[1]	TC1 = (Smem == K16)	N	4	1	X	AU_ALU	1	.	.	1	.	.	.	
[2]	TC2 = (Smem == K16)	N	4	1	X	AU_ALU	1	.	.	1	.	.	.	
Complement Accumulator, Auxiliary, or Temporary Register Bit (page 5-128)														
	cbit(src-AU, Baddr)	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	cbit(src-DU, Baddr)	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	
Complement Accumulator, Auxiliary, or Temporary Register Content (page 5-129)														
	dst-AU = ~src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = ~src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = ~src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
Complement Memory Bit (page 5-130)														
	cbit(Smem, src)	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
Compute Exponent of Accumulator Content (page 5-131)														
	Tx = exp(ACx)	Y	3	1	X	DU_BIT + AU_ALU	.	.	.	.	.	.	1	
Compute Mantissa and Exponent of Accumulator Content (page 5-132)														
	ACy = mant(ACx), Tx = exp(ACx)	Y	3	1	X	DU_BIT + DU_SHIFT + AU_ALU	.	.	.	.	.	.	1	
Count Accumulator Bits (page 5-134)														
[1]	Tx = count(ACx, ACy, TC1)	Y	3	1	X	DU_BIT + AU_ALU	.	.	.	.	.	.	1	
[2]	Tx = count(ACx, ACy, TC2)	Y	3	1	X	DU_BIT + AU_ALU	.	.	.	.	.	.	1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Dual 16-Bit Additions (page 5-135)														
[1]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
Dual 16-Bit Addition and Subtraction (page 5-140)														
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem − Tx	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) − Tx	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
Dual 16-Bit Subtractions (page 5-145)														
[1]	HI(ACy) = HI(ACx) − HI(Lmem), LO(ACy) = LO(ACx) − LO(Lmem)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[2]	HI(ACy) = HI(Lmem) − HI(ACx), LO(ACy) = LO(Lmem) − LO(ACx)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[3]	HI(ACx) = Tx − HI(Lmem), LO(ACx) = Tx − LO(Lmem)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[4]	HI(ACx) = HI(Lmem) − Tx, LO(ACx) = LO(Lmem) − Tx	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
Dual 16-Bit Subtraction and Addition (page 5-154)														
[1]	HI(ACx) = Smem − Tx, LO(ACx) = Smem + Tx	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[2]	HI(ACx) = HI(Lmem) − Tx, LO(ACx) = LO(Lmem) + Tx	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
Execute Conditionally (page 5-159)														
[1]	if (cond) execute(AD_Unit)	N	2	1	AD	PU_UNIT	.	.	.	.	.	.	.	
[2]	if (cond) execute(D_Unit)	N	2	1	X	PU_UNIT	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Expand Accumulator Bit Field (page 5-166)														
	dst-AU = field_expand(ACx, k16)	N	4	1	X	DU_BIT + AU_ALU	.	.	.	.	.	.	1	
	dst-DU = field_expand(ACx, k16)	N	4	1	X	DU_BIT	.	.	.	.	.	.	.	
Extract Accumulator Bit Field (page 5-167)														
	dst-AU = field_extract(ACx, k16)	N	4	1	X	DU_BIT + AU_ALU	.	.	.	.	.	.	1	
	dst-DU = field_extract(ACx, k16)	N	4	1	X	DU_BIT	.	.	.	.	.	.	.	
Finite Impulse Response Filter, Antisymmetrical (page 5-168)														
	firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	1	.	2	1	.	.	
Finite Impulse Response Filter, Symmetrical (page 5-170)														
	firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	1	.	2	1	.	.	
Idle (page 5-172)														
	idle	N	4	?	D	PU_UNIT	.	.	.	.	.	.	.	
Least Mean Square (LMS) (page 5-173)														
	lms(Xmem, Ymem, ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	.	.	2	.	.	.	
Least Mean Square (LMSF) (page 5-175)														
	lmsf(Xmem, Ymem, ACx, ACy)	N	4	1	X	DU_MAC1 + DU_MAC2 + DU_ALU	2	.	.	2	.	1	.	
Linear Addressing Qualifier (page 5-179)														
	linear()	N	1	1	AD		.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Load Accumulator from Memory (page 5-180)														
[1]	ACx = rnd(Smem << Tx)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[2]	ACx = low_byte(Smem) << #SHIFTW	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[3]	ACx = high_byte(Smem) << #SHIFTW	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[4]	ACx = Smem << #16	N	2	1	X	DU_LOAD	1	.	.	1	.	.	.	
[5]	ACx = uns(Smem)	N	3	1	X	DU_LOAD	1	.	.	1	.	.	.	
[6]	ACx = uns(Smem) << #SHIFTW	N	4	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[7]	ACx = M40(dbl(Lmem))	N	3	1	X	DU_LOAD	1	.	.	2	.	.	.	
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	N	3	1	X	DU_LOAD	2	.	.	2	.	.	.	
Load Accumulator Pair from Memory (page 5-191)														
[1]	pair(HI(ACx)) = Lmem	N	3	1	X	DU_LOAD	1	.	.	2	.	.	.	
[2]	pair(LO(ACx)) = Lmem	N	3	1	X	DU_LOAD	1	.	.	2	.	.	.	
Load Accumulator with Immediate Value (page 5-196)														
[1]	ACx = K16 << #16	N	4	1	X	DU_LOAD	.	.	.	.	.	.	.	
[2]	ACx = K16 << #SHFT	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
Load Accumulator from Memory with Parallel Store Accumulator Content to Memory (page 5-189)														
	ACy = Xmem << #16, Ymem = HI(ACx << T2)	N	4	1	X	DU_LOAD + DU_SHIFT	2	.	.	2	.	2	.	
Load Accumulator, Auxiliary, or Temporary Register from Memory (page 5-199)														
[1]	dst-AU = Smem	N	2	1	X	AU_LOAD	1	.	.	1	.	.	.	
	dst-DU = Smem	N	2	1	X	DU_LOAD	1	.	.	1	.	.	.	
[2]	dst-AU = uns(high_byte(Smem))	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
	dst-DU = uns(high_byte(Smem))	N	3	1	X	DU_LOAD	1	.	.	1	.	.	.	
[3]	dst-AU = uns(low_byte(Smem))	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
	dst-DU = uns(low_byte(Smem))	N	3	1	X	DU_LOAD	1	.	.	1	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Load Accumulator, Auxiliary, or Temporary Register with Immediate Value (page 5-205)														
[1]	dst-AU = k4	Y	2	1	X	AU_LOAD	.	.	.	.	.	.	.	
	dst-DU = k4	Y	2	1	X	DU_LOAD	.	.	.	.	.	.	.	
[2]	dst-AU = −k4	Y	2	1	X	AU_LOAD	.	.	.	.	.	.	.	
	dst-DU = −k4	Y	2	1	X	DU_LOAD	.	.	.	.	.	.	.	
[3]	dst-AU = K16	N	4	1	X	AU_LOAD	.	.	.	.	.	.	.	
	dst-DU = K16	N	4	1	X	DU_LOAD	.	.	.	.	.	.	.	
Load Auxiliary or Temporary Register Pair from Memory (page 5-209)														
	pair(TAx) = Lmem	N	3	1	X	AU_LOAD	1	.	.	2	.	.	.	
Load CPU Register from Memory (page 5-210)														
[1]	BK03 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[2]	BK47 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[3]	BKC = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[4]	BSA01 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[5]	BSA23 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[6]	BSA45 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[7]	BSA67 = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[8]	BSAC = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[9]	BRC0 = Smem	N	3	1	X		1	.	.	1	.	.	.	
[10]	BRC1 = Smem	N	3	1	X		1	.	.	1	.	.	.	
[11]	CDP = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[12]	CSR = Smem	N	3	1	X		1	.	.	1	.	.	.	
[13]	DP = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[14]	DPH = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[15]	PDP = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[16]	SP = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[17]	SSP = Smem	N	3	1	X	AU_LOAD	1	.	.	1	.	.	.	
[18]	TRN0 = Smem	N	3	1	X	DU_LOAD	1	.	.	1	.	.	.	
[19]	TRN1 = Smem	N	3	1	X	DU_LOAD	1	.	.	1	.	.	.	
[20]	RETA = dbl(Lmem)	N	3	5	X		1	.	.	2	.	.	.	

Load CPU Register with Immediate Value (page 5-213)

No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[1]	BK03 = k12	Y	3	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[2]	BK47 = k12	Y	3	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[3]	BKC = k12	Y	3	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[4]	BRC0 = k12	Y	3	1	AD		.	.	.	.	.	.	.	
[5]	BRC1 = k12	Y	3	1	AD		.	.	.	.	.	.	.	
[6]	CSR = k12	Y	3	1	AD		.	.	.	.	.	.	.	
[7]	DPH = k7	Y	3	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[8]	PDP = k9	Y	3	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[9]	BSA01 = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[10]	BSA23 = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[11]	BSA45 = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[12]	BSA67 = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[13]	BSAC = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[14]	CDP = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[15]	DP = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[16]	SP = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	
[17]	SSP = k16	N	4	1	AD	AU_LOAD	.	.	.	.	.	.	.	

Load Extended Auxiliary Register from Memory (page 5-215)

No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
	XAdst = dbl(Lmem)	N	3	1	X		1	.	.	2	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
	Load Extended Auxiliary Register with Immediate Value (page 5-216)													
	XAdst = k23	N	6	1	AD	AU_LOAD	1	.	.	1	.	.	.	
	Load Memory with Immediate Value (page 5-217)													
[1]	Smem = K8	N	3	1	X		1	.	.	.	.	1	.	
[2]	Smem = K16	N	4	1	X		1	.	.	.	.	1	.	
	Lock Access Qualifier (page 5-218)													
	lock()	N	2	1	D		.	.	.	.	.	.	.	
	Memory Delay (page 5-220)													
	delay(Smem)	N	2	1	X		2	1	.	1	1	1	.	
	Memory-Mapped Register Access Qualifier (page 5-221)													
	mmap()	N	1	1	D		.	.	.	.	.	.	.	
	Modify Auxiliary Register Content (page 5-222)													
	mar(Smem)	N	2	1	AD		1	.	.	1	.	.	.	
	Modify Auxiliary Register Content with Parallel Multiply (page 5-224)													
	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	X	DU_MAC1	2	1	.	2	1	.	.	
	Modify Auxiliary Register Content with Parallel Multiply and Accumulate (page 5-226)													
[1]	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1	2	1	.	2	1	.	.	
[2]	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1	2	1	.	2	1	.	.	
	Modify Auxiliary Register Content with Parallel Multiply and Subtract (page 5-231)													
	mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1	2	1	.	2	1	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Modify Auxiliary or Temporary Register Content (page 5-233)														
[1]	mar(TAy = TAx)	N	3	1	AD		1	.	.	.	.	.	.	
[2]	mar(TAx = P8)	N	3	1	AD		1	.	.	.	.	.	.	
[3]	mar(TAx = D16)	N	4	1	AD		1	.	.	.	.	.	.	
Modify Auxiliary or Temporary Register Content by Addition (page 5-237)														
[1]	mar(TAy + TAx)	N	3	1	AD		1	.	.	.	.	.	.	
[2]	mar(TAx + P8)	N	3	1	AD		1	.	.	.	.	.	.	
Modify Auxiliary or Temporary Register Content by Subtraction (page 5-241)														
[1]	mar(TAy − TAx)	N	3	1	AD		1	.	.	.	.	.	.	
[2]	mar(TAx − P8)	N	3	1	AD		1	.	.	.	.	.	.	
Modify Data Stack Pointer (SP) (page 5-245)														
	SP = SP + K8	Y	2	1	AD		.	.	.	.	.	.	.	
Modify Extended Auxiliary Register Content (page 5-246)														
[1]	XAdst = mar(Smem)	N	3	1	AD		1	.	.	1	.	.	.	
[2]	mar(XACdst = XACsrc)	Y	3	1	AD		1	.	.	.	.	.	.	
Modify Extended Auxiliary Register Content by Addition (page 5-249)														
	mar(XACdst + XACsrc)	Y	3	1	AD		1	.	.	.	.	.	.	
Modify Extended Auxiliary Register Content by Subtraction (page 5-251)														
	mar(XACdst − XACsrc)	Y	3	1	AD		1	.	.	.	.	.	.	
Move Accumulator Content to Auxiliary or Temporary Register (page 5-253)														
	TAx = HI(ACx)	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Move Accumulator, Auxiliary, or Temporary Register Content (page 5-254)														
	dst-AU = src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
Move Auxiliary or Temporary Register Content to Accumulator (page 5-256)														
	HI(ACx) = TAx	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	
Move Auxiliary or Temporary Register Content to CPU Register (page 5-257)														
[1]	BRC0 = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[2]	BRC1 = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[3]	CDP = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[4]	CSR = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[5]	SP = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[6]	SSP = TAx	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
Move CPU Register Content to Auxiliary or Temporary Register (page 5-259)														
[1]	TAx = BRC0	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[2]	TAx = BRC1	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[3]	TAx = CDP	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[4]	TAx = RPTC	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[5]	TAx = SP	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[6]	TAx = SSP	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
Move Extended Auxiliary Register Content (page 5-261)														
	xdst-AU = xsrc-AU	N	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	xdst-AU = xsrc-DU	N	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	xdst-DU = xsrc	N	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	

Move Memory to Memory (page 5-262)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	Smem = coef(Cmem)	N	3	1	X		2	.	.	1	.	1	.	
[2]	coef(Cmem) = Smem	N	3	1	X		2	.	.	1	.	1	.	
[3]	Lmem = dbl(coef(Cmem))	N	3	1	X		2	.	.	2	.	2	.	
[4]	dbl(coef(Cmem)) = Lmem	N	3	1	X		2	.	.	2	.	2	.	
[5]	dbl(Ymem) = dbl(Xmem)	N	3	1	X		2	.	.	2	.	2	.	
[6]	Ymem = Xmem	N	3	1	X		2	.	.	2	.	2	.	

Multiply (MPY) (page 5-269)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	ACy = rnd(ACy * ACx)	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACy = rnd(ACx * Tx)	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[3]	ACy = rnd(ACx * K8)	Y	3	1	X	DU_ALU	.	.	.	.	.	.	.	
[4]	ACy = rnd(ACx * K16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	N	3	1	X	DU_ALU	1	1	.	1	1	.	.	
[6]	ACy = rnd(Smem * ACx)[, T3 = Smem]	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[7]	ACx = rnd(Smem * K8)[, T3 = Smem]	N	4	1	X	DU_MAC1	1	.	.	1	.	.	.	
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	
[9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
[10]	ACx = rnd(Smem * uns(coef(Cmem)))	N	3	1	X	DU_MAC1	1	1	.	1	1	.	.	

Multiply with Parallel Multiply and Accumulate (page 5-283)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	
Multiply with Parallel Multiply and Subtract (page 5-295)														
[1]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[2]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[3]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
Multiply with Parallel Store Accumulator Content to Memory (page 5-305)														
	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	X	DU_MAC1 + DU_SHIFT	2	.	.	2	.	2	.	
Multiply and Accumulate (MAC) (page 5-308)														
[1]	ACy = rnd(ACy + (ACx * Tx))	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACy = rnd((ACy * Tx) + ACx)	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[3]	ACy = rnd(ACx + (Tx * K8))	Y	3	1	X	DU_MAC1	.	.	.	.	.	.	.	
[4]	ACy = rnd(ACx + (Tx * K16))	N	4	1	X	DU_MAC1	.	.	.	.	.	.	.	
[5]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	1	.	1	1	.	.	
[6]	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
[7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
[8]	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem]	N	4	1	X	DU_MAC1	1	.	.	1	.	.	.	
[9]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	
[10]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	
[11]	ACx = rnd(ACx + (Smem * uns(coef(Cmem))))	N	3	1	X	DU_MAC1	1	1	.	1	1	.	.	
Multiply and Accumulate with Parallel Delay (page 5-325)														
	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	N	3	1	X	DU_MAC1	2	1	.	1	1	1	.	
Multiply and Accumulate with Parallel Load Accumulator from Memory (page 5-327)														
	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	
	Multiply and Accumulate with Parallel Multiply (page 5-329)													
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[4]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[5]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
	Multiply and Accumulate with Parallel Multiply and Subtract (page 5-347)													
[1]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[2]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[4]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
	Multiply and Accumulate with Parallel Store Accumulator Content to Memory (page 5-367)													
	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	X	DU_MAC1 + DU_SHIFT	2	.	.	2	.	2	.	
	Multiply and Subtract (MAS) (page 5-369)													
[1]	ACy = rnd(ACy – (ACx * Tx))	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACx = rnd(ACx – (Smem * coef(Cmem)))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	1	.	1	1	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[3]	ACy = rnd(ACy – (Smem * ACx))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
[4]	ACy = rnd(ACx – (Tx * Smem))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
[5]	ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	
[6]	ACx = rnd(ACx − (Smem * uns(coef(Cmem))))	N	3	1	X	DU_MAC1	1	1	.	1	1	.	.	
	Multiply and Subtract with Parallel Load Accumulator from Memory (page 5-379)													
	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	X	DU_MAC1	2	.	.	2	.	.	.	
	Multiply and Subtract with Parallel Multiply (page 5-381)													
[1]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
	Multiply and Subtract with Parallel Multiply and Accumulate (page 5-390)													
[1]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[3]	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[4]	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
	Multiply and Subtract with Parallel Store Accumulator Content to Memory (page 5-401)													
	ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	X	DU_MAC1 + DU_SHIFT	2	.	.	2	.	2	.	
	Negate Accumulator, Auxiliary, or Temporary Register Content (page 5-403)													
	dst-AU = −src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = −src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	dst-DU = −src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
	No Operation (NOP) (page 5-405)													
[1]	nop	Y	1	1	D		.	.	.	.	.	.	.	
[2]	nop_16	Y	2	1	D		.	.	.	.	.	.	.	
	Parallel Modify Auxiliary Register Contents (page 5-406)													
	mar(Xmem) , mar(Ymem) , mar(coef(Cmem))	N	4	1	X		2	1	.	2	1	.	.	
	Parallel Multiplies (page 5-407)													
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
	Parallel Multiply and Accumulates (page 5-419)													
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_ALU	2	1	.	2	1	.	.	
[3]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[4]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[5]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[7]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[8]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[9]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[10]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem))))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
[11]	ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem)))))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
[12]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem)))))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
Parallel Multiply and Subtracts (page 5-454)														
[1]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(coef(Cmem))))))	N	4	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	1	.	.	
[2]	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem)))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	1	2	.	.	
[3]	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem)))))))	N	4	1	X	DU_MAC1 + DU_MAC2	1	1	.	2	2	.	.	
[4]	ACy = M40(rnd(ACy − (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem)))))))	N	5	1	X	DU_MAC1 + DU_MAC2	2	1	.	2	2	.	.	
Peripheral Port Register Access Qualifiers (page 5-466)														
[1]	readport()	N	1	1	D		.	.	.	.	.	.	.	
[2]	writeport()	N	1	1	D		.	.	.	.	.	.	.	
Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers (page 5-468)														
	xdst-AU = popboth()	Y	2	1	X	AU_LOAD	1	.	1	2	.	.	.	
	xdst-DU = popboth()	Y	2	1	X	DU_LOAD	1	.	1	2	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
Pop Top of Stack (page 5-469)														
[1]	dst1-AU, dst2-AU = pop()	Y	2	1	X	AU_LOAD	1	.	1	2	.	.	.	
	dst1-DU, dst2-DU = pop()	Y	2	1	X	DU_LOAD	1	.	1	2	.	.	.	
	dst1-AU, dst2-DU = pop()	Y	2	1	X	AU_LOAD + DU_LOAD	1	.	1	2	.	.	.	
	dst1-DU, dst2-AU = pop()	Y	2	1	X	DU_LOAD + AU_LOAD	1	.	1	2	.	.	.	
[2]	dst-AU = pop()	Y	2	1	X	AU_LOAD	1	.	1	1	.	.	.	
	dst-DU = pop()	Y	2	1	X	DU_LOAD	1	.	1	1	.	.	.	
[3]	dst-AU, Smem = pop()	N	3	1	X	AU_LOAD	1	.	1	2	.	1	.	
	dst-DU, Smem = pop()	N	3	1	X	DU_LOAD	1	.	1	2	.	1	.	
[4]	ACx = dbl(pop())	Y	2	1	X	DU_LOAD	1	.	1	2	.	.	.	
[5]	Smem = pop()	N	2	1	X		1	.	1	1	.	1	.	
[6]	dbl(Lmem) = pop()	N	2	1	X		1	.	1	2	.	2	.	
Push Accumulator or Extended Auxiliary Register Content to Stack Pointers (page 5-476)														
	pushboth(xsrc)	Y	2	1	X		1	.	1	.	.	2	.	
Push to Top of Stack (page 5-477)														
[1]	push(src1, src2)	Y	2	1	X		1	.	1	.	.	2	.	
[2]	push(src)	Y	2	1	X		1	.	1	.	.	1	.	
[3]	push(src, Smem)	N	3	1	X		1	.	1	1	.	2	.	
[4]	dbl(push(ACx))	Y	2	1	X		1	.	1	.	.	2	.	
[5]	push(Smem)	N	2	1	X		1	.	1	1	.	1	.	
[6]	push(dbl(Lmem))	N	2	1	X		1	.	1	2	.	2	.	
Repeat Block of Instructions Unconditionally (page 5-484)														
[1]	localrepeat{ }	Y	2	1	AD	PU_UNIT	.	.	.	.	.	.	.	
[2]	blockrepeat{ }	Y	3	1	AD	PU_UNIT	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
	Repeat Single Instruction Conditionally (page 5-495)													
	while (cond && (RPTC < k8)) repeat	Y	3	1	AD	PU_UNIT	.	.	.	.	.	.	.	
	Repeat Single Instruction Unconditionally (page 5-498)													
[1]	repeat(k8)	Y	2	1	AD	PU_UNIT	.	.	.	.	.	.	.	
[2]	repeat(k16)	Y	3	1	AD	PU_UNIT	.	.	.	.	.	.	.	
[3]	repeat(CSR)	Y	2	1	AD	PU_UNIT	.	.	.	.	.	.	.	
	Repeat Single Instruction Unconditionally and Decrement CSR (page 5-503)													
	repeat(CSR), CSR −= k4	Y	2	1	X	AU_ALU + PU_UNIT	.	.	.	.	.	.	.	
	Repeat Single Instruction Unconditionally and Increment CSR (page 5-505)													
[1]	repeat(CSR), CSR += TAx	Y	2	1	X	AU_ALU + PU_UNIT	.	.	.	.	.	.	.	
[2]	repeat(CSR), CSR += k4	Y	2	1	X	AU_ALU + PU_UNIT	.	.	.	.	.	.	.	
	Return Conditionally (page 5-508)													
	if (cond) return	Y	3	5/5†	R	PU_UNIT	1	.	1	2	.	.	.	
	† x/y cycles: x cycles = condition true, y cycles = condition false													
	Return Unconditionally (page 5-510)													
	return	Y	2	5	D	PU_UNIT	1	.	1	2	.	.	.	
	Return from Interrupt (page 5-512)													
	return_int	N	2	5	D	PU_UNIT	1	.	1	2	.	.	.	
	Rotate Left Accumulator, Auxiliary, or Temporary Register Content (page 5-514)													
	dst-AU = BitOut \\ src-AU \\ BitIn	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = BitOut \\ src-DU \\ BitIn	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = BitOut \\ src-AU \\ BitIn	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU
2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	
Rotate Right Accumulator, Auxiliary, or Temporary Register Content (page 5-516)														
	dst-AU = BitIn // src-AU // BitOut	Y	3	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = BitIn // src-DU // BitOut	Y	3	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = BitIn // src-AU // BitOut	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	See Note 1.
Round Accumulator Content (page 5-518)														
	ACy = rnd(ACx)	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	
Saturate Accumulator Content (page 5-520)														
	ACy = saturate(rnd(ACx))	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	
Set Accumulator, Auxiliary, or Temporary Register Bit (page 5-522)														
	bit(src-AU, Baddr) = #1	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	bit(src-DU, Baddr) = #1	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	
Set Memory Bit (page 5-523)														
	bit(Smem, src) = #1	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
Set Status Register Bit (page 5-524)														
[1]	bit(ST0, k4) = #1	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[2]	bit(ST1, k4) = #1	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[3]	bit(ST2, k4) = #1	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
[4]	bit(ST3, k4) = #1	Y	2	1[†]	X	AU_ALU	.	.	.	.	.	.	.	
Shift Accumulator Content Conditionally (page 5-527)														
[1]	ACx = sftc(ACx, TC1)	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[2]	ACx = sftc(ACx, TC2)	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
Shift Accumulator Content Logically (page 5-529)														
[1]	ACy = ACx <<< Tx	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	

[†] When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[2]	ACy = ACx <<< #SHIFTW	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	

Shift Accumulator, Auxiliary, or Temporary Register Content Logically (page 5-532)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	dst-AU = dst-AU <<< #1	Y	2	1	X	AU_ALU + DU_SHIFT	.	.	.	.	.	.	.	
	dst-DU = dst-DU <<< #1	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[2]	dst-AU = dst-AU >>> #1	Y	2	1	X	AU_ALU + DU_SHIFT	.	.	.	.	.	.	.	
	dst-DU = dst-DU >>> #1	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	

Signed Shift of Accumulator Content (page 5-535)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	ACy = ACx << Tx	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[2]	ACy = ACx << #SHIFTW	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[3]	ACy = ACx <<C Tx	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[4]	ACy = ACx <<C #SHIFTW	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	

Signed Shift of Accumulator, Auxiliary, or Temporary Register Content (page 5-544)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	dst-AU = dst-AU >> #1	Y	2	1	X	AU_ALU + DU_SHIFT	.	.	.	.	.	.	.	
	dst-DU = dst-DU >> #1	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[2]	dst-AU = dst-AU << #1	Y	2	1	X	AU_ALU + DU_SHIFT	.	.	.	.	.	.	.	
	dst-DU = dst-DU << #1	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	

Software Interrupt (page 5-549)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	intr(k5)	N	2	3	D	PU_UNIT	1	.	1	.	.	2	.	

Software Reset (page 5-551)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	reset	N	2	?	D	PU_UNIT	.	.	.	.	.	.	.	

Software Trap (page 5-555)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	trap(k5)	N	2	?	D	PU_UNIT	1	.	1	.	.	2	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

No.	Instruction	E	S	C	Pipe	Operator	Address Generation Unit			Buses				Notes
							DA	CA	SA	DR	CR	DW	ACB	
Square (page 5-557)														
[1]	ACy = rnd(ACx * ACx)	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACx = rnd(Smem * Smem)[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
Square and Accumulate (page 5-560)														
[1]	ACy = rnd(ACy + (ACx * ACx))	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
Square and Subtract (page 5-563)														
[1]	ACy = rnd(ACy – (ACx * ACx))	Y	2	1	X	DU_MAC1	.	.	.	.	.	.	.	
[2]	ACy = rnd(ACx – (Smem * Smem))[, T3 = Smem]	N	3	1	X	DU_MAC1	1	.	.	1	.	.	.	
Square Distance (page 5-566)														
	sqdst(Xmem, Ymem, ACx, ACy)	N	4	1	X	DU_ALU + DU_MAC1	2	.	.	2	.	.	.	
Store Accumulator Content to Memory (page 5-568)														
[1]	Smem = HI(ACx)	N	2	1	X		1	.	.	.	.	1	.	
[2]	Smem = HI(rnd(ACx))	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[3]	Smem = LO(ACx << Tx)	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[4]	Smem = HI(rnd(ACx << Tx))	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[5]	Smem = LO(ACx << #SHIFTW)	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[6]	Smem = HI(ACx << #SHIFTW)	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[7]	Smem = HI(rnd(ACx << #SHIFTW))	N	4	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[8]	Smem = HI(saturate(uns(rnd(ACx))))	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[9]	Smem = HI(saturate(uns(rnd(ACx << Tx))))	N	3	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[10]	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	N	4	1	X	DU_SHIFT	1	.	.	.	.	1	.	
[11]	dbl(Lmem) = ACx	N	3	1	X		1	.	.	.	.	2	.	
[12]	dbl(Lmem) = saturate(uns(ACx))	N	3	1	X	DU_SHIFT	1	.	.	.	.	2	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[13]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	N	3	1	X	DU_SHIFT	1	.	.	.	.	2	.	
[14]	Xmem = LO(ACx), Ymem = HI(ACx)	N	3	1	X		2	.	.	.	.	2	.	
Store Accumulator Pair Content to Memory (page 5-588)														
[1]	Lmem = pair(HI(ACx))	N	3	1	X		1	.	.	.	.	2	.	
[2]	Lmem = pair(LO(ACx))	N	3	1	X		1	.	.	.	.	2	.	
Store Accumulator, Auxiliary, or Temporary Register Content to Memory (page 5-591)														
[1]	Smem = src	N	2	1	X		1	.	.	.	.	1	.	
[2]	high_byte(Smem) = src	N	3	1	X		1	.	.	.	.	1	.	
[3]	low_byte(Smem) = src	N	3	1	X		1	.	.	.	.	1	.	
Store Auxiliary or Temporary Register Pair Content to Memory (page 5-595)														
	Lmem = pair(TAx)	N	3	1	X		1	.	.	.	.	2	.	
Store CPU Register Content to Memory (page 5-596)														
[1]	Smem = BK03	N	3	1	X		1	.	.	.	.	1	.	
[2]	Smem = BK47	N	3	1	X		1	.	.	.	.	1	.	
[3]	Smem = BKC	N	3	1	X		1	.	.	.	.	1	.	
[4]	Smem = BSA01	N	3	1	X		1	.	.	.	.	1	.	
[5]	Smem = BSA23	N	3	1	X		1	.	.	.	.	1	.	
[6]	Smem = BSA45	N	3	1	X		1	.	.	.	.	1	.	
[7]	Smem = BSA67	N	3	1	X		1	.	.	.	.	1	.	
[8]	Smem = BSAC	N	3	1	X		1	.	.	.	.	1	.	
[9]	Smem = BRC0	N	3	1	X		1	.	.	.	.	1	.	
[10]	Smem = BRC1	N	3	1	X		1	.	.	.	.	1	.	
[11]	Smem = CDP	N	3	1	X		1	.	.	.	.	1	.	
[12]	Smem = CSR	N	3	1	X		1	.	.	.	.	1	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU
2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[13]	Smem = DP	N	3	1	X		1	.	.	.	.	1	.	
[14]	Smem = DPH	N	3	1	X		1	.	.	.	.	1	.	
[15]	Smem = PDP	N	3	1	X		1	.	.	.	.	1	.	
[16]	Smem = SP	N	3	1	X		1	.	.	.	.	1	.	
[17]	Smem = SSP	N	3	1	X		1	.	.	.	.	1	.	
[18]	Smem = TRN0	N	3	1	X		1	.	.	.	.	1	.	
[19]	Smem = TRN1	N	3	1	X		1	.	.	.	.	1	.	
[20]	dbl(Lmem) = RETA	N	3	5	X		1	.	.	.	.	2	.	

Store Extended Auxiliary Register Content to Memory (page 5-600)

	dbl(Lmem) = XAsrc	N	3	1	X		1	.	.	.	.	2	.	

Subtract Conditionally (page 5-601)

	subc(Smem, ACx, ACy)	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	

Subtraction (page 5-603)

[1]	dst-AU = dst-AU − src-AU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = dst-AU − src-DU	Y	2	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = dst-DU − src	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[2]	dst-AU = dst-AU − k4	Y	2	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-DU = dst-DU − k4	Y	2	1	X	DU_ALU	.	.	.	.	.	.	.	
[3]	dst-AU = src-AU − K16	N	4	1	X	AU_ALU	.	.	.	.	.	.	.	
	dst-AU = src-DU − K16	N	4	1	X	AU_ALU	.	.	.	.	.	.	1	
	dst-DU = src − K16	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	See Note 1.
[4]	dst-AU = src-AU − Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = src-DU − Smem	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = src − Smem	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[5]	dst-AU = Smem − src-AU	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
	dst-AU = Smem − src-DU	N	3	1	X	AU_ALU	1	.	.	1	.	.	1	
	dst-DU = Smem − src	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	See Note 1.
[6]	ACy = ACy − (ACx << Tx)	Y	2	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[7]	ACy = ACy − (ACx << #SHIFTW)	Y	3	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[8]	ACy = ACx − (K16 << #16)	N	4	1	X	DU_ALU	.	.	.	.	.	.	.	
[9]	ACy = ACx − (K16 << #SHFT)	N	4	1	X	DU_SHIFT	.	.	.	.	.	.	.	
[10]	ACy = ACx − (Smem << Tx)	N	3	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[11]	ACy = ACx − (Smem << #16)	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[12]	ACy = (Smem << #16) − ACx	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[13]	ACy = ACx – uns(Smem) – BORROW	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[14]	ACy = ACx – uns(Smem)	N	3	1	X	DU_ALU	1	.	.	1	.	.	.	
[15]	ACy = ACx – (uns(Smem) << #SHIFTW)	N	4	1	X	DU_SHIFT	1	.	.	1	.	.	.	
[16]	ACy = ACx − dbl(Lmem)	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[17]	ACy = dbl(Lmem) − ACx	N	3	1	X	DU_ALU	1	.	.	2	.	.	.	
[18]	ACx = (Xmem << #16) − (Ymem << #16)	N	3	1	X	DU_ALU	2	.	.	2	.	.	.	

Subtraction with Parallel Store Accumulator Content to Memory (page 5-627)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	ACy = (Xmem << #16) − ACx, Ymem = HI(ACy << T2)	N	4	1	X	DU_ALU + DU_SHIFT	2	.	.	2	.	2	.	

Swap Accumulator Content (page 5-629)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	swap(AC0, AC2)	Y	2	1	X	DU_SWAP	.	.	.	.	.	.	.	
[2]	swap(AC1, AC3)	Y	2	1	X	DU_SWAP	.	.	.	.	.	.	.	

Swap Accumulator Pair Content (page 5-630)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
	swap(pair(AC0), pair(AC2))	Y	2	1	X	DU_SWAP	.	.	.	.	.	.	.	

Swap Auxiliary Register Content (page 5-631)

No.	Instruction	E	S	C	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	Notes
[1]	swap(AR0, AR1)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
[2]	swap(AR0, AR2)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[3]	swap(AR1, AR3)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Auxiliary Register Pair Content (page 5-632)														
	swap(pair(AR0), pair(AR2))	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Auxiliary and Temporary Register Content (page 5-633)														
[1]	swap(AR4, T0)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[2]	swap(AR5, T1)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[3]	swap(AR6, T2)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[4]	swap(AR7, T3)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Auxiliary and Temporary Register Pair Content (page 5-635)														
[1]	swap(pair(AR4), pair(T0))	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[2]	swap(pair(AR6), pair(T2))	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Auxiliary and Temporary Register Pairs Content (page 5-637)														
	swap(block(AR4), block(T0))	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Temporary Register Content (page 5-639)														
[1]	swap(T0, T2)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
[2]	swap(T1, T3)	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Swap Temporary Register Pair Content (page 5-640)														
	swap(pair(T0), pair(T2))	Y	2	1	AD	AU_SWAP	.	.	.	.	.	.	.	
Test Accumulator, Auxiliary, or Temporary Register Bit (page 5-641)														
[1]	TC1 = bit(src-AU, Baddr)	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	TC1 = bit(src-DU, Baddr)	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	
[2]	TC2 = bit(src-AU, Baddr)	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	TC2 = bit(src-DU, Baddr)	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Table 4−1. Algebraic Instruction Set Summary (Continued)

							Address Generation Unit			Buses				
No.	**Instruction**	**E**	**S**	**C**	**Pipe**	**Operator**	**DA**	**CA**	**SA**	**DR**	**CR**	**DW**	**ACB**	**Notes**
Test Accumulator, Auxiliary, or Temporary Register Bit Pair (page 5-643)														
	bit(src-AU, pair(Baddr))	N	3	1	X	AU_ALU	1	.	.	.	.	.	.	
	bit(src-DU, pair(Baddr))	N	3	1	X	DU_BIT	1	.	.	.	.	.	.	
Test Memory Bit (page 5-645)														
[1]	TCx = bit(Smem, src)	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
[2]	TCx = bit(Smem, k4)	N	3	1	X	AU_ALU	1	.	.	1	.	.	.	
Test and Clear Memory Bit (page 5-648)														
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #0	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
[2]	TC2 = bit(Smem, k4), bit(Smem, k4) = #0	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
Test and Complement Memory Bit (page 5-649)														
[1]	TC1 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
[2]	TC2 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
Test and Set Memory Bit (page 5-650)														
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	
[2]	TC2 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	X	AU_ALU	1	.	.	1	.	1	.	

Notes:
1) dst-DU, src-AU or dst-DU, src-DU
2) dst-DU, src-AU or dst-AU, src-DU

Chapter 5

Instruction Set Descriptions

This chapter provides detailed information on the TMS320C55x™ DSP algebraic instruction set.

See Section 1.1, *Instruction Set Terms, Symbols, and Abbreviations*, for definitions of symbols and abbreviations used in the description of each instruction. See Chapter 4 for a summary of the instruction set.

abdst *Absolute Distance*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**abdst(**Xmem, Ymem, ACx, ACy**)**	No	4	1	X

Opcode

```
|1000 0110|XXXM MMYY|YMMM DDDD|1111 xxn%
```

Operands ACx, ACy, Xmem, Ymem

Description This instruction executes two operations in parallel: one in the D-unit MAC and one in the D-unit ALU:

```
ACy = ACy + |HI(ACx)|
ACx = (Xmem << #16) − (Ymem << #16)
```

The absolute value of accumulator ACx content is computed and added to accumulator ACy content through the D-unit MAC. When an overflow is detected according to M40:

- ❑ the destination accumulator overflow status bit (ACOVy) is set
- ❑ the destination register (ACy) is saturated according to SATD

The Ymem content shifted left 16 bits is subtracted from the Xmem content shifted left 16 bits in the D-unit ALU.

- ❑ Input operands (Xmem and Ymem) are sign extended to 40 bits according to SXMD.
- ❑ CARRY status bit depends on M40. Subtraction borrow bit is reported in CARRY status bit. It is the logical complement of CARRY status bit.
- ❑ When an overflow is detected according to M40:
  - ■ the destination accumulator overflow status bit (ACOVx) is set
  - ■ the destination register (ACx) is saturated according to SATD

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

When C54CM = 1, the subtract operation does not have any overflow detection, report, and saturation after the shifting operation.

Status Bits		
	Affected by	C54CM, FRCT, M40, SATD, SXMD
	Affects	ACOVx, ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Square Distance

Example

Syntax	Description
abdst(*AR0+, *AR1, AC0, AC1)	The absolute value of the content of AC0 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 is subtracted from the content addressed by AR0 and the result is stored in AC0. The content of AR0 is incremented by 1.

```
Before                               After
AC0          00 0000 0000            AC0          00 4500 0000
AC1          00 E800 0000            AC1          00 E800 0000
AR0                   202            AR0                   203
AR1                   302            AR1                   302
202                  3400            202                  3400
302                  EF00            302                  EF00
ACOV0                   0            ACOV0                   0
ACOV1                   0            ACOV1                   0
CARRY                   0            CARRY                   0
M40                     1            M40                     1
SXMD                    1            SXMD                    1
```

ABS *Absolute Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = |src|	Yes	2	1	X

Opcode `| 0011  001E | FSSS  FDDD`

Operands dst, src

Description This instruction computes the absolute value of the source register (src).

- ❑ When the destination register (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
  - If M40 = 0, the sign of the source register is extracted at bit position 31. If src(31) = 1, the source register content is negated. If src(31) = 0, the source register content is moved to the destination accumulator.
  - If M40 = 1, the sign of the source register is extracted at bit position 39. If src(39) = 1, the source register content is negated. If src(39) = 0, the source register content is moved to the destination accumulator.
  - During the 40-bit move operation, an overflow and CARRY bit status are detected according to M40:
    - The destination accumulator overflow status bit (ACOVx) is set.
    - The destination register is saturated according to SATD.
    - The CARRY status bit is updated as follows: If the result of the operation stored in the destination register is 0, CARRY is set; otherwise, CARRY is cleared.
- ❑ When the destination register (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The sign of the source register is extracted at bit position 15. If src(15) = 1, the source register content is negated. If src(15) = 0, the source register content is moved to the destination register. Overflow is detected at bit position 15.
  - The destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When C54CM =1, this instruction is executed as if M40 status bit was locally set to 1. To ensure compatibility versus overflow detection and saturation of destination accumulator, this instruction must be executed with M40 = 0.

Status Bits Affected by C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Addition with Absolute Value

Example 1

Syntax	Description
AC1 = |AC0|	The absolute value of the content of AC0 is stored in AC1.

```
Before                         After
AC1         00 0000 2000       AC1         7D FFFF EDCC
AC0         82 0000 1234       AC0         82 0000 1234
M40                    1       M40                    1
```

Example 2

Syntax	Description
AC1 = |AR1|	The absolute value of the content of AR1 is stored in AC1.

```
Before                         After
AC1         00 0000 2000       AC1         00 0000 0000
AR1                 0000       AR1                 0000
CARRY                  0       CARRY                  1
```

Example 3

Syntax	Description
AC1 = |AR1|	The absolute value of the content of AR1 is stored in AC1. Since SXMD = 1, AR1 content is sign extended. The resulting 40-bit data is negated since M40 = 0 and AR1(31) = 1.

```
Before                         After
AC1         00 0000 2000       AC1         00 0000 7900
AR1                 8700       AR1                 8700
M40                    0       M40                    0
SXMD                   1       SXMD                   1
```

Example 4

Syntax	Description
T1 = |AC0|	The absolute value of the content of AC0(15−0) is stored in T1. The sign bit is extracted at AC0(15). Since AC0(15) = 0, T1 = AC0(15−0).

```
Before                          After
T1                2000          T1                1234
AC0        80 0002 1234         AC0        80 0002 1234
```

Example 5

Syntax	Description
T1 = |AC0|	The absolute value of the content of AC0(15−0) is stored in T1. The sign bit is extracted at AC0(15). Since AC0(15) = 1, T1 equals the negated value of AC0(15−0).

```
Before                          After
T1                2000          T1                6DCC
AC0        80 0002 9234         AC0        80 0002 9234
```

ADD *Addition*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	X
[2]	dst = dst + k4	Yes	2	1	X
[3]	dst = src + K16	No	4	1	X
[4]	dst = src + Smem	No	3	1	X
[5]	ACy = ACy + **(**ACx **<<** Tx**)**	Yes	2	1	X
[6]	ACy = ACy + **(**ACx **<< #**SHIFTW**)**	Yes	3	1	X
[7]	ACy = ACx + **(**K16 **<< #16)**	No	4	1	X
[8]	ACy = ACx + **(**K16 **<< #**SHFT**)**	No	4	1	X
[9]	ACy = ACx + **(**Smem **<<** Tx**)**	No	3	1	X
[10]	ACy = ACx + **(**Smem **<< #16)**	No	3	1	X
[11]	ACy = ACx + uns(Smem) + **CARRY**	No	3	1	X
[12]	ACy = ACx + uns(Smem)	No	3	1	X
[13]	ACy = ACx + **(**uns(Smem) **<< #**SHIFTW**)**	No	4	1	X
[14]	ACy = ACx + **dbl(**Lmem**)**	No	3	1	X
[15]	ACx = **(**Xmem **<< #16)** + **(**Ymem **<< #16)**	No	3	1	X
[16]	Smem = Smem + K16	No	4	1	X

Description These instructions perform an addition operation.

Status Bits

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

See Also

See the following other related instructions:

- ❑ Addition or Subtraction Conditionally
- ❑ Addition or Subtraction Conditionally with Shift
- ❑ Addition with Absolute Value
- ❑ Addition with Parallel Store Accumulator Content to Memory
- ❑ Addition, Subtraction, or Move Accumulator Content Conditionally
- ❑ Dual 16-Bit Additions
- ❑ Dual 16-Bit Addition and Subtraction
- ❑ Dual 16-Bit Subtraction and Addition
- ❑ Subtraction

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	X

Opcode `| 0010 010E | FSSS FDDD`

Operands dst, src

Description This instruction performs an addition operation between two registers.

- ❑ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Addition overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + AC1	The content of AC1 is added to the content of AC0 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst + k4	Yes	2	1	X

Opcode `0100 000E kkkk FDDD`

Operands dst, k4

Description This instruction performs an addition operation between a register content and a 4-bit unsigned constant, k4.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Overflow detection and CARRY status bit depends on M40.
  - ■ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ Addition overflow detection is done at bit position 15.
  - ■ When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + k4	The content of AC0 is added to an unsigned 4-bit value and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src + K16	No	4	1	X

Opcode `0111 1011 | KKKK KKKK | KKKK KKKK | FDDD FSSS`

Operands dst, K16, src

Description This instruction performs an addition operation between a register content and a 16-bit signed constant, K16.

- ❑ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Addition overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 + #2E00h	The content of AC0 is added to the signed 16-bit value (2E00h) and the result is stored in AC1.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src + Smem	No	3	1	X

Opcode

```
|1101  0110|AAAA  AAAI|FDDD  FSSS
```

Operands dst, Smem, src

Description This instruction performs an addition operation between a register content and the content of a memory (Smem) location.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - ■ The content of the memory location is sign extended to 40 bits according to SXMD.
  - ■ Overflow detection and CARRY status bit depends on M40.
  - ■ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - ■ Addition overflow detection is done at bit position 15.
  - ■ When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Status Bits	Affected by	M40, SATA, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
T1 = T0 + *AR3+	The content of T0 is added to the content addressed by AR3 and the result is stored in T1. AR3 is incremented by 1.

```
Before                       After
AR3              0302        AR3              0303
302              EF00        302              EF00
T0               3300        T0               3300
T1                  0        T1               2200
CARRY               0        CARRY               1
```

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy + **(**ACx **<<** Tx**)**	Yes	2	1	X

Opcode `0101 101E DDSS ss00`

Operands ACx, ACy, Tx

Description This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the content of Tx.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
- The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (AC1 << T0)	The content of AC1 shifted by the content of T0 is added to the content of AC0 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACy + **(**ACx **<<** #SHIFTW**)**	Yes	3	1	X

Opcode

```
|0001 000E|DDSS 0011|xxSH IFTW
```

Operands ACx, ACy, SHIFTW

Description This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the 6-bit value, SHIFTW.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (AC1 << #31)	The content of AC1 shifted left by 31 bits is added to the content of AC0 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx + **(**K16 **<< #16)**	No	4	1	X

Opcode `0111 1010 | KKKK KKKK | KKKK KKKK | SSDD 000x`

Operands ACx, ACy, K16

Description This instruction performs an addition operation between an accumulator content ACx and a 16-bit signed constant, K16, shifted left by 16 bits.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (#2E00h << #16)	A signed 16-bit value (2E00h) shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = ACx + **(**K16 **<<** #SHFT**)**	No	4	1	X

Opcode `| 0111 0000 | KKKK KKKK | KKKK KKKK | SSDD SHFT`

Operands ACx, ACy, K16, SHFT

Description This instruction performs an addition operation between an accumulator content ACx and a 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (#2E00h << #15)	A signed 16-bit value (2E00h) shifted left by 15 bits is added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = ACx + **(**Smem **<<** Tx**)**	No	3	1	X

Opcode | 1101 1101 | AAAA AAAI | SSDD ss00

Operands ACx, ACy, Tx, Smem

Description This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted by the content of Tx.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- ❑ An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
- ❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (*AR1 << T0)	The content addressed by AR1 shifted left by the content of T0 is added to the content of AC1 and the result is stored in AC0.

```
Before                              After
AC0          00 0000 0000           AC0          00 2330 0000
AC1          00 2300 0000           AC1          00 2300 0000
T0                   000C           T0                   000C
AR1                  0200           AR1                  0200
200                  0300           200                  0300
SXMD                    0           SXMD                    0
M40                     0           M40                     0
ACOV0                   0           ACOV0                   0
CARRY                   0           CARRY                   1
```

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = ACx + **(**Smem **<< #16)**	No	3	1	X

Opcode `| 1101 1110 | AAAA AAAI | SSDD 0100`

Operands ACx, ACy, Smem

Description This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted left by 16 bits.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

❑ The shift operation is equivalent to the signed shift instruction.

❑ Overflow detection and CARRY status bit depends on M40. If the result of the addition generates a carry, the CARRY status bit is set; otherwise, the CARRY status bit is not affected.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (*AR3 << #16)	The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx + uns(Smem) + **CARRY**	No	3	1	X

Opcode | 1101 1111 | AAAA AAAI | SSDD 100u

Operands ACx, ACy, Smem

Description This instruction performs an addition operation of the accumulator content ACx, the content of a memory (Smem) location, and the value of the CARRY status bit.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are extended to 40 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ❑ Overflow detection and CARRY status bit depends on M40.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by CARRY, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + uns(*AR3) + CARRY	The CARRY status bit and the unsigned content addressed by AR3 are added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = ACx + uns(Smem)	No	3	1	X

Opcode `1101 1111 | AAAA AAAI | SSDD 110u`

Operands ACx, ACy, Smem

Description This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are extended to 40 bits according to uns.

- If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
- If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.

❑ Overflow detection and CARRY status bit depends on M40.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SATD, SXMD
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + uns(*AR3)	The unsigned content addressed by AR3 is added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACy = ACx + **(**uns(Smem) **<< #SHIFTW)**	No	4	1	X

Opcode `| 1111  1001 | AAAA  AAAI | uxSH  IFTW | SSDD  00xx`

Operands ACx, ACy, SHIFTW, Smem

Description This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted by the 6-bit value, SHIFTW.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (uns(*AR3) << #31)	The unsigned content addressed by AR3 shifted left by 31 bits is added to the content of AC1 and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ACy = ACx + **dbl(**Lmem**)**	No	3	1	X

Opcode `| 1110 1101 | AAAA AAAI | SSDD 000n`

Operands ACx, ACy, Lmem

Description This instruction performs an addition operation between an accumulator content ACx and the content of data memory operand dbl(Lmem).

- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + dbl(*AR3+)	The content (long word) addressed by AR3 and AR3 + 1 is added to the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ACx = **(**Xmem **<< #16)** + **(**Ymem **<< #16)**	No	3	1	X

Opcode `1000 0001 | XXXM MMYY | YMMM 00DD`

Operands ACx, Xmem, Ymem

Description This instruction performs an addition operation between the content of data memory operand Xmem shifted left 16 bits, and the content of data memory operand Ymem shifted left 16 bits.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

❑ The shift operation is equivalent to the signed shift instruction.

❑ Overflow detection and CARRY status bit depends on M40.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = (*AR3 << #16) + (*AR4 << #16)	The content addressed by AR3 shifted left by 16 bits is added to the content addressed by AR4 shifted left by 16 bits and the result is stored in AC0.

Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	Smem = Smem + K16	No	4	1	X

Opcode `| 1111 0111 | AAAA AAAI | KKKK KKKK | KKKK KKKK`

Operands K16, Smem

Description This instruction performs an addition operation between a 16-bit signed constant, K16, and the content of a memory (Smem) location.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD and shifted by 16 bits to the MSBs before being added.
- Addition overflow is detected at bit position 31. If an overflow is detected, accumulator 0 overflow status bit (ACOV0) is set.
- Addition carry report in CARRY status bit is extracted at bit position 31.
- If SATD is 1 when an overflow is detected, the result is saturated before being stored in memory. Saturation values are 7FFFh or 8000h.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by SATD, SXMD

Affects ACOV0, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = *AR3 + #2E00h	The content addressed by AR3 is added to a signed 16-bit value (2E00h) and the result is stored back into the location addressed by AR3.

ADDV Addition with Absolute Value

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + |ACx|)	Yes	2	1	X

Opcode | 0101 010E | DDSS 000% |

Operands ACx, ACy

Description This instruction computes the absolute value of accumulator ACx and adds the result to accumulator ACy. This instruction is performed in the D-unit MAC:

❏ The absolute value of accumulator ACx is computed by multiplying ACx(32–16) by 00001h or 1FFFFh depending on bit 32 of the source accumulator.

❏ If FRCT = 1, the absolute value is multiplied by 2.

❏ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❏ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.

❏ When an addition overflow is detected, the accumulator is saturated according to SATD.

❏ The result of the absolute value of the higher part of ACx is in the lower part of ACy.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❏ Absolute Value
- ❏ Addition
- ❏ Addition or Subtraction Conditionally
- ❏ Addition or Subtraction Conditionally with Shift
- ❏ Addition, Subtraction, or Move Accumulator Content Conditionally

Example

Syntax	**Description**
AC0 = AC0 + |AC1|	The absolute value of AC1 is added to the content of AC0 and the result is stored in AC0.

ADD::MOV *Addition with Parallel Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx + **(**Xmem **<< #16),** Ymem = **HI(**ACy **<< T2)**	No	4	1	X

Opcode

```
|1000 0111|XXXM MMYY|YMMM SSDD|100x xxxx
```

Operands ACx, ACy, T2, Xmem, Ymem

Description This instruction performs two operations in parallel: addition and store.

The first operation performs an addition between an accumulator content ACx and the content of data memory operand Xmem shifted left by 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
- When an overflow is detected, the accumulator is saturated according to SATD.

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACy(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

- If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = ACx + (Xmem << #16),
  Ymem = HI(saturate(uns(ACy << T2)))

- If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = ACx + (Xmem << #16),
  Ymem = HI(saturate(ACy << T2))

Status Bits Affected by C54CM, M40, RDM, SATD, SST, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Addition
- Store Accumulator Content to Memory
- Subtraction with Parallel Store Accumulator Content to Memory

Example

Syntax	Description
AC0 = AC1 + (*AR3 << #16), *AR4 = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31−16) is stored at the address of AR4.

ADDSUBCC *Addition or Subtraction Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **adsc(**Smem, ACx**, TC1)**	No	3	1	X
[2]	ACy = **adsc(**Smem, ACx**, TC2)**	No	3	1	X

Opcode

```
TC1    1101 1110 | AAAA AAAI | SSDD 0000
TC2    1101 1110 | AAAA AAAI | SSDD 0001
```

Operands ACx, ACy, Smem, TCx

Description This instruction evaluates the selected TCx status bit and based on the result of the test, either an addition or a subtraction is performed. Evaluation of the condition on the TCx status bit is performed during the Execute phase of the instruction.

TC1 or TC2	Operation
0	ACy = ACx − (Smem << #16)
1	ACy = ACx + (Smem << #16)

- **TCx = 0**, then ACy = ACx − (Smem << #16):

  This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from accumulator ACx and stores the result in accumulator ACy.

  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - The shift operation is equivalent to the signed shift instruction.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.

- **TCx = 1**, then ACy = ACx + (Smem << #16):

  This instruction performs an addition operation between accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.

- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD, TCx

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❏ Addition or Subtraction Conditionally with Shift

❏ Addition, Subtraction, or Move Accumulator Content Conditionally

Example 1

Syntax	Description
AC0 = adsc(*AR3, AC1, TC1)	If TC1 = 1, the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

Example 2

Syntax	Description
AC1 = adsc(*AR1, AC0, TC2)	TC2 = 1, the content addressed by AR1 shifted left by 16 bits is added to the content of AC0 and the result is stored in AC1. The result generated an overflow and a carry.

```
Before                          After
AC0         00 EC00 0000        AC0         00 EC00 0000
AC1         00 0000 0000        AC1         01 1F00 0000
AR1                 0200        AR1                 0200
200                 3300        200                 3300
TC2                    1        TC2                    1
SXMD                   0        SXMD                   0
M40                    0        M40                    0
ACOV1                  0        ACOV1                  1
CARRY                  0        CARRY                  1
```

ADDSUB2CC *Addition or Subtraction Conditionally with Shift*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **ads2c(**Smem, ACx, Tx**, TC1, TC2)**	No	3	1	X

Opcode

```
|1101  1101|AAAA  AAAI|SSDD  ss10
```

Operands ACx, ACy, Tx, Smem, TC1, TC2

Description This instruction evaluates the TC1 status bit and based on the result of the test, either an addition or a subtraction is performed; this instruction evaluates the TC2 status bit and based on the result of the test, either a shift left by 16 bits or the content of Tx is performed. Evaluation of the condition on the TCx status bits is performed during the Execute phase of the instruction.

TC1	TC2	Operation
0	0	ACy = ACx − (Smem << Tx)
0	1	ACy = ACx − (Smem << #16)
1	0	ACy = ACx + (Smem << Tx)
1	1	ACy = ACx + (Smem << #16)

❑ **TC1 = 0 and TC2 = 0**, then ACy = ACx − (Smem << Tx):

This instruction subtracts the content of a memory (Smem) location shifted left by the content of Tx from an accumulator ACx and stores the result in accumulator ACy.

❑ **TC1 = 0 and TC2 = 1**, then ACy = ACx − (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

❑ **TC1 = 1 and TC2 = 0**, then ACy = ACx + (Smem << Tx):

This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by the content of Tx and stores the result in accumulator ACy.

❑ **TC1 = 1 and TC2 = 1**, then ACy = ACx + (Smem << #16):

This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

❑ An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits	Affected by	C54CM, M40, SATD, SXMD, TC1, TC2
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Addition or Subtraction Conditionally

❑ Addition, Subtraction, or Move Accumulator Content Conditionally

Example

Syntax	Description
AC2 = ads2c(*AR2, AC0, T1, TC1, TC2)	TC1 = 1 and TC2 = 0, the content addressed by AR2 shifted left by the content of T1 is added to the content of AC0 and the result is stored in AC2. The result generated an overflow.

```
Before                                 After
AC0          00 EC00 0000              AC0          00 EC00 0000
AC2          00 0000 0000              AC2          00 EC00 CC00
AR2                  0201              AR2                  0201
201                  3300              201                  3300
T1                   0002              T1                   0002
TC1                     1              TC1                     1
TC2                     0              TC2                     0
M40                     0              M40                     0
ACOV2                   0              ACOV2                   1
CARRY                   0              CARRY                   0
```

ADDSUBCC *Addition, Subtraction, or Move Accumulator Content Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **adsc(**Smem, ACx**, TC1, TC2)**	No	3	1	X

Opcode `|1101 1110|AAAA AAAI|SSDD 0010`

Operands ACx, ACy, Smem, TC1, TC2

Description This instruction evaluates the TCx status bits and based on the result of the test, an addition, a move, or a subtraction is performed. Evaluation of the condition on the TCx status bits is performed during the Execute phase of the instruction.

TC1	TC2	Operation
0	0	ACy = ACx − (Smem << #16)
0	1	ACy = ACx
1	0	ACy = ACx + (Smem << #16)
1	1	ACy = ACx

❑ **TC2 = 1**, then ACy = ACx:

This instruction moves the content of ACx to ACy.

- The 40-bit move operation is performed in the D-unit ALU.
- During the 40-bit move operation, an overflow is detected according to M40:
  - the destination accumulator overflow status bit (ACOVy) is set.
  - the destination register (ACy) is saturated according to SATD.

❑ **TC1 = 0 and TC2 = 0**, then ACy = ACx − (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

- ❑ **TC1 = 1 and TC2 = 0**, then ACy = ACx + (Smem << #16):

  This instruction performs an addition operation between accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.

  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are sign extended to 40 bits according to SXMD.
  - ■ The shift operation is equivalent to the signed shift instruction.
  - ■ Overflow detection and CARRY status bit depends on M40.
  - ■ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by	C54CM, M40, SATD, SXMD, TC1, TC2
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Addition or Subtraction Conditionally
- ❑ Addition or Subtraction Conditionally with Shift

Example

Syntax	**Description**
AC0 = adsc(*AR3, AC1, TC1, TC2)	If TC2 = 1, the content of AC1 is stored in AC0. If TC2 = 0 and TC1 = 1, the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC2 = 0 and TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

AND *Bitwise AND*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src	Yes	2	1	X
[2]	dst = src & k8	Yes	3	1	X
[3]	dst = src & k16	No	4	1	X
[4]	dst = src & Smem	No	3	1	X
[5]	ACy = ACy & **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X
[6]	ACy = ACx & **(**k16 **<<< #16)**	No	4	1	X
[7]	ACy = ACx & **(**k16 **<<<** #SHFT**)**	No	4	1	X
[8]	Smem = Smem & k16	No	4	1	X

Description These instructions perform a bitwise AND operation:

- In the D-unit, if the destination operand is an accumulator.
- In the A-unit ALU, if the destination operand is an auxiliary or temporary register.
- In the A-unit ALU, if the destination operand is the memory.

Status Bits

Affected by	C54CM
Affects	none

See Also See the following other related instructions:

- Bitwise AND Memory with Immediate Value and Compare to Zero
- Bitwise OR
- Bitwise Exclusive OR (XOR)

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src	Yes	2	1	X

Opcode

```
| 0010  100E | FSSS  FDDD
```

Operands dst, src

Description This instruction performs a bitwise AND operation between two registers.

- ❑ When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 & AC0	The content of AC0 is ANDed with the content of AC1 and the result is stored in AC1.

```
Before                                  After
AC0            7E 2355 4FC0             AC0            7E 2355 4FC0
AC1            0F E340 5678             AC1            0E 2340 4640
```

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = src & k8	Yes	3	1	X

Opcode `| 0001 100E | kkkk kkkk | FDDD FSSS`

Operands dst, k8, src

Description This instruction performs a bitwise AND operation between a source (src) register content and an 8-bit value, k8.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 & #FFh	The content of AC1 is ANDed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src & k16	No	4	1	X

Opcode `| 0111 1101 | kkkk kkkk | kkkk kkkk | FDDD FSSS`

Operands dst, k16, src

Description This instruction performs a bitwise AND operation between a source (src) register content and a 16-bit unsigned constant, k16.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 & #FFFFh	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src & Smem	No	3	1	X

Opcode

```
| 1101  1001 | AAAA  AAAI | FDDD  FSSS
```

Operands dst, Smem, src

Description This instruction performs a bitwise AND operation between a source (src) register content and a memory (Smem) location.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 & *AR3	The content of AC1 is ANDed with the content addressed by AR3 and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy & **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X

Opcode `0001 000E DDSS 0000 xxSH IFTW`

Operands ACx, ACy, SHIFTW

Description This instruction performs a bitwise AND operation between an accumulator (ACy) content and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.

- ❑ The shift and AND operations are performed in one cycle in the D-unit shifter.
- ❑ When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.
- ❑ The input operand (ACx) is shifted by a 6-bit immediate value in the D-unit shifter.
- ❑ The CARRY status bit is not affected by the logical shift operation.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the intermediary logical shift is performed as if M40 is locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits Affected by C54CM, M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 & (AC1 <<< #30)	The content of AC0 is ANDed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx & **(**k16 **<<< #16)**	No	4	1	X

Opcode

```
|0111  1010|kkkk  kkkk|kkkk  kkkk|SSDD  010x
```

Operands ACx, ACy, k16

Description This instruction performs a bitwise AND operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are zero extended to 40 bits.
- ❑ The input operand (k16) is shifted 16 bits to the MSBs.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 & (#FFFFh <<< #16)	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx & **(**k16 **<<<** #SHFT**)**	No	4	1	X

Opcode `| 0111 0010 | kkkk kkkk | kkkk kkkk | SSDD SHFT`

Operands ACx, ACy, k16, SHFT

Description This instruction performs a bitwise AND operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by the 4-bit value, SHFT.

❑ The shift and AND operations are performed in one cycle in the D-unit shifter.

❑ When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.

❑ The input operand (k16) is shifted by a 4-bit immediate value in the D-unit shifter.

❑ The CARRY status bit is not affected by the logical shift operation.

Status Bits Affected by C54CM, M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 & (#FFFFh <<< #15)	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

Bitwise AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem & k16	No	4	1	X

Opcode

```
|1111 0100|AAAA AAAI|kkkk kkkk|kkkk kkkk
```

Operands k16, Smem

Description This instruction performs a bitwise AND operation between a memory (Smem) location and a 16-bit unsigned constant, k16.

❑ The operation is performed on 16 bits in the A-unit ALU.

❑ The result is stored in memory.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR1 = *AR1 & #0FC0	The content addressed by AR1 is ANDed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

```
Before                      After
*AR1            5678        *AR1            0640
```

BAND *Bitwise AND Memory with Immediate Value and Compare to Zero*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = Smem & k16	No	4	1	X
[2]	**TC2** = Smem & k16	No	4	1	X

Opcode

```
TC1   | 1111 0010 | AAAA AAAI | kkkk kkkk | kkkk kkkk
TC2   | 1111 0011 | AAAA AAAI | kkkk kkkk | kkkk kkkk
```

Operands k16, Smem, TCx

Description This instruction performs a bit field manipulation in the A-unit ALU. The 16-bit field mask, k16, is ANDed with the memory (Smem) operand and the result is compared to 0:

```
if( ((Smem) AND k16 ) == 0)
   TCx = 0
else
   TCx = 1
```

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Bitwise AND

Example

Syntax	Description
TC1 = *AR0 & #0060h	The unsigned 16-bit value (0060h) is ANDed with the content addressed by AR0. The result is 1, TC1 is set to 1.

```
Before                  After
*AR0           0040     *AR0           0040
TC1               0     TC1               1
```

OR *Bitwise OR*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst | src	Yes	2	1	X
[2]	dst = src | k8	Yes	3	1	X
[3]	dst = src | k16	No	4	1	X
[4]	dst = src | Smem	No	3	1	X
[5]	ACy = ACy | **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X
[6]	ACy = ACx | **(**k16 **<<< #16)**	No	4	1	X
[7]	ACy = ACx | **(**k16 **<<<** #SHFT**)**	No	4	1	X
[8]	Smem = Smem | k16	No	4	1	X

Description These instructions perform a bitwise OR operation:

- In the D-unit, if the destination operand is an accumulator.
- In the A-unit ALU, if the destination operand is an auxiliary or temporary register.
- In the A-unit ALU, if the destination operand is the memory.

Status Bits Affected by C54CM

Affects none

See Also See the following other related instructions:

- Bitwise AND
- Bitwise Exclusive OR (XOR)

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst | src	Yes	2	1	X

Opcode `| 0010 101E | FSSS FDDD`

Operands dst, src

Description This instruction performs a bitwise OR operation between two registers.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 | AC1	The content of AC0 is ORed with the content of AC1 and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = src | k8	Yes	3	1	X

Opcode `| 0001 101E | kkkk kkkk | FDDD FSSS`

Operands dst, k8, src

Description This instruction performs a bitwise OR operation between a source (src) register content and an 8-bit value, k8.

❑ When the destination (dst) operand is an accumulator:

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are zero extended to 40 bits.
- If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.

❑ When the destination (dst) operand is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 | #FFh	The content of AC1 is ORed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src | k16	No	4	1	X

Opcode `| 0111 1110 | kkkk kkkk | kkkk kkkk | FDDD FSSS`

Operands dst, k16, src

Description This instruction performs a bitwise OR operation between a source (src) register content and a 16-bit unsigned constantk16.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 | #FFFFh	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src | Smem	No	3	1	X

Opcode `| 1101 1010 | AAAA AAAI | FDDD FSSS`

Operands dst, Smem, src

Description This instruction performs a bitwise OR operation between a source (src) register content and a memory (Smem) location.

❑ When the destination (dst) operand is an accumulator:

- ■ The operation is performed on 40 bits in the D-unit ALU.
- ■ Input operands are zero extended to 40 bits.
- ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.

❑ When the destination (dst) operand is an auxiliary or temporary register:

- ■ The operation is performed on 16 bits in the A-unit ALU.
- ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 | *AR3	The content of AC1 is ORed with the content addressed by AR3 and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy | **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X

Opcode

```
|0001  000E|DDSS  0001|xxSH  IFTW
```

Operands ACx, ACy, SHIFTW

Description This instruction performs a bitwise OR operation between an accumulator (ACy) content and and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.

- ❑ The shift and OR operations are performed in one cycle in the D-unit shifter.
- ❑ When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.
- ❑ The input operand (ACx) is shifted by a 6-bit immediate value in the D-unit shifter.
- ❑ The CARRY status bit is not affected by the logical shift operation.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the intermediary logical shift is performed as if M40 is locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits

Affected by	C54CM, M40
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 | (AC0 <<< #4)	The content of AC1 is ORed with the content of AC0 logically shifted left by 4 bits and the result is stored in AC1.

```
Before                            After
AC0         7E 2355 4FC0          AC0         7E 2355 4FC0
AC1         0F E340 5678          AC1         0F F754 FE78
```

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx | **(**k16 **<<< #16)**	No	4	1	X

Opcode `0111 1010 | kkkk kkkk | kkkk kkkk | SSDD 011x`

Operands ACx, ACy, k16

Description This instruction performs a bitwise OR operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are zero extended to 40 bits.
- ❑ The input operand (k16) is shifted 16 bits to the MSBs.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 | (#FFFFh <<< #16)	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx | **(**k16 **<<<** #SHFT**)**	No	4	1	X

Opcode `|0111 0011|kkkk kkkk|kkkk kkkk|SSDD SHFT`

Operands ACx, ACy, k16, SHFT

Description This instruction performs a bitwise OR operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by the 4-bit value, SHFT.

- The shift and OR operations are performed in one cycle in the D-unit shifter.
- When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.
- The input operand (k16) is shifted by a 4-bit immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation

Status Bits Affected by C54CM, M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 | (#FFFFh <<< #15)	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

Bitwise OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem | k16	No	4	1	X

Opcode | `1111 0101 | AAAA AAAI | kkkk kkkk | kkkk kkkk`

Operands k16, Smem

Description This instruction performs a bitwise OR operation between a memory (Smem) location and a 16-bit unsigned constant, k16.

- ❑ The operation is performed on 16 bits in the A-unit ALU.
- ❑ The result is stored in memory.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR1 = *AR1 | #0FC0h	The content addressed by AR1 is ORed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

```
Before                 After
*AR1        5678       *AR1        5FF8
```

XOR *Bitwise Exclusive OR (XOR)*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src	Yes	2	1	X
[2]	dst = src ^ k8	Yes	3	1	X
[3]	dst = src ^ k16	No	4	1	X
[4]	dst = src ^ Smem	No	3	1	X
[5]	ACy = ACy ^ **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X
[6]	ACy = ACx ^ **(**k16 **<<< #16)**	No	4	1	X
[7]	ACy = ACx ^ **(**k16 **<<<** #SHFT**)**	No	4	1	X
[8]	Smem = Smem ^ k16	No	4	1	X

Description These instructions perform a bitwise exclusive-OR (XOR) operation:

- ❑ In the D-unit, if the destination operand is an accumulator.
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register.
- ❑ In the A-unit ALU, if the destination operand is the memory.

Status Bits

Affected by	C54CM
Affects	none

See Also See the following other related instructions:

- ❑ Bitwise AND
- ❑ Bitwise OR

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src	Yes	2	1	X

Opcode | `0010 110E | FSSS FDDD`

Operands dst, src

Description This instruction performs a bitwise exclusive-OR (XOR) operation between two registers.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits

	Affected by	none
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 ^ AC0	The content of AC0 is XORed with the content of AC1 and the result is stored in AC1.

```
Before                       After
AC0         7E 2355 4FC0     AC0         7E 2355 4FC0
AC1         0F E340 5678     AC1         71 C015 19B8
```

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = src ^ k8	Yes	3	1	X

Opcode `0001 110E | kkkk kkkk | FDDD FSSS`

Operands dst, k8, src

Description This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and an 8-bit value, k8.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 ^ #FFh	The content of AC1 is XORed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src ^ k16	No	4	1	X

Opcode `0111 1111 | kkkk kkkk | kkkk kkkk | FDDD FSSS`

Operands dst, k16, src

Description This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and a 16-bit unsigned constant, k16.

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ Input operands are zero extended to 40 bits.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 ^ #FFFFh	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src ^ Smem	No	3	1	X

Opcode

```
| 1101  1011 | AAAA  AAAI | FDDD  FSSS
```

Operands dst, Smem, src

Description This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and a memory (Smem) location.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are zero extended to 40 bits.
  - If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- When the destination (dst) operand is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 ^ *AR3	The content of AC1 is XORed with the content addressed by AR3 and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy ^ **(**ACx **<<<** #SHIFTW**)**	Yes	3	1	X

Opcode `| 0001 000E | DDSS 0010 | xxSH IFTW`

Operands ACx, ACy, SHIFTW

Description This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACy) content and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.

- The shift and XOR operations are performed in one cycle in the D-unit shifter.
- When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.
- The input operand (ACx) is shifted by a 6-bit immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the intermediary logical shift is performed as if M40 is locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.

Status Bits Affected by C54CM, M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 ^ (AC1 <<< #30)	The content of AC0 is XORed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx ^ **(**k16 **<<< #16)**	No	4	1	X

Opcode `| 0111 1010 | kkkk kkkk | kkkk kkkk | SSDD 100x`

Operands ACx, ACy, k16

Description This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are zero extended to 40 bits.
- ❑ The input operand (k16) is shifted 16 bits to the MSBs.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 ^ (#FFFFh <<< #16)	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx ^ **(**k16 **<<<** #SHFT**)**	No	4	1	X

Opcode | `0111 0100` | `kkkk kkkk` | `kkkk kkkk` | `SSDD SHFT`

Operands ACx, ACy, k16, SHFT

Description This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by the 4-bit value, SHFT.

❑ The shift and XOR operations are performed in one cycle in the D-unit shifter.

❑ When M40 = 0 and C54CM = 0, input operands ACx(31−0) are zero extended to 40 bits. Otherwise, ACx(39−0) is used as is.

❑ The input operand (k16) is shifted by a 4-bit immediate value in the D-unit shifter.

❑ The CARRY status bit is not affected by the logical shift operation.

Status Bits Affected by C54CM, M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 ^ (#FFFFh <<< #15)	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

Bitwise Exclusive OR (XOR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem ^ k16	No	4	1	X

Opcode

```
| 1111  0110 | AAAA  AAAI | kkkk  kkkk | kkkk  kkkk
```

Operands k16, Smem

Description This instruction performs a bitwise exclusive-OR (XOR) operation between a memory (Smem) location and a 16-bit unsigned constant, k16.

- The operation is performed on 16 bits in the A-unit ALU.
- The result is stored in memory.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = *AR3 ^ #FFFFh	The content addressed by AR3 is XORed with the unsigned 16-bit value (FFFFh) and the result is stored in the location addressed by AR3.

BCC *Branch Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[1]	**if (**cond**) goto** l4	No	2	6/5	R
[2]	**if (**cond**) goto** L8	Yes	3	6/5	R
[3]	**if (**cond**) goto** L16	No	4	6/5	R
[4]	**if (**cond**) goto** P24	No	5	5/5	R

[†] x/y cycles: x cycles = condition true, y cycles = condition false

Description These instructions evaluate a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into l4, Lx, or P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

The instruction selection depends on the branch offset between the current PC value and the program branch address specified by the label.

These instructions cannot be repeated.

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

See Also See the following other related instructions:

- ❑ Branch Unconditionally
- ❑ Branch on Auxiliary Register Not Zero
- ❑ Call Conditionally
- ❑ Compare and Branch

Branch Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	**if (**cond**) goto** l4	No	2	6/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

Opcode |0110 0111|1CCC CCCC

Operands cond, l4

Description This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into l4. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx
	Affects	ACOVx

Repeat This instruction cannot be repeated.

Example

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

```
         if (AC0 != #0) goto branch
         … …                            address:  004057
         … …
branch   ……                                       00405A
:
```

```
Before                          After
AC0        00 0000 3000         AC0        00 0000 3000
PC               004055         PC               00405A
```

Branch Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[2]	**if (**cond**) goto** L8	Yes	3	6/5	R
[3]	**if (**cond**) goto** L16	No	4	6/5	R

[†] x/y cycles: x cycles = condition true, y cycles = condition false

Opcode

```
 L8                      |0000  010E|xCCC  CCCC|LLLL  LLLL
 L16          |0110  1101|xCCC  CCCC|LLLL  LLLL|LLLL  LLLL
```

Operands cond, Lx

Description This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into Lx. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx
	Affects	ACOVx

Repeat This instruction cannot be repeated.

Example

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

```
branch   ……                                  00305A
:
         if (AC0 != #0) goto branch
         … …                       address:  004057
         … …
```

```
Before                        After
AC0          00 0000 3000     AC0          00 0000 3000
PC                 004055     PC                 00305A
```

Branch Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[4]	**if (**cond**) goto** P24	No	5	5/5	R

[†] x/y cycles: x cycles = condition true, y cycles = condition false

Opcode `|0110 1000|xCCC CCCC|PPPP PPPP|PPPP PPPP|PPPP PPPP`

Operands cond, P24

Description This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

Repeat This instruction cannot be repeated.

Example

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

```
          .sect “code1”
          … …
          if (AC0 != #0) goto branch
          … …                               address:  004057
          .sect “code2”
branch    ……                                          00F05A
:
```

```
Before                          After
AC0          00 0000 3000       AC0          00 0000 3000
PC                 004055       PC                 00F05A
```

B *Branch Unconditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**goto** ACx	No	2	10	X
[2]	**goto** L7	Yes	2	6[†]	AD
[3]	**goto** L16	Yes	3	6[†]	AD
[4]	**goto** P24	No	4	5	D

[†] This instruction executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

Description This instruction branches to a 24-bit program address defined by the content of the 24 lowest bits of an accumulator (ACx), or to a program address defined by the program address label assembled into Lx or P24.

These instructions cannot be repeated.

Status Bits

Affected by	none
Affects	none

See Also See the following other related instructions:

- ❑ Branch Conditionally
- ❑ Branch on Auxiliary Register Not Zero
- ❑ Call Unconditionally
- ❑ Compare and Branch

Branch Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**goto** ACx	No	2	10	X

Opcode | 1001 0001 | xxxx xxSS

Operands ACx

Description This instruction branches to a 24-bit program address defined by the content of the 24 lowest bits of an accumulator (ACx).

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
goto AC0	Program control is passed to the program address defined by the content of AC0(23–0).

```
Before                         After
AC0         00 0000 403D       AC0         00 0000 403D
PC                001F0A       PC                00403D
```

Branch Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[2]	**goto** L7	Yes	2	6	AD
[3]	**goto** L16	Yes	3	6	AD

[†] Executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

Opcode

```
 L7                          | 0100  101E | 0LLL  LLLL
L16              | 0000  011E | LLLL  LLLL | LLLL  LLLL
```

Operands Lx

Description This instruction branches to a program address defined by a program address label assembled into Lx.

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

```
         goto branch
         AC0 = #1                       address:  004044
         … …
branch:  … …                                      006047
         AC0 = #0
```

```
Before                         After
PC              004042         PC              006047
AC0       00 0000 0001         AC0       00 0000 0000
```

Branch Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	**goto** P24	No	4	5	D

Opcode | `0110 1010` | `PPPP PPPP` | `PPPP PPPP` | `PPPP PPPP`

Operands P24

Description This instruction branches to a program address defined by a program address label assembled into P24.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

```
         goto branch
         AC0 = #1                         address:  004044
         … …
branch:  … …                                        006047
         AC0 = #0
```

```
Before                          After
PC              004042          PC              006047
AC0       00 0000 0001          AC0       00 0000 0000
```

BCC *Branch on Auxiliary Register Not Zero*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[1]	**if (**ARn_mod **!= #0) goto** L16	No	4	6/5	AD

[†] x/y cycles: x cycles = condition true, y cycles = condition false

Opcode

```
|1111  1100|AAAA  AAAI|LLLL  LLLL|LLLL  LLLL
```

Operands ARn_mod, L16

Description This instruction performs a conditional branch (selected auxiliary register content not equal to 0) of the program counter (PC). The program branch address is specified as a 16-bit signed offset, L16, relative to PC. Use this instruction to branch within a 64K-byte window centered on the current PC value.

The possible addressing operands can be grouped into three categories:

- ARx not modified (ARx as base pointer), some examples:
  *AR1; No modification or offset
  *AR1(#15); Use 16-bit immediate value (15) as offset
  *AR1(T0); Use content of T0 as offset
  *AR1(short(#4)); Use 3-bit immediate value (4) as offset

- ARx modified before being compared to 0, some examples:
  *−AR1; Decrement by 1 before comparison
  *+AR1(#20); Add 16-bit immediate value (20) before comparison

- ARx modified after being compared to 0, some examples:
  *AR1+; Increment by 1 after comparison
  *(AR1 − T1); Subtract content of T1 after comparison

1) The content of the selected auxiliary register (ARn) is premodified in the address generation unit.

2) The (premodified) content of ARn is compared to 0 and sets the condition in the address phase of the pipeline.

3) If the condition is not true, a branch occurs. If the condition is true, the instructions are executed in sequence.

4) The content of ARn is postmodified in the address generation unit.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

The premodifier *ARn(T0) is not available; *ARn(AR0) is available.

The postmodifiers *(ARn + T0) and *(ARn − T0) are not available; *(ARn + AR0) and *(ARn − AR0) are available.

The legality of the modifier usage is checked by the assembler when using the .c54cm_on and .c54cm_off assembler directives.

Status Bits Affected by C54CM

Affects none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- ❑ Branch Conditionally
- ❑ Branch Unconditionally
- ❑ Compare and Branch

Example 1

Syntax	Description
if (*AR1(#6) != #0) goto branch	The content of AR1 is compared to 0. The content is not 0, program control is passed to the program address label defined by branch.

```
         If (*AR1(#6) != #0) goto branch          address:  004004
         … …                                      ;         00400A


         … …
branch   … …                                      ;         00400C
:
```

```
Before                    After
AR1             0005      AR1             0005
PC            004004      PC            00400C
```

Example 2

Syntax	Description
if (*AR3− != #0) goto branch	The content of AR3 is compared to 0. The content is 0, program control is passed to the next instruction (the branch is not taken). AR3 is decremented by 1 after the comparison.

```
         If (*AR3– != #0) goto branch           address:  00400F
         … …                                    ;         004013
         … …
branch   … …                                    ;         004015
:
```

```
Before                        After
AR3              0000         AR3           FFFF
PC             00400F         PC          004013
```

CALLCC *Call Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	**if (**cond**) call** L16	No	4	6/5	R
[2]	**if (**cond**) call** P24	No	5	5/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

Description

These instructions evaluate a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into L16 or P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

The instruction selection depends on the branch offset between the current PC value and program subroutine address specified by the label.

These instructions cannot be repeated.

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

See Also See the following other related instructions:

- ❑ Branch Conditionally
- ❑ Call Unconditionally
- ❑ Return Conditionally
- ❑ Return Unconditionally

Call Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	**if (**cond**) call** L16	No	4	6/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

Opcode

```
|0110  1110|xCCC  CCCC|LLLL  LLLL|LLLL  LLLL
```

Operands cond, L16

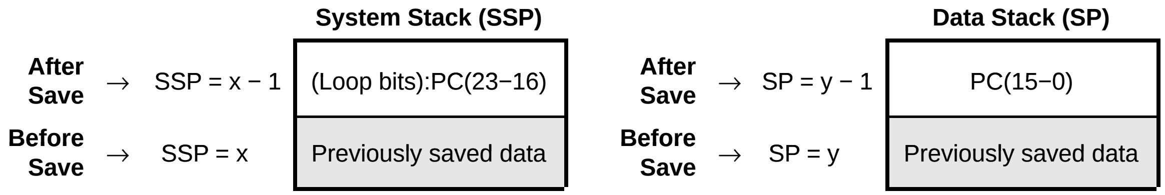
Description This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into L16. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

❑ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.

❑ The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.

❑ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.



Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx
	Affects	ACOVx
Repeat	This instruction cannot be repeated.	

Example

Syntax	Description
if (AC1 >= #2000h) call (subroutine)	The content of AC1 is equal to or greater than 2000h, control is passed to the program address label, subroutine. The program counter (PC) is loaded with the subroutine program address.

Call Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	**if (**cond**) call** P24	No	5	5/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

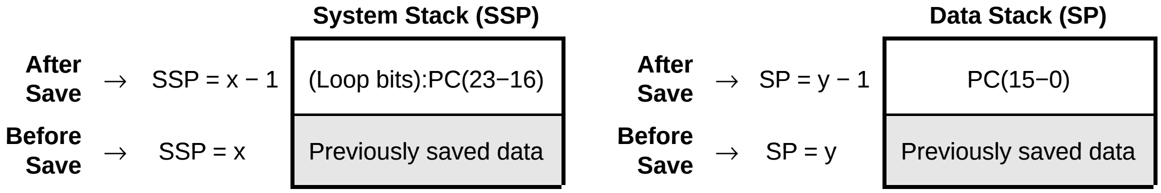
Opcode `| 0110 1001 | xCCC CCCC | PPPP PPPP | PPPP PPPP | PPPP PPPP`

Operands cond, P24

Description This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ❑ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ❑ The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ❑ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.



Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx
	Affects	ACOVx

Repeat This instruction cannot be repeated.

Example

Syntax	**Description**
if (TC1) call FOO	If TC1 is set to 1, control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value. If TC1 is cleared to 0, the program counter is incremented by 6 and the next instruction is executed.

CALL *Call Unconditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**call** ACx	No	2	10	X
[2]	**call** L16	Yes	3	6	AD
[3]	**call** P24	No	4	5	D

Description

This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx, or a program address label assembled into L16 or P24.

Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

These instructions cannot be repeated.

Status Bits

Affected by	none
Affects	none

See Also

See the following other related instructions:

❑ Branch Unconditionally

❑ Call Conditionally

❑ Return Conditionally

❑ Return Unconditionally

Call Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**call** ACx	No	2	10	X

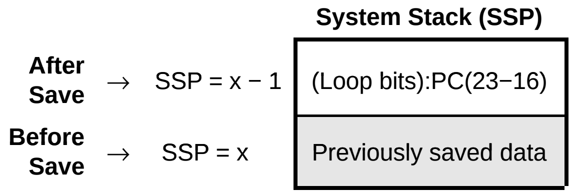
Opcode `1001 0010 xxxx xxSS`

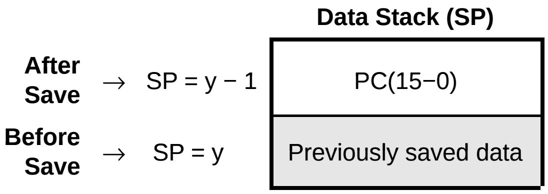
Operands ACx

Description This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ❑ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ❑ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ❑ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.





Status Bits		
	Affected by	none
	Affects	none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
call AC0	Program control is passed to the program address defined by the content of AC0(23–0).

Call Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**call** L16	Yes	3	6	AD

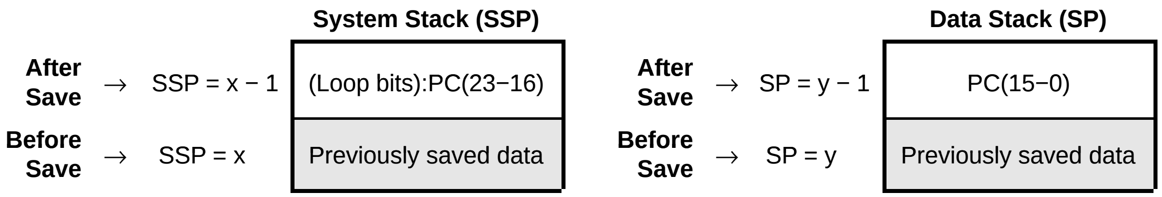
Opcode | `0000 100E | LLLL LLLL | LLLL LLLL`

Operands L16

Description This instruction passes control to a specified subroutine program address defined by a program address label assembled into L16.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.



Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
call FOO	Program control is passed to the program address label (FOO) assembled into the signed 16-bit offset value relative to the program counter register.

Call Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	**call** P24	No	4	5	D

Opcode `| 0110 1100 | PPPP PPPP | PPPP PPPP | PPPP PPPP`

Operands P24

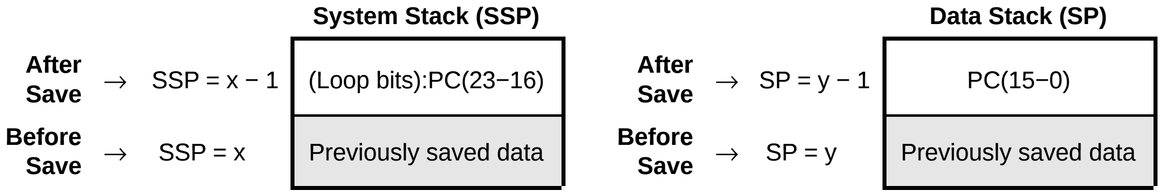
Description This instruction passes control to a specified subroutine program address defined by a program address label assembled into P24.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

❑ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.

❑ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.

❑ The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.



Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
call FOO	Program control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value.

.CR *Circular Addressing Qualifier*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**circular()**	No	1	1	AD

Opcode `1001 1101`

Operands none

Description This instruction is an instruction qualifier that can be paralleled only with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing or mar instructions. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone instruction (assembler generates an error message).

When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done circularly (as if ST2_55 register bits 0 to 8 were set to 1).

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction does not affect *ARn, *ARn+, *ARn−, and *ARn(DR0) addressing modes of dual memory access (Xmem/Ymem) instruction.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

BCLR — *Clear Accumulator, Auxiliary, or Temporary Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(**src, Baddr**)** = **#0**	No	3	1	X

Opcode

```
|1110 1100|AAAA AAAI|FSSS 001x
```

Operands Baddr, src

Description This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction clears to 0 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Clear Memory Bit
- Clear Status Register Bit
- Complement Accumulator, Auxiliary, or Temporary Register Bit
- Set Accumulator, Auxiliary, or Temporary Register Bit

Example

Syntax	Description
bit(AC0, AR3) = #0	The bit at the position defined by the content of AR3(4–0) in AC0 is cleared to 0.

BCLR *Clear Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(**Smem, src**)** = **#0**	No	3	1	X

Opcode `1110 0011 | AAAA AAAI | FSSS 1101`

Operands Smem, src

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction clears to 0 a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Clear Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Clear Status Register Bit
- ❑ Complement Memory Bit
- ❑ Set Memory Bit

Example

Syntax	Description
bit(*AR3, AC0) = #0	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is cleared to 0.

BCLR *Clear Status Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(ST0,** k4**)** = **#0**	Yes	2	1	X
[2]	**bit(ST1,** k4**)** = **#0**	Yes	2	1	X
[3]	**bit(ST2,** k4**)** = **#0**	Yes	2	1	X
[4]	**bit(ST3,** k4**)** = **#0**	Yes	2	1†	X

† When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Opcode	ST0	0100 011E	kkkk 0000
	ST1	0100 011E	kkkk 0010
	ST2	0100 011E	kkkk 0100
	ST3	0100 011E	kkkk 0110

Operands k4, STx

Description These instructions perform a bit manipulation in the A-unit ALU.

These instructions clear to 0 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0, ST1, ST2, or ST3).

It is not allowed to access DP register mapped in ST0 register with bit(ST0, k4) = #0 instruction. Therefore, k4 cannot have a value of 0−8.

It is not allowed to access ASM bit field in ST1 with bit(ST1, k4) = #0 instruction. Therefore, k4 cannot have a value of 0−4.

Compatibility with C54x devices (C54CM = 1)

C55x DSP status registers bit mapping (Figure 5−1, page 5-92) does not correspond to C54x DSP status register bits.

Status Bits Affected by none

Affects Selected status bits

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- Clear Accumulator, Auxiliary, or Temporary Register Bit
- Clear Memory Bit
- Set Status Register Bit

Example

Syntax	Description
bit(ST2, #ST2_AR2LC) = #0; AR2LC = bit 2	The ST2 bit position defined by the label (ST2_AR2LC, bit 2) is cleared to 0.

```
Before                      After
ST2_55          0006        ST2_55          0002
```

Figure 5−1. Status Registers Bit Mapping

ST0_55

15	14	13	12	11	10	9
ACOV2†	**ACOV3**†	**TC1**†	TC2	CARRY	ACOV0	ACOV1
R/W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−0	R/W−0

8	0
DP	
R/W−0	

ST1_55

15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	**M40**†	SATD	SXMD
R/W−0	R/W−0	R/W−1	R/W−0	R/W−1	R/W−0	R/W−0	R/W−1

7	6	5	4	0
C16	FRCT	**C54CM**†	ASM	
R/W−0	R/W−0	R/W−1	R/W−0	

ST2_55

15	14	13	12	11	10	9	8
ARMS	Reserved		DBGM	EALLOW	RDM	Reserved	CDPLC
R/W−0			R/W−1	R/W−0	R/W−0		R/W−0

7	6	5	4	3	2	1	0
AR7LC	AR6LC	AR5LC	AR4LC	AR3LC	AR2LC	AR1LC	AR0LC
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

ST3_55

15	14	13	12	11	8
CAFRZ†	**CAEN**†	**CACLR**†	**HINT**‡	Reserved (always write 1100b)	
R/W−0	R/W−0	R/W−0	R/W−1		

7	6	5	4	3	2	1	0
CBERR†	MPNMC§	**SATA**†	Reserved		CLKOFF	SMUL	SST
R/W−0	R/W−pins	R/W−0			R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; -*n* = Value after reset

† Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

‡ The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

§ The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

CMP Compare Accumulator, Auxiliary, or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = uns(src RELOP dst)	Yes	3	1	X
[2]	**TC2** = uns(src RELOP dst)	Yes	3	1	X

Opcode

TC1	0001 001E	FSSS cc00	FDDD xux0
TC2	0001 001E	FSSS cc00	FDDD xux1

Operands dst, RELOP, src, TCx

Description This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits	Affected by	C54CM, M40
	Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ❑ Compare Memory with Immediate Value

Example 1

Syntax	Description
TC1= AC1 = = T1	The signed content of AC1(15−0) is compared to the content of T1 and because they are equal, TC1 is set to 1.

```
Before                        After
AC1         00 0028 0400      AC1         00 0028 0400
T1                  0400      T1                  0400
TC1                    0      TC1                    1
```

Example 2

Syntax	Description
TC1= T1 > = AC1	The content of T1 is compared to the signed content of AC1(15−0). The content of T1 is greater than the content of AC1, TC1 is set to 1.

```
Before                        After
T1                  0500      T1                  0500
AC1         80 0000 0400      AC1         80 0000 0400
TC1                    0      TC1                    1
```

CMPAND *Compare Accumulator, Auxiliary, or Temporary Register Content with AND*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = TCy & uns(src RELOP dst)	Yes	3	1	X
[2]	TCx = **!**TCy & uns(src RELOP dst)	Yes	3	1	X

Description These instructions perform a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU.

Status Bits

Affected by	C54CM, M40, TCy
Affects	TCx

See Also See the following other related instructions:

- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ❑ Compare Memory with Immediate Value

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy & uns(src RELOP dst)				
[1a]	**TC1** = **TC2** & uns(src RELOP dst)	Yes	3	1	X
[1b]	**TC2** = **TC1** & uns(src RELOP dst)	Yes	3	1	X

Opcode | 0001 001E | FSSS cc01 | FDDD 0utt

Operands dst, RELOP, src, TC1, TC2

Description This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	If M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	If M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits

	Affected by	C54CM, M40, TCy
	Affects	TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC2 = TC1 & AC1 == AC2	The content of AC1(31−0) is compared to the content of AC2(31−0). The contents are equal (true), TC2 = TC1 & 1.

```
Before                            After
AC1          80 0028 0400         AC1          80 0028 0400
AC2          00 0028 0400         AC2          00 0028 0400
M40                     0         M40                     0
TC1                     1         TC1                     1
TC2                     0         TC2                     1
```

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = **!**TCy & uns(src RELOP dst)				
[2a]	**TC1** = **!TC2** & uns(src RELOP dst)	Yes	3	1	X
[2b]	**TC2** = **!TC1** & uns(src RELOP dst)	Yes	3	1	X

Opcode | 0001 001E | FSSS cc01 | FDDD 1utt

Operands dst, RELOP, src, TC1, TC2

Description This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40, TCy

Affects TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC2 = !TC1 & AC1 == AC2	The content of AC1(31−0) is compared to the content of AC2(31−0). The contents are equal (true), TC2 = !TC1 & 1.

```
Before                          After
AC1         80 0028 0400        AC1         80 0028 0400
AC2         00 0028 0400        AC2         00 0028 0400
M40                    0        M40                    0
TC1                    1        TC1                    1
TC2                    0        TC2                    0
```

CMPOR *Compare Accumulator, Auxiliary, or Temporary Register Content with OR*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = TCy | uns(src RELOP dst)	Yes	3	1	X
[2]	TCx = **!**TCy | uns(src RELOP dst)	Yes	3	1	X

Description These instructions perform a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU.

Status Bits

Affected by	C54CM, M40, TCy
Affects	TCx

See Also See the following other related instructions:

❑ Compare Accumulator, Auxiliary, or Temporary Register Content

❑ Compare Accumulator, Auxiliary, or Temporary Register Content with AND

❑ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum

❑ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum

❑ Compare Memory with Immediate Value

Compare Accumulator, Auxiliary, or Temporary Register Content with OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy | uns(src RELOP dst)				
[1a]	**TC1** = **TC2** | uns(src RELOP dst)	Yes	3	1	X
[1b]	**TC2** = **TC1** | uns(src RELOP dst)	Yes	3	1	X

Opcode

```
|0001 001E|FSSS cc10|FDDD 0utt
```

Operands dst, RELOP, src, TC1, TC2

Description This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits

Affected by	C54CM, M40, TCy
Affects	TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC2 = TC1 | uns(AC1 != AR1)	The unsigned content of AC1(15−0) is compared to the unsigned content of AR1. The contents are equal (false), TC2 = TC1 | 0.

```
Before                             After
AC1         00 8028 0400           AC1         00 8028 0400
AR1                 0400           AR1                 0400
TC1                    1           TC1                    1
TC2                    0           TC2                    1
```

Compare Accumulator, Auxiliary, or Temporary Register Content with OR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = **!**TCy | uns(src RELOP dst)				
[2a]	**TC1** = **!TC2** | uns(src RELOP dst)	Yes	3	1	X
[2b]	**TC2** = **!TC1** | uns(src RELOP dst)	Yes	3	1	X

Opcode

```
|0001  001E|FSSS  cc10|FDDD  1utt
```

Operands dst, RELOP, src, TC1, TC2

Description This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits	Affected by	C54CM, M40, TCy
	Affects	TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC2 = !TC1 | uns(AC1 != AR1)	The unsigned content of AC1(15−0) is compared to the unsigned content of AR1. The contents are equal (false), TC2 = !TC1 | 0.

```
Before                        After
AC1        00 8028 0400       AC1        00 8028 0400
AR1                0400       AR1                0400
TC1                   1       TC1                   1
TC2                   1       TC2                   0
```

MAX — *Compare Accumulator, Auxiliary, or Temporary Register Content Maximum*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = **max(**src, dst**)**	Yes	2	1	X

Opcode

```
| 0010  111E | FSSS  FDDD
```

Operands dst, src

Description This instruction performs a maximum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

❑ When the destination operand (dst) is an accumulator:

- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
- The operation is performed on 40 bits in the D-unit ALU:

  If M40 = 0, src(31–0) content is compared to dst(31–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(31–0) > dst(31–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
     else
  step3: CARRY = 1
  ```

  If M40 = 1, src(39–0) content is compared to dst(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(39–0) > dst(39–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
     else
  step3: CARRY = 1
  ```

- There is no overflow detection, overflow report, and saturation.

- When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

    The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

    ```
    step1: if (src(15–0) > dst(15–0))
    step2: dst = src
    ```
  - There is no overflow detection and saturation.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD
- The operation is performed on 40 bits in the D-unit ALU:

  The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(39–0) > AC1(39–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
         else
  step3: { CARRY = 1; dst(39–0) = AC1(39–0) }
  ```

There is no overflow detection, overflow report, and saturation.

Status Bits	Affected by	C54CM, M40, SXMD
	Affects	CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ❑ Compare and Select Accumulator Content Maximum
- ❑ Compare Memory with Immediate Value

Example 1

Syntax	Description
AC1 = max(AC2, AC1)	The content of AC2 is less than the content of AC1, the content of AC1 remains the same and the CARRY status bit is set to 1.

```
Before                                  After
AC2          00 0000 0000               AC2          00 0000 0000
AC1          00 8500 0000               AC1          00 8500 0000
SXMD                    1               SXMD                    1
M40                     0               M40                     0
CARRY                   0               CARRY                   1
```

Example 2

Syntax	Description
AC1 = max(AR1, AC1)	The content of AR1 is less than the content of AC1, the content of AC1 remains the same and the CARRY status bit is set to 1.

```
Before                                  After
AR1                  8020               AR1                  8020
AC1          00 0000 0040               AC1          00 0000 0040
CARRY                   0               CARRY                   1
```

Example 3

Syntax	Description
T1 = max(AC1, T1)	The content of AC1(15–0) is greater than the content of T1, the content of AC1(15−0) is stored in T1 and the CARRY status bit is cleared to 0.

```
Before                                  After
AC1          00 0000 8020               AC1          00 0000 8020
T1                   8010               T1                   8020
CARRY                   0               CARRY                   0
```

MIN *Compare Accumulator, Auxiliary, or Temporary Register Content Minimum*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = **min(**src, dst**)**	Yes	2	1	X

Opcode `0011 000E FSSS FDDD`

Operands dst, src

Description This instruction performs a minimum comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

❑ When the destination operand (dst) is an accumulator:

- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
- The operation is performed on 40 bits in the D-unit ALU:

  If M40 = 0, src(31–0) content is compared to dst(31–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(31–0) < dst(31–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
     else
  step3: CARRY = 1
  ```

  If M40 = 1, src(39–0) content is compared to dst(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(39–0) < dst(39–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
     else
  step3: CARRY = 1
  ```

- There is no overflow detection, overflow report, and saturation.

- When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

    The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

    ```
    step1: if (src(15–0) < dst(15–0))
    step2: dst = src
    ```
  - There is no overflow detection and saturation.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD

- The operation is performed on 40 bits in the D-unit ALU:

  The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

  ```
  step1: if (src(39–0) < AC1(39–0))
  step2: { CARRY = 0; dst(39–0) = src(39–0) }
     else
  step3: { CARRY = 1; dst(39–0) = AC1(39–0) }
  ```

There is no overflow detection, overflow report, and saturation.

Status Bits	Affected by	C54CM, M40, SXMD
	Affects	CARRY

Repeat This instruction can be repeated.

See Also

See the following other related instructions:

- Compare Accumulator, Auxiliary, or Temporary Register Content
- Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- Compare and Select Accumulator Content Minimum
- Compare Memory with Immediate Value

Example

Syntax	Description
T1 = min(AC1, T1)	The content of AC1(15–0) is greater than the content of T1, the content of T1 remains the same and the CARRY status bit is set to 1.

```
Before                          After
AC1        00 8000 0000         AC1        00 8000 0000
T1                 8020         T1                 8020
CARRY                 0         CARRY                 1
```

BCC *Compare and Branch*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**compare** (uns(src RELOP K8)) **goto** L8	No	4	7/6	X

† x/y cycles: x cycles = condition true, y cycles = condition false

Opcode

```
|0110  1111|FSSS  ccxu|KKKK  KKKK|LLLL  LLLL
```

Operands K8, L8, RELOP, src

Description This instruction performs a comparison operation between a source (src) register content and an 8-bit signed value, K8. The instruction performs a comparison in the D-unit ALU or in the A-unit ALU. The comparison is performed in the execute phase of the pipeline. If the result of the comparison is true, a branch occurs.

The program branch address is specified as an 8-bit signed offset, L8, relative to the program counter (PC). Use this instruction to branch within a 256-byte window centered on the current PC value.

The comparison depends on the optional uns keyword and, for accumulator comparisons, on M40.

- ❑ In the case of an unsigned comparison, the 8-bit constant, K8, is zero extended to:
  - ■ 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - ■ 40 bits, if the source (src) operand is an accumulator.
- ❑ In the case of a signed comparison, the 8-bit constant, K8, is sign extended to:
  - ■ 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - ■ 40 bits, if the source (src) operand is an accumulator.

As the following table shows, the uns keyword specifies an unsigned comparison; M40 defines the comparison bit width of the accumulator.

uns	src	Comparison Type
no	TAx	16-bit signed comparison in A-unit ALU
no	ACx	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	16-bit unsigned comparison in A-unit ALU
yes	ACx	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the conditions testing the accumulator contents are all performed as if M40 was set to 1.

Status Bits

Affected by	C54CM, M40
Affects	none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- Branch Conditionally
- Branch Unconditionally
- Branch on Auxiliary Register Not Zero

Example 1

Syntax	Description
compare (AC0 >= #12) goto branch	The signed content of AC0 is compared to the sign-extended 8-bit value (12). Because the content of AC0 is greater than or equal to 12, program control is passed to the program address label defined by branch (004078h).

```
          compare (AC0 >= #12) goto branch
          … …                                    address:  00 4075
          … …
branch    … …                                              00 4078
:
```

```
Before                           After
AC0         00 0000 3000         AC0         00 0000 3000
PC                004071         PC                004078
```

Example 2

Syntax	Description
compare (T1 != #1) goto branch	The content of T1 is not equal to 1, program control is passed to the next instruction (the branch is not taken).

```
         compare (T1 != #1) goto branch
         … …                              address:  00407D
         … …
branch   … …                                        004080
:
```

Before		**After**	
T1	0000	T1	0000
PC	4079	PC	407D

MAXDIFF *Compare and Select Accumulator Content Maximum*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**max_diff(**ACx, ACy, ACz, ACw**)**	Yes	3	1	X
[2]	**max_diff_dbl(**ACx, ACy, ACz, ACw, TRNx**)**	Yes	3	1	X

Description

Instruction [1] performs two paralleled 16-bit extremum selections in the D-unit ALU. Instruction [2] performs a single 40-bit extremum selection in the D-unit ALU.

Status Bits

Affected by	C54CM, M40, SATD
Affects	ACOVw, CARRY

See Also

See the following other related instructions:

- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- ❑ Compare and Select Accumulator Content Minimum

Compare and Select Accumulator Content Maximum

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**max_diff(**ACx, ACy, ACz, ACw**)**	Yes	3	1	X

Opcode

```
|0001 000E|DDSS 1100|SSDD nnnn
```

Operands ACw, ACx, ACy, ACz

Description This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual maximum search.

The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

For each datapath (high and low):

- ACx and ACy are the source accumulators.
- The differences are stored in accumulator ACw.
- The subtraction computation is equivalent to the dual 16-bit subtractions instruction.
- For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh (positive) and 8000h (negative).
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- The extremum is stored in accumulator ACz.
- The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

    If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> #1
TRN1 = TRN1 >> #1
ACw(39–16) = ACy(39–16) – ACx(39–16)
ACw(15–0) = ACy(15–0) – ACx(15–0)
If (ACx(31–16) > ACy(31–16))
   { bit(TRN0, 15) = #0 ; ACz(39–16) = ACx(39–16) }
else
   { bit(TRN0, 15) = #1 ; ACz(39–16) = ACy(39–16) }
if (ACx(15–0) > ACy(15–0))
   { bit(TRN1, 15) = #0 ; ACz(15–0) = ACx(15–0) }
else
   { bit(TRN1, 15) = #1 ; ACz(15–0) = ACy(15–0) }
```

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD
	Affects	ACOVw, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
max_diff(AC0, AC1, AC2, AC1)	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39−16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The maximum is stored in AC2. The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than AC1(31–16), AC0(39–16) is stored in AC2(39–16) and TRN0(15) is cleared to 0. AC0(15–0) is greater than AC1(15–0), AC0(15–0) is stored in AC2(15–0) and TRN1(15) is cleared to 0.

```
Before                              After
AC0         10 2400 2222            AC0         10 2400 2222
AC1         90 0000 0000            AC1         FF 8000 DDDE
AC2         00 0000 0000            AC2         10 2400 2222
SATD                   1            SATD                   1
TRN0                1000            TRN0                0800
TRN1                0100            TRN1                0080
ACOV1                  0            ACOV1                  1
CARRY                  1            CARRY                  0
```

Compare and Select Accumulator Content Maximum

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2a]	**max_diff_dbl(**ACx, ACy, ACz, ACw**, TRN0)**	Yes	3	1	X
[2b]	**max_diff_dbl(**ACx, ACy, ACz, ACw**, TRN1)**	Yes	3	1	X

Opcode

```
TRN0                 |0001  000E|DDSS  1101|SSDD  xxx0

TRN1                 |0001  000E|DDSS  1101|SSDD  xxx1
```

Operands ACw, ACx, ACy, ACz, TRNx

Description This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a maximum search.

❑ ACx and ACy are the two source accumulators.

❑ The difference between the source accumulators is stored in accumulator ACw.

❑ The subtraction computation is equivalent to the subtraction instruction.

❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

❑ The extremum between the source accumulators is stored in accumulator ACz.

❑ The extremum computation is similar to the compare register content maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.

❑ According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is cleared to 0; otherwise, it is set to 1.

If M40 = 0:

```
TRNx = TRNx >> #1
ACw(39–0) = ACy(39–0) – ACx(39–0)
If (ACx(31–0) > ACy(31–0))
   { bit(TRNx, 15) = #0 ; ACz(39–0) = ACx(39–0) }
else
   { bit(TRNx, 15) = #1 ; ACz(39–0) = ACy(39–0) }
```

If M40 = 1:

```
TRNx = TRNx >> #1
ACw(39–0) = ACy(39–0) – ACx(39–0)
If (ACx(39–0) > ACy(39–0))
   { bit(TRNx, 15) = #0 ; ACz(39–0) = ACx(39–0) }
else
   { bit(TRNx, 15) = #1 ; ACz(39–0) = ACy(39–0) }
```

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

Status Bits Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
max_diff_dbl(AC0, AC1, AC2, AC3, TRN1)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The maximum is stored in AC2. The content of TRN1 is shifted right 1 bit. AC0 is greater than AC1, AC0 is stored in AC2 and TRN1(15) is cleared to 0.

```
Before                         After
AC0        10 2400 2222        AC0        10 2400 2222
AC1        00 8000 DDDE        AC1        00 8000 DDDE
AC2        00 0000 0000        AC2        10 2400 2222
AC3        00 0000 0000        AC3        F0 5C00 BBBC
M40                   1        M40                   1
SATD                  1        SATD                  1
TRN1               0080        TRN1               0040
ACOV3                 0        ACOV3                 0
CARRY                 0        CARRY                 0
```

MINDIFF *Compare and Select Accumulator Content Minimum*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**min_diff(**ACx, ACy, ACz, ACw**)**	Yes	3	1	X
[2]	**min_diff_dbl(**ACx, ACy, ACz, ACw, TRNx**)**	Yes	3	1	X

Description Instruction [1] performs two paralleled 16-bit extremum selections in the D-unit ALU. Instruction [2] performs a single 40-bit extremum selection in the D-unit ALU.

Status Bits

Affected by	C54CM, M40, SATD
Affects	ACOVw, CARRY

See Also See the following other related instructions:

- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content
- ❑ Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- ❑ Compare and Select Accumulator Content Maximum

Compare and Select Accumulator Content Minimum

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**min_diff(**ACx, ACy, ACz, ACw**)**	Yes	3	1	X

Opcode `|0001 000E|DDSS 1110|SSDD xxxx`

Operands ACw, ACx, ACy, ACz

Description This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual minimum search.

The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

For each datapath (high and low):

- ACx and ACy are the source accumulators.
- The differences are stored in accumulator ACw.
- The subtraction computation is equivalent to the dual 16-bit subtractions instruction.
- For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh (positive) and 8000h (negative).
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- The extremum is stored in accumulator ACz.
- The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

    If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> #1
TRN1 = TRN1 >> #1
ACw(39–16) = ACy(39–16) – ACx(39–16)
ACw(15–0) = ACy(15–0) – ACx(15–0)
If (ACx(31–16) < ACy(31–16))
   { bit(TRN0, 15) = #0 ; ACz(39–16) = ACx(39–16) }
else
   { bit(TRN0, 15) = #1 ; ACz(39–16) = ACy(39–16) }
if (ACx(15–0) < ACy(15–0))
   { bit(TRN1, 15) = #0 ; ACz(15–0) = ACx(15–0) }
else
   { bit(TRN1, 15) = #1 ; ACz(15–0) = ACy(15–0) }
```

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD
	Affects	ACOVw, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
min_diff(AC0, AC1, AC2, AC1)	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39−16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The minimum is stored in AC2 (sign bit extracted at bits 31 and 15). The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than or equal to AC1(31–16), AC1(39–16) is stored in AC2(39–16) and TRN0(15) is set to 1. AC0(15–0) is greater than or equal to AC1(15–0), AC1(15–0) is stored in AC2(15–0) and TRN1(15) is set to 1.

```
Before                          After
AC0          10 2400 2222       AC0          10 2400 2222
AC1          00 8000 DDDE       AC1          FF 8000 BBBC
AC2          10 2400 2222       AC2          00 8000 DDDE
SATD                    1       SATD                    1
TRN0                 0800       TRN0                 8400
TRN1                 0040       TRN1                 8020
ACOV1                   0       ACOV1                   1
CARRY                   0       CARRY                   1
```

Compare and Select Accumulator Content Minimum

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2a]	**min_diff_dbl(**ACx, ACy, ACz, ACw**, TRN0)**	Yes	3	1	X
[2b]	**min_diff_dbl(**ACx, ACy, ACz, ACw**, TRN1)**	Yes	3	1	X

Opcode

```
TRN0        |0001 000E|DDSS 1111|SSDD xxx0
TRN1        |0001 000E|DDSS 1111|SSDD xxx1
```

Operands ACw, ACx, ACy, ACz, TRNx

Description This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a minimum search.

- ACx and ACy are the two source accumulators.
- The difference between the source accumulators is stored in accumulator ACw.
- The subtraction computation is equivalent to the subtraction instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.
- The extremum between the source accumulators is stored in accumulator ACz.
- The extremum computation is similar to the compare register content maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is cleared to 0; otherwise, it is set to 1.

If M40 = 0:

```
TRNx = TRNx >> #1
ACw(39–0) = ACy(39–0) – ACx(39–0)
If (ACx(31–0) < ACy(31–0))
   { bit(TRNx, 15) = #0 ; ACz(39–0) = ACx(39–0) }
else
   { bit(TRNx, 15) = #1 ; ACz(39–0) = ACy(39–0) }
```

If M40 = 1:

```
TRNx = TRNx >> #1
ACw(39–0) = ACy(39–0) – ACx(39–0)
If (ACx(39–0) < ACy(39–0))
   { bit(TRNx, 15) = #0 ; ACz(39–0) = ACx(39–0) }
else
   { bit(TRNx, 15) = #1 ; ACz(39–0) = ACy(39–0) }
```

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

Status Bits Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
min_diff_dbl(AC0, AC1, AC2, AC3, TRN0)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The minimum is stored in AC2. The content of TRN0 is shifted right 1 bit. If AC0 is less than AC1, AC0 is stored in AC2 and TRN0(15) is cleared to 0; otherwise, AC1 is stored in AC2 and TRN0(15) is set to 1.

CMP *Compare Memory with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = **(**Smem == K16**)**	No	4	1	X
[2]	**TC2** = **(**Smem == K16**)**	No	4	1	X

Opcode

```
TC1        |1111  0000|AAAA  AAAI|KKKK  KKKK|KKKK  KKKK
TC2        |1111  0001|AAAA  AAAI|KKKK  KKKK|KKKK  KKKK
```

Operands K16, Smem, TCx

Description This instruction performs a comparison in the A-unit ALU. The data memory operand Smem is compared to the 16-bit signed constant, K16. If they are equal, the TCx status bit is set to 1; otherwise, it is cleared to 0.

```
if((Smem) == K16)
   TCx = 1
else
   TCx = 0
```

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Compare Accumulator, Auxiliary, or Temporary Register Content

Example 1

Syntax	Description
TC1 = (*AR1+ == #400h)	The content addressed by AR1 is compared to the signed 16-bit value (400h). Because they are equal, TC1 is set to 1. AR1 is incremented by 1.

```
Before                    After
AR1          0285         AR1          0286
0285         0400         0285         0400
TC1             0         TC1             1
```

Example 2

Syntax	Description
TC2 = (*AR1 == #400h)	The content addressed by AR1 is compared to the signed 16-bit value (400h). Because they are not equal, TC2 is cleared to 0.

Before		**After**	
AR1	0285	AR1	0285
0285	0000	0285	0000
TC2	0	TC2	0

BNOT *Complement Accumulator, Auxiliary, or Temporary Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**cbit(**src, Baddr**)**	No	3	1	X

Opcode | `1110 1100 | AAAA AAAI | FSSS 011x`

Operands Baddr, src

Description This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction complements a single bit, as defined by the bit addressing mode, Baddr, of the source register.

The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Clear Accumulator, Auxiliary, or Temporary Register Bit
- Complement Accumulator, Auxiliary, or Temporary Register Content
- Complement Memory Bit
- Set Accumulator, Auxiliary, or Temporary Register Bit

Example

Syntax	Description
cbit(T0, AR1)	The bit at the position defined by the content of AR1(3−0) in T0 is complemented.

```
Before                    After
T0           E000         T0           F000
AR1          000C         AR1          000C
```

NOT *Complement Accumulator, Auxiliary, or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = ~src	Yes	2	1	X

Opcode `| 0011 011E | FSSS FDDD`

Operands dst, src

Description This instruction computes the 1s complement (bitwise complement) of the content of the source register (src).

- ❑ When the destination (dst) operand is an accumulator:
  - ■ The bit inversion is performed on 40 bits in the D-unit ALU and the result is stored in the destination accumulator.
  - ■ If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - ■ The bit inversion is performed on 16 bits in the A-unit ALU and the result is stored in the destination auxiliary or temporary register.
  - ■ If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Complement Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Negate Accumulator, Auxiliary, or Temporary Register Content

Example

Syntax	Description
AC1 = ~AC0	The content of AC0 is complemented and the result is stored in AC1.

```
Before                             After
AC0         7E 2355 4FC0           AC0         7E 2355 4FC0
AC1         00 2300 5678           AC1         81 DCAA B03F
```

BNOT *Complement Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**cbit(**Smem, src**)**	No	3	1	X

Opcode `1110 0011 | AAAA AAAI | FSSS 111x`

Operands Smem, src

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction complements a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❏ Clear Memory Bit
- ❏ Complement Accumulator, Auxiliary, or Temporary Register Bit
- ❏ Complement Accumulator, Auxiliary, or Temporary Register Content
- ❏ Set Memory Bit

Example

Syntax	Description
cbit(*AR3, AC0)	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is complemented.

EXP — *Compute Exponent of Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Tx = **exp(**ACx**)**	Yes	3	1	X

Opcode

```
|0001 000E|xxSS 1000|xxdd xxxx
```

Operands ACx, Tx

Description This instruction computes the exponent of the source accumulator ACx in the D-unit shifter. The result of the operation is stored in the temporary register Tx. The A-unit ALU is used to make the move operation.

This exponent is a signed 2s-complement value in the –8 to 31 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting 8 from this value. The number of leading bits is the number of shifts to the MSBs needed to align the accumulator content on a signed 40-bit representation.

ACx is not modified after the execution of this instruction. If ACx is equal to 0, Tx is loaded with 0.

This instruction produces in Tx the opposite result than computed by the Compute Mantissa and Exponent of Accumulator Content instruction (page 5-132).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Compute Mantissa and Exponent of Accumulator Content

Example

Syntax	Description
T1 = exp(AC0)	The exponent is computed by subtracting 8 from the number of leading bits in the content of AC0. The exponent value is a signed 2s-complement value in the –8 to 31 range and is stored in T1.

```
Before                          After
AC0         FF FFFF FFCB        AC0         FF FFFF FFCB
T1                  0000        T1                  0019
```

MANT::NEXP *Compute Mantissa and Exponent of Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **mant(**ACx**),** Tx = **−exp(**ACx**)**	Yes	3	1	X2

Opcode

```
|0001 000E|DDSS 1001|xxdd xxxx
```

Operands ACx, ACy, Tx

Description This instruction computes the exponent and mantissa of the source accumulator ACx. The computation of the exponent and the mantissa is executed in the D-unit shifter. The exponent is computed and stored in the temporary register Tx. The A-unit is used to make the move operation. The mantissa is stored in the accumulator ACy.

The exponent is a signed 2s-complement value in the –31 to 8 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting this value from 8. The number of leading bits is the number of shifts to the MSBs needed to align the accumulator content on a signed 40-bit representation.

The mantissa is obtained by aligning the ACx content on a signed 32-bit representation. The mantissa is computed and stored in ACy.

- ❑ The shift operation is performed on 40 bits.
  - ■ When shifting to the LSBs, bit 39 of ACx is extended to bit 31.
  - ■ When shifting to the MSBs, 0 is inserted at bit position 0.
- ❑ If ACx is equal to 0, Tx is loaded with 8000h.

This instruction produces in Tx the opposite result than computed by the Compute Exponent of Accumulator Content instruction (page 5-131).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Compute Exponent of Accumulator Content

Example 1

Syntax	Description
AC1 = mant(AC0), T1 = −exp(AC0)	The exponent is computed by subtracting the number of leading bits in the content of AC0 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T1. The mantissa is computed by aligning the content of AC0 on a signed 32-bit representation. The mantissa value is stored in AC1.

```
Before                          After
AC0         21 0A0A 0A0A        AC0         21 0A0A 0A0A
AC1         FF FFFF F001        AC1         00 4214 1414
T1                  0000        T1                  0007
```

Example 2

Syntax	Description
AC1 = mant(AC0), T1 = −exp(AC0)	The exponent is computed by subtracting the number of leading bits in the content of AC0 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T1. The mantissa is computed by aligning the content of AC0 on a signed 32-bit representation. The mantissa value is stored in AC1.

```
Before                          After
AC0         00 E804 0000        AC0         00 E804 0000
AC1         FF FFFF F001        AC1         00 7402 0000
T1                  0000        T1                  0001
```

BCNT *Count Accumulator Bits*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Tx = **count(**ACx, ACy**, TC1)**	Yes	3	1	X
[2]	Tx = **count(**ACx, ACy**, TC2)**	Yes	3	1	X

Opcode

```
TC1        |0001  000E|xxSS  1010|SSdd  xxx0
TC2        |0001  000E|XXSS  1010|SSdd  xxx1
```

Operands ACx, ACy, Tx, TCx

Description This instruction performs bit field manipulation in the D-unit shifter. The result is stored in the selected temporary register (Tx). The A-unit ALU is used to make the move operation.

Accumulator ACx is ANDed with accumulator ACy. The number of bits set to 1 in the intermediary result is evaluated and stored in the selected temporary register (Tx). If the number of bits is even, the selected TCx status bit is cleared to 0. If the number of bits is odd, the selected TCx status bit is set to 1.

Status Bits Affected by none

Affects TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
T1 = count(AC1, AC2, TC1)	The content of AC1 is ANDed with the content of AC2, the number of bits set to 1 in the result is evaluated and stored in T1. The number of bits set to 1 is odd, TC1 is set to 1.

```
Before                             After
AC1          7E 2355 4FC0          AC1          7E 2355 4FC0
AC2          0F E340 5678          AC2          0F E340 5678
T1                   0000          T1                   000B
TC1                     0          TC1                     1
```

ADD *Dual 16-Bit Additions*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACy**)** = **HI(**Lmem**)** + **HI(**ACx**),** **LO(**ACy**)** = **LO(**Lmem**)** + **LO(**ACx**)**	No	3	1	X
[2]	**HI(**ACx**)** = **HI(**Lmem**)** + Tx, **LO(**ACx**)** = **LO(**Lmem**)** + Tx	No	3	1	X

Description These instructions perform two paralleled addition operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits

Affected by	C54CM, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also See the following other related instructions:

❑ Addition

❑ Addition or Subtraction Conditionally

❑ Addition or Subtraction Conditionally with Shift

❑ Addition with Parallel Store Accumulator Content to Memory

❑ Addition, Subtraction, or Move Accumulator Content Conditionally

❑ Dual 16-Bit Addition and Subtraction

❑ Dual 16-Bit Subtraction and Addition

Dual 16-Bit Additions

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACy**)** = **HI(**Lmem**)** + **HI(**ACx**),** **LO(**ACy**)** = **LO(**Lmem**)** + **LO(**ACx**)**	No	3	1	X

Opcode `|1110 1110|AAAA AAAI|SSDD 000x`

Operands ACx, ACy, Lmem

Description This instruction performs two paralleled addition operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:

- the lower part is used as one of the 16-bit operands of the ALU low part
- the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part

❑ The data memory operand dbl(Lmem) addresses are aligned:

- if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
- if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1

❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.

- For the operations performed in the ALU low part, overflow is detected at bit position 15.
- For the operations performed in the ALU high part, overflow is detected at bit position 31.

❑ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ❑ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ❑ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
HI(AC0) = HI(*AR3) + HI(AC1), LO(AC0) = LO(*AR3) + LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is added to the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

Dual 16-Bit Additions

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**HI(**ACx**)** = **HI(**Lmem**)** + Tx, **LO(**ACx**)** = **LO(**Lmem**)** + Tx	No	3	1	X

Opcode

```
|1110 1110|AAAA AAAI|ssDD 100x
```

Operands ACx, Lmem, Tx

Description This instruction performs two paralleled addition operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ❑ The temporary register Tx:
  - ■ is used as one of the 16-bit operands of the ALU low part
  - ■ is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - ■ the lower part is used as one of the 16-bit operands of the ALU low part
  - ■ the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ❑ The data memory operand dbl(Lmem) addresses are aligned:
  - ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - ■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- ❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - ■ For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - ■ For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(AC0) = HI(*AR3) + T0, LO(AC0) = LO(*AR3) + T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

ADDSUB *Dual 16-Bit Addition and Subtraction*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACx**)** = Smem + Tx, **LO(**ACx**)** = Smem – Tx	No	3	1	X
[2]	**HI(**ACx**)** = **HI(**Lmem**)** + Tx, **LO(**ACx**)** = **LO(**Lmem**)** – Tx	No	3	1	X

Description

These instructions perform two paralleled addition and subtraction operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits

Affected by	C54CM, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also

See the following other related instructions:

❑ Addition

❑ Dual 16-Bit Additions

❑ Dual 16-Bit Subtractions

❑ Dual 16-Bit Subtraction and Addition

❑ Subtraction

Dual 16-Bit Addition and Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACx**)** = Smem + Tx, **LO(**ACx**)** = Smem – Tx	No	3	1	X

Opcode `1101 1110 | AAAA AAAI | ssDD 1000`

Operands ACx, Smem, Tx

Description This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(AC1) = *AR1 + T1, LO(AC1) = *AR1 – T1	Both instructions are performed in parallel. The content addressed by AR1 is added to the content of T1 and the result is stored in AC1(39–16). The duplicated content of T1 is subtracted from the duplicated content addressed by AR1 and the result is stored in AC1(15–0).

```
Before                          After
AC1         00 2300 0000        AC1         00 2300 A300
T1                  4000        T1                  4000
AR1                 0201        AR1                 0201
201                 E300        201                 E300
SXMD                   1        SXMD                   1
M40                    1        M40                    1
ACOV0                  0        ACOV0                  0
CARRY                  0        CARRY                  1
```

Dual 16-Bit Addition and Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**HI(**ACx**)** = **HI(**Lmem**)** + Tx, **LO(**ACx**)** = **LO(**Lmem**)** – Tx	No	3	1	X

Opcode

```
|1110 1110|AAAA AAAI|ssDD 110x
```

Operands ACx, Lmem, Tx

Description This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(AC0) = HI(*AR3) + T0, LO(AC0) = LO(*AR3) – T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

SUB *Dual 16-Bit Subtractions*

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	**HI(**ACy**)** = **HI(**ACx**)** – **HI(**Lmem**),** **LO(**ACy**)** = **LO(**ACx**)** – **LO(**Lmem**)**	No	3	1	X
[2]	**HI(**ACy**)** = **HI(**Lmem**)** – **HI(**ACx**),** **LO(**ACy**)** = **LO(**Lmem**)** – **LO(**ACx**)**	No	3	1	X
[3]	**HI(**ACx**)** = Tx – **HI(**Lmem**),** **LO(**ACx**)** = Tx – **LO(**Lmem**)**	No	3	1	X
[4]	**HI(**ACx**)** = **HI(**Lmem**)** – Tx, **LO(**ACx**)** = **LO(**Lmem**)** – Tx	No	3	1	X

Description

These instructions perform two paralleled subtraction operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits

Affected by C54CM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

See Also

See the following other related instructions:

❑ Addition or Subtraction Conditionally

❑ Addition or Subtraction Conditionally with Shift

❑ Addition, Subtraction, or Move Accumulator Content Conditionally

❑ Dual 16-Bit Addition and Subtraction

❑ Dual 16-Bit Subtraction and Addition

❑ Subtract Conditionally

❑ Subtraction

❑ Subtraction with Parallel Store Accumulator Content to Memory

Dual 16-Bit Subtractions

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACy**)** = **HI(**ACx**)** – **HI(**Lmem**),** **LO(**ACy**)** = **LO(**ACx**)** – **LO(**Lmem**)**	No	3	1	X

Opcode `| 1110 1110 | AAAA AAAI | SSDD 001x`

Operands ACx, ACy, Lmem

Description This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path).

- ❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ❑ The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- ❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- ❑ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ❑ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ❑ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
HI(AC0) = HI(AC1) – HI(*AR3), LO(AC0) = LO(AC1) – LO(*AR3)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 (sign extended to 24 bits) is subtracted from the content of AC1(39–16) and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the content of AC1(15–0) and the result is stored in AC0(15–0).

Dual 16-Bit Subtractions

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[2]	**HI(**ACy**)** = **HI(**Lmem**)** – **HI(**ACx**),** **LO(**ACy**)** = **LO(**Lmem**)** – **LO(**ACx**)**	No	3	1	X

Opcode `| 1110 1110 | AAAA AAAI | SSDD 010x`

Operands ACx, ACy, Lmem

Description This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:

  ■ the lower part is used as one of the 16-bit operands of the ALU low part

  ■ the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part

❑ The data memory operand dbl(Lmem) addresses are aligned:

  ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1

  ■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1

❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVy) is set.

  ■ For the operations performed in the ALU low part, overflow is detected at bit position 15.

  ■ For the operations performed in the ALU high part, overflow is detected at bit position 31.

❑ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ❑ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ❑ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits	Affected by	C 16, C54CM, SATD, SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
HI(AC0) = HI(*AR3) – HI(AC1), LO(AC0) = LO(*AR3) – LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

Dual 16-Bit Subtractions

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[3]	**HI(**ACx**)** = Tx – **HI(**Lmem**),** **LO(**ACx**)** = Tx – **LO(**Lmem**)**	No	3	1	X

Opcode `| 1110 1110 | AAAA AAAI | ssDD 011x`

Operands ACx, Lmem, Tx

Description This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

❑ The temporary register Tx:

  ■ is used as one of the 16-bit operands of the ALU low part

  ■ is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part

❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:

  ■ the lower part is used as one of the 16-bit operands of the ALU low part

  ■ the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part

❑ The data memory operand dbl(Lmem) addresses are aligned:

  ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1

  ■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1

❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.

  ■ For the operations performed in the ALU low part, overflow is detected at bit position 15.

  ■ For the operations performed in the ALU high part, overflow is detected at bit position 31.

- ❑ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(AC0) = T0 – HI(*AR3), LO(AC0) = T0 – LO(*AR3)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 is subtracted from the content of T0 and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the duplicated content of T0 and the result is stored in AC0(15–0).

Dual 16-Bit Subtractions

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[4]	**HI(**ACx**)** = **HI(**Lmem**)** – Tx, **LO(**ACx**)** = **LO(**Lmem**)** – Tx	No	3	1	X

Opcode `1110 1110 | AAAA AAAI | ssDD 101x`

Operands ACx, Tx, Lmem

Description This instruction performs two paralleled subtraction operations in one cycle. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- ❑ The temporary register Tx:
  - ■ is used as one of the 16-bit operands of the ALU low part
  - ■ is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- ❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - ■ the lower part is used as one of the 16-bit operands of the ALU low part
  - ■ the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- ❑ The data memory operand dbl(Lmem) addresses are aligned:
  - ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - ■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- ❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - ■ For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - ■ For the operations performed in the ALU high part, overflow is detected at bit position 31.

- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits Affected by C16, C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
HI(AC0) = HI(*AR3) – T0, LO(AC0) = LO(*AR3) – T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

SUBADD *Dual 16-Bit Subtraction and Addition*

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	**HI(**ACx**)** = Smem – Tx, **LO(**ACx**)** = Smem + Tx	No	3	1	X
[2]	**HI(**ACx**)** = **HI(**Lmem**)** – Tx, **LO(**ACx**)** = **LO(**Lmem**)** + Tx	No	3	1	X

Description

These instructions perform two paralleled subtraction and addition operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits

Affected by	C54CM, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also

See the following other related instructions:

❑ Addition

❑ Dual 16-Bit Additions

❑ Dual 16-Bit Addition and Subtraction

❑ Dual 16-Bit Subtractions

❑ Subtraction

Dual 16-Bit Subtraction and Addition

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	**HI(**ACx**)** = Smem – Tx, **LO(**ACx**)** = Smem + Tx	No	3	1	X

Opcode `1101 1110 | AAAA AAAI | ssDD 1001`

Operands ACx, Smem, Tx

Description This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

- The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
  - For the operations performed in the ALU low part, overflow is detected at bit position 15.
  - For the operations performed in the ALU high part, overflow is detected at bit position 31.
- For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.

- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits Affected by C54CM, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
HI(AC0) = *AR3 – T0, LO(AC0) = *AR3 + T0	Both instructions are performed in parallel. The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the duplicated content addressed by AR3 and the result is stored in AC0(15–0).

Dual 16-Bit Subtraction and Addition

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[2]	**HI(**ACx**)** = **HI(**Lmem**)** – Tx, **LO(**ACx**)** = **LO(**Lmem**)** + Tx	No	3	1	X

Opcode

```
|1110 1110|AAAA AAAI|ssDD 111x
```

Operands ACx, Lmem, Tx

Description This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

❑ The temporary register Tx:
- is used as one of the 16-bit operands of the ALU low part
- is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part

❑ The data memory operand dbl(Lmem) is divided into two 16-bit parts:
- the lower part is used as one of the 16-bit operands of the ALU low part
- the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part

❑ The data memory operand dbl(Lmem) addresses are aligned:
- if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
- if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1

❑ For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVx) is set.
- For the operations performed in the ALU low part, overflow is detected at bit position 15.
- For the operations performed in the ALU high part, overflow is detected at bit position 31.

- ❑ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- ❑ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - ■ For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - ■ For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

- ❑ When C54CM = 1 and C16 = 1, the instruction behaves like a dual 16-bit instruction and the carry is not propagated at bit 15 in the D-unit ALU.
- ❑ When C54CM = 1 and C16 = 0, the instruction behaves like a single arithmetic instruction and the carry is propagated at bit 15 in the D-unit ALU.

Status Bits	Affected by	C16, C54CM, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(AC0) = HI(*AR3) – T0, LO(AC0) = LO(*AR3) + T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

XCC *Execute Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**if (**cond**) execute(AD_Unit)**	No	2	1	AD
[2]	**if (**cond**) execute(D_Unit)**	No	2	1	X

Description

These instructions evaluate a single condition defined by the cond field and allow you to control execution of all operations implied by the instruction or part of the instruction. See Table 1−3 for a list of conditions.

Instruction [1] allows you to control the entire execution flow from the address phase to the execute phase of the pipeline. Instruction [2] allows you to only control the execution flow from the execute phase of the pipeline. The use of a label, where control of the execute conditionally instruction ends, is optional.

- ❑ These instructions may be executed alone.
- ❑ These instructions may be executed with two paralleled instructions.
- ❑ These instructions may be executed with the instruction with which it is paralleled.
- ❑ These instructions may be executed with the previous instruction.
- ❑ These instructions may be executed with the previous instruction and two paralleled instructions.
- ❑ These instructions cannot be repeated.
- ❑ These instructions cannot be used as the last instruction in a repeat loop structure.
- ❑ These instructions cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

Execute Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**if (**cond**) execute(AD_Unit)**	No	2	1	AD

Opcode

```
|1001  0110|0CCC  CCCC
|1001  1110|0CCC  CCCC
|1001  1111|0CCC  CCCC
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands cond

Description This instruction evaluates a single condition defined by the cond field and allows you to control the execution flow of an instruction, or instructions, from the address phase to the execute phase of the pipeline. See Table 1−3 for a list of conditions.

When this instruction moves into the address phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 3-cycle latency for the condition testing.

❑ This instruction may be executed alone:

```
      if(cond) execute(AD_unit)
      instruction_executes_conditionally
label:
```

❑ This instruction may be executed with two paralleled instructions:

```
      if(cond) execute(AD_unit)
      instruction_1_executes_conditionally
      || instruction_2_executes_conditionally
label:
```

❑ This instruction may be executed with the instruction with which it is paralleled:

```
      if(cond) execute(AD_unit)
      || instruction_executes_conditionally
label:
```

- This instruction may be executed with a previous instruction:

```
          previous_instruction
          || if(cond) execute(AD_unit)
          instruction_executes_conditionally
label:
```

- This instruction may be executed with a previous instruction and two paralleled instructions:

```
          previous_instruction
          || if(cond) execute(AD_unit)
          instruction_1_executes_conditionally
          || instruction_2_executes_conditionally
label:
```

This instruction cannot be used as the last instruction in a repeat loop structure.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

Repeat This instruction cannot be repeated.

Example 1

Syntax	Description

if (TC1) execute(AD_unit)
mar(*AR1+)
AC1 = AC1 + *AR1
	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1). The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.

```
Before                         After
AC1          00 0000 4300      AC1          00 0000 6321
TC1                     1      TC1                     1
CARRY                   1      CARRY                   0
AR1                  0200      AR1                  0201
200                  2020      200                  2020
201                  2021      201                  2021
```

Example 2

Syntax	Description

if (TC1) execute(AD_unit)
mar(*AR1+)
AC1 = AC1 + *AR1
	TC1 is not equal to 1, the next instruction is not executed (AR1 is not incremented). The content of AC1 is added to the content addressed by AR1 (2020h) and the result is stored in AC1.

```
Before                            After
AC1          00 0000 4300         AC1          00 0000 6320
TC1                     0         TC1                     0
CARRY                   1         CARRY                   0
AR1                  0200         AR1                  0200
200                  2020         200                  2020
201                  2021         201                  2021
```

Execute Conditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**if (**cond**) execute(D_Unit)**	No	2	1	X

Opcode

```
|1001  0110|1CCC  CCCC
|1001  1110|1CCC  CCCC
|1001  1111|1CCC  CCCC
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands cond

Description This instruction evaluates a single condition defined by the cond field and allows you to control the execution flow of an instruction, or instructions, from the execute phase of the pipeline. This instruction differs from instruction [1] because in this instruction operations performed in the address phase are always executed. See Table 1−3 for a list of conditions.

When this instruction moves into the execute phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 0-cycle latency for the condition testing.

❑ This instruction may be executed alone:

```
    if(cond) execute(D_unit)
    instruction_executes_conditionally
label:
```

❑ This instruction may be executed with two paralleled instructions:

```
    if(cond) execute(D_unit)
    instruction_1_executes_conditionally
    || instruction_2_executes_conditionally
label:
```

❑ This instruction may be executed with the instruction with which it is paralleled. When this instruction syntax is used and the instruction to be executed conditionally is a store-to-memory instruction, there is a 1-cycle latency for the condition setting.

```
    if(cond) execute(D_unit)
    || instruction_executes_conditionally
label:
```

❑ This instruction may be executed with a previous instruction:

```
       previous_instruction
       || if(cond) execute(D_unit)
       instruction_executes_conditionally
label:
```

❑ This instruction may be executed with a previous instruction and two paralleled instructions:

```
       previous_instruction
       || if(cond) execute(D_unit)
       instruction_1_executes_conditionally
       || instruction_2_executes_conditionally
label:
```

This instruction cannot be used as the last instruction in a repeat loop structure.

When the instruction to be executed conditionally is an instruction to read data from memory, the data read operation is performed regardless of the condition and the read data is discarded at the execute phase if the condition is false.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

and an instruction to read data from I/O space.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

Repeat This instruction cannot be repeated.

Example 1

Syntax	Description

if (TC1) execute(D_unit)
mar(*AR1+)
AC1 = AC1 + *AR1
	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1). The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.

```
Before                             After
AC1          00 0000 4300          AC1          00 0000 6321
TC1                     1          TC1                     1
CARRY                   1          CARRY                   0
AR1                  0200          AR1                  0201
200                  2020          200                  2020
201                  2021          201                  2021
```

Example 2

Syntax	Description

if (TC1) execute(D_unit)
mar(*AR1+)
AC1 = AC1 + *AR1
	TC1 is not equal to 1, the next instruction would not be executed; however, since the next instruction is a pointer modification, AR1 is incremented by 1 in the address phase. The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.

```
Before                             After
AC1          00 0000 4300          AC1          00 0000 6321
TC1                     0          TC1                     0
CARRY                   1          CARRY                   0
AR1                  0200          AR1                  0201
200                  2020          200                  2020
201                  2021          201                  2021
```

BFXPA *Expand Accumulator Bit Field*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = **field_expand(**ACx, k16**)**	No	4	1	X

Opcode | `0111 0110` | `kkkk kkkk` | `kkkk kkkk` | `FDDD 01SS`

Operands ACx, dst, k16

Description This instruction performs a bit field manipulation in the D-unit shifter. When the destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus carries the output of the D-unit shifter directly into dst.

The 16-bit field mask, k16, is scanned from the least significant bits (LSBs) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the 16 LSBs of the source accumulator (ACx) bits are extracted and separated with 0 toward the MSBs. The result is stored in the dst.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Extract Accumulator Bit Field

Example

Syntax	Description
T2 = field_expand(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the bit in AC0 is extracted and separated with 0 toward the MSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

```
Execution
#k16 (8024h)          1000 0000 0010 0100
AC0(15–0)             0010 1011 0110 0101
T2                    1000 0000 0000 0100
```

```
Before                          After
AC0         00 2300 2B65        AC0         00 2300 2B65
T2                  0000        T2                  8004
```

BFXTR *Extract Accumulator Bit Field*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = **field_extract(**ACx, k16**)**	No	4	1	X

Opcode `| 0111 0110 | kkkk kkkk | kkkk kkkk | FDDD 00SS`

Operands ACx, dst, k16

Description This instruction performs a bit field manipulation in the D-unit shifter. When the destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus carries the output of the D-unit shifter directly into dst.

The 16-bit field mask, k16, is scanned from the least significant bits (LSBs) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the corresponding 16 LSBs of the source accumulator (ACx) bits are extracted and packed toward the LSBs. The result is stored in the dst.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Expand Accumulator Bit Field

Example

Syntax	Description
T2 = field_extract(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the corresponding bit in AC0 is extracted and packed toward the LSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

```
Execution
#k16 (8024h)          1000 0000 0010 0100
AC0(15–0)             0101 0101 1010 1010
T2                    0000 0000 0000 0010
```

```
Before                         After
AC0         00 2300 55AA       AC0         00 2300 55AA
T2                  0000       T2                  0002
```

FIRSSUB *Finite Impulse Response Filter, Antisymmetrical*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**firsn(**Xmem, Ymem, **coef(**Cmem**)**, ACx, ACy**)**	No	4	1	X

Opcode

```
|1000 0101|XXXM MMYY|YMMM 11mm|DDx1 DDU%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations: multiply and accumulate (MAC), and subtraction. The firsn() operation is executed:

```
ACy = ACy + (ACx * Cmem),
ACx = (Xmem << #16) - (Ymem << #16)
```

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of ACx(32−16) and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.

- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by C54CM, FRCT, M40, SATD, SMUL, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- Finite Impulse Response Filter, Symmetrical

Example

Syntax	Description
firsn(*AR0, *AR1, coef(*CDP), AC0, AC1)	The content of AC0(32–16) multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0.

```
Before                         After
AC0        00 6900 0000        AC0        00 4500 0000
AC1        00 0023 0000        AC1        FF D8ED 3F00
*AR0               3400        *AR0               3400
*AR1               EF00        *AR1               EF00
*CDP               A067        *CDP               A067
ACOV0                 0        ACOV0                 0
ACOV1                 0        ACOV1                 0
CARRY                 0        CARRY                 0
FRCT                  0        FRCT                  0
SXMD                  0        SXMD                  0
```

FIRSADD *Finite Impulse Response Filter, Symmetrical*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**firs(**Xmem, Ymem, **coef(**Cmem**)**, ACx, ACy**)**	No	4	1	X

Opcode

```
|1000  0101|XXXM  MMYY|YMMM  11mm|DDx0  DDU%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations: multiply and accumulate (MAC), and addition. The firs() operation is executed:

```
ACy = ACy + (ACx * Cmem),
ACx = (Xmem << #16) + (Ymem << #16)
```

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of ACx(32−16) and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

❑ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.

❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.

❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The second operation performs an addition operation between the content of data memory operand Xmem, shifted left 16 bits, and the content of data memory operand Ymem, shifted left 16 bits.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by	C54CM, FRCT, M40, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Finite Impulse Response Filter, Antisymmetrical

Example

Syntax	Description
firs(*AR0, *AR1, coef(*CDP), AC0, AC1)	The content of AC0(32–16) multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC1 and the result is stored in AC1. The content addressed by AR0 shifted left by 16 bits is added to the content addressed by AR1 shifted left by 16 bits and the result is stored in AC0.

```
Before                      After
AC0        00 6900 0000     AC0        00 2300 0000
AC1        00 0023 0000     AC1        FF D8ED 3F00
*AR0               3400     *AR0               3400
*AR1               EF00     *AR1               EF00
*CDP               A067     *CDP               A067
ACOV0                 0     ACOV0                 0
ACOV1                 0     ACOV1                 0
CARRY                 0     CARRY                 1
FRCT                  0     FRCT                  0
SXMD                  0     SXMD                  0
```

IDLE *Idle*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**idle**	No	4	?	D

Opcode `0111 1010 | xxxx xxxx | xxxx xxxx | xxxx 110x`

Operands none

Description This instruction forces the program being executed to wait until an interrupt or a reset occurs. The power-down mode that the processor operates in depends on a configuration register accessible through the peripheral access mechanism.

Status Bits

Affected by	INTM
Affects	none

Repeat This instruction cannot be repeated.

LMS *Least Mean Square*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**lms(**Xmem, Ymem, ACx, ACy**)**	No	4	1	X

Opcode

```
|1000 0110|XXXM MMYY|YMMM DDDD|110x xxx%
```

Operands ACx, ACy, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC), and addition. The instruction is executed:

```
ACy = ACy + (Xmem * Ymem),
ACx = rnd(ACx + (Xmem << #16))
```

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

The second operation performs an addition between an accumulator content and the content of data memory operand Xmem shifted left by 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40.
- Rounding is performed according to RDM.

- When an overflow is detected on the result of the rounding, the accumulator is saturated according to SATD. Note that no overflow detection is performed on the intermediate result after the addition but before the rounding.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the 16 lowest bits of ACx. The addition operation has no overflow detection, report, and saturation after the shifting operation.

Status Bits

Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
lms(*AR0, *AR1, AC0, AC1)	The content addressed by AR0 multiplied by the content addressed by AR1 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR0 shifted left by 16 bits is added to the content of AC0. The result is rounded and stored in AC0.

```
Before                        After
AC0         00 1111 2222      AC0         00 2111 0000
AC1         00 1000 0000      AC1         00 1200 0000
*AR0                1000      *AR0                1000
*AR1                2000      *AR1                2000
ACOV0                  0      ACOV0                  0
ACOV1                  0      ACOV1                  0
CARRY                  0      CARRY                  0
FRCT                   0      FRCT                   0
```

LMSF Least Mean Square (LMSF)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**lmsf(**Xmem, Ymem, ACx, ACy**)**	No	4	1	X

Opcode

```
|1000 0111|XXXM MMYY|YMMM SSDD|0110 0001
```

Operands ACx, ACy, Xmem, Ymem, T3

Description This instruction performs three parallel operations in one cycle. The operations are executed in the D-unit MAC and D-unit ALU. The instruction is executed :

```
ACx = T3 * (Ymem)
ACy = ACy + (Xmem) * (Ymem)
Xmem = HI(rnd(ACx + (Xmem)<<#16))
```

The first operation performs a multiplication in D-unit MAC1. The input operands of the multiplier are the content of data register T3 and the content of data memory operand Ymem. The implied T3 operand is sign extended to 17 bits in the MAC1. The data memory operand Ymem is addressed by DAGEN path Y by using Ymem addressing mode, driven on the CDB bus, and sign extended to 17 bits in the MAC1.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

The second operation performs a multiplication and an addition in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Xmem and the content of data memory operand Ymem. The data memory operand Xmem is addressed by DAGEN path X by using Xmem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2. The other data memory operand Ymem is addressed by DAGEN path Y by using the Ymem addressing mode, driven on data bus CDB, and sign extended to 17 bits in the MAC2.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

The third operation performs an addition between an accumulator content and the content of data memory operand Xmem in the D-unit ALU. The data memory operand Xmem is driven on the DDB bus as described in the above second operation, sign extended to 40 bits according to SXMD, shifted to the left by 16 bits, and supplied to the D-unit ALU.

- The shift operation is identical to the arithmetic shift instruction. Therefore, an overflow detection, report, and saturation is done after the shifting operation.
- Overflow and CARRY detection are operated as M40 is locally set to 0.
- Addition overflow is always detected at bit position 31.
- Addition carry report in CARRY status bit is always extracted at bit position 31.
- A rounding is always performed on the result of the addition. The rounding operation depends on the RDM status value.
- When RDM is 0, the biased rounding to the infinite is performed. 2^15 is added to the 40-bit result of the accumulation.
- When RDM is 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit result of accumulation, 2^15 is added as the following pseudo code description.

  ```
  if(2^15 < bit(15–0) < 2^16)
     add 2^15 to the 40–bit result of the accumulation
  else if(bit(15–0) == 2^15)
     if(bit(16) == 1)
        add 2^15 to the 40–bit result of the
           accumulation
  ```

- When an overflow is detected on the result of the rounding, the accumulator is saturated according to SATD. Note that no overflow detection is performed on the intermediate result after the addition but before the rounding.

- If an overflow resulting from the shift, or the addition/rounding, is detected, then the accumulator 0 overflow status bit is set (ACOV0). (In the exceptional case, even if the result of addition is overflowed, the rounding operation may suppress the overflow report.)
- When an overflow is detected, the result is saturated according to SATD, before being stored in memory. Saturation values are 7FFFh or 8000h.
- The result of the third operation, high part of ACx is stored into the data memory location addressed by Xmem via the Ebus.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0 and C54CM = 1, compatibility is ensured due to following the implementation of the lms instruction.

- The rounding is performed without clearing the 16 lowest bits of ACx.
- The addition operation has no overflow detection, report, and saturation after the shifting operation.

Status Bits

Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD,

Affects ACOVx, ACOVy, ACOV0, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description

lmsf(*AR2−,*AR3+,AC0,AC1);
SXM=1, FRCT=1;
assuming 4KW bank DARAM
	The product of the content addressed by AR2 and the content addressed by AR3 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR2, shifted to the left by 16 bits, is added to the content of AC0. The result is rounded and stored in AC0.

```
Execution
T3[16:0] * ((Ymem)[16:0])) -> ACx[39:0]
ACy[39:0]+(Xmem)[16:0]*(Ymem))[16:0])) -> ACy[39:0]
HI(rnd(ACx[39:0]+((Xmem)<<#16))) -> Xmem
```

```
Before                          After
AC0         00 3FFF 8000        AC0         00 0200 0000
AC1         00 0000 8000        AC1         00 0004 8000
T3                  8000        T3                  8000
XAR2             00 30FF        XAR2             00 30FE
XAR3             00 2000        XAR3             00 2001
Data memory
2000h               FE00        2000h               FE00
30FFh               FF00        30FFh               3F00
```

.LR Linear Addressing Qualifier

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**linear()**	No	1	1	AD

Opcode `1001 1100`

Operands none

Description This instruction is an instruction qualifier that can be paralleled only with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing or mar instructions. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone instruction (assembler generates an error message).

When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done linearly (as if ST2_55 register bits 0 to 8 were cleared to 0).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

MOV *Load Accumulator from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(Smem **<<** Tx)	No	3	1	X
[2]	ACx = **low_byte(**Smem**)** **<<** #SHIFTW	No	3	1	X
[3]	ACx = **high_byte(**Smem**)** **<<** #SHIFTW	No	3	1	X
[4]	ACx = Smem **<< #16**	No	2	1	X
[5]	ACx = uns(Smem)	No	3	1	X
[6]	ACx = uns(Smem) **<<** #SHIFTW	No	4	1	X
[7]	ACx = M40(**dbl(**Lmem**)**)	No	3	1	X
[8]	**LO(**ACx**)** = Xmem, **HI(**ACx**)** = Ymem	No	3	1	X

Description

These instructions load a 16-bit signed constant, K16, the content of a memory (Smem) location, the content of a data memory operand (Lmem), or the content of dual data memory operands (Xmem and Ymem) to a selected accumulator (ACx).

Status Bits

Affected by	C54CM, M40, RDM, SATD, SXMD
Affects	ACOVx

See Also

See the following other related instructions:

- ❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator Pair from Memory
- ❑ Load Accumulator with Immediate Value
- ❑ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ❑ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- ❑ Load Auxiliary or Temporary Register Pair from Memory
- ❑ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(Smem **<<** Tx)	No	3	1	X

Opcode

```
|1101  1101|AAAA  AAAI|x%DD  ss11
```

Operands ACx, Smem, Tx

Description This instruction loads the content of a memory (Smem) location shifted by the content of Tx to the accumulator (ACx):

- The input operand is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation. The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits Affected by C54CM, M40, RDM, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 << T0	AC0 is loaded with the content addressed by AR3 shifted by the content of T0.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = **low_byte(**Smem**) <<** #SHIFTW	No	3	1	X

Opcode | 1110 0001 | AAAA AAAI | DDSH IFTW

Operands ACx, SHIFTW, Smem

Description This instruction loads the low-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

❑ The content of the memory location is sign extended to 40 bits according to SXMD.

❑ The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = low_byte(*AR3) << #31	The low-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = **high_byte(**Smem**) <<** #SHIFTW	No	3	1	X

Opcode

```
1110 0010 | AAAA AAAI | DDSH IFTW
```

Operands ACx, SHIFTW, Smem

Description This instruction loads the high-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

- ❑ The content of the memory location is sign extended to 40 bits according to SXMD.
- ❑ The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.
- ❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = high_byte(*AR3) << #31	The high-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = Smem **<< #16**	No	2	1	X

Opcode 1011 00DD AAAA AAAI

Operands ACx, Smem

Description This instruction loads the content of a memory (Smem) location shifted left by 16 bits to the accumulator (ACx):

- ❑ The input operand is sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ The input operand is shifted left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, overflow detection, report, and saturation is done after the shifting operation

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = *AR3+ << #16	The content addressed by AR3 shifted left by 16 bits is loaded into AC1. AR3 is incremented by 1.

```
Before                          After
AC1       00 0200 FC00          AC1       00 3400 0000
AR3               0200          AR3               0201
200               3400          200               3400
```

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = uns(Smem)	No	3	1	X

Opcode `1101 1111 | AAAA AAAI | xxDD 010u`

Operands ACx, Smem

Description This instruction loads the content of a memory (Smem) location to the accumulator (ACx):

- ❑ The memory operand is extended to 40 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ❑ The load operation in the accumulator uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3)	The content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = uns(Smem) **<<** #SHIFTW	No	4	1	X

Opcode `| 1111 1001 | AAAA AAAI | uxSH IFTW | xxDD 10xx`

Operands ACx, SHIFTW, Smem

Description This instruction loads the content of a memory (Smem) location, shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):

- ❑ The memory operand is extended to 40 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ❑ The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) << #31	The content addressed by AR3 is zero extended to 40 bits, shifted left by 31 bits, and loaded into AC0.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = M40(**dbl(**Lmem**)**)	No	3	1	X

Opcode | 1110 1101 | AAAA AAAI | xxDD 100g

Operands ACx, Lmem

Description This instruction loads the content of data memory operand (Lmem) to the accumulator (ACx):

- ❑ The input operand is sign extended to 40 bits according to SXMD.
- ❑ The load operation in the accumulator uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ Status bit M40 is locally set to 1, if the optional M40 keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = dbl(*AR3–)	The content (long word) addressed by AR3 and AR3 + 1 is loaded into AC0. Because this instruction is a long−operand instruction, AR3 is decremented by 2 after the execution.

Load Accumulator from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	**LO(**ACx**)** = Xmem, **HI(**ACx**)** = Ymem	No	3	1	X

Opcode `1000 0001 | XXXM MMYY | YMMM 10DD`

Operands ACx, Xmem, Ymem

Description This instruction performs a dual 16-bit load of accumulator high and low parts. The operation is executed in dual 16-bit mode; however, it is independent of the 40-bit D-unit ALU. The 16 lower bits of the accumulator are separated from the higher 24 bits and the 8 guard bits are attached to the higher 16-bit datapath.

- ❑ The data memory operand Xmem is loaded as a 16-bit operand to the destination accumulator (ACx) low part. And, according to SXMD the data memory operand Ymem is sign extended to 24 bits and is loaded to the destination accumulator (ACx) high part.
- ❑ For the load operations in higher accumulator bits, overflow detection is performed at bit position 31. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ If SATD is 1 when an overflow is detected on the higher data path, a saturation is performed with saturation value of 00 7FFFh.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, this instruction is executed as if SATD was locally cleared to 0.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
LO(AC0) = *AR3, HI(AC0) = *AR4	The content at the location addressed by AR4, sign extended to 24 bits, is loaded into AC0(39–16) and the content at the location addressed by AR3 is loaded into AC0(15–0).

MOV::MOV *Load Accumulator from Memory with Parallel Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = Xmem **<< #16,** Ymem = **HI(**ACx **<< T2)**	No	4	1	X

Opcode `|1000 0111|XXXM MMYY|YMMM SSDD|110x xxxx`

Operands ACx, ACy, T2, Xmem, Ymem

Description This instruction performs two operations in parallel: load and store.

The first operation loads the content of data memory operand Xmem shifted left by 16 bits to the accumulator ACy.

- The input operand is sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- The input operand is shifted left by 16 bits according to M40.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACx(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

- If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = Xmem << #16,
  Ymem = HI(saturate(uns(ACx << T2)))

- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = Xmem << #16,
  Ymem = HI(saturate(ACx << T2))

Status Bits Affected by C54CM, M40, RDM, SATD, SST, SXMD

Affects ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Load Accumulator from Memory
- ❑ Load Accumulator Pair from Memory
- ❑ Load Accumulator with Immediate Value
- ❑ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ❑ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Example

Syntax	**Description**
AC0 = *AR3 << #16, *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31−16) is stored at the address of AR4.

MOV *Load Accumulator Pair from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**pair(HI(**ACx**))** = Lmem	No	3	1	X
[2]	**pair(LO(**ACx**))** = Lmem	No	3	1	X

Description These instructions load the content of a data memory operand (Lmem) to the selected accumulator pair, ACx and AC(x + 1).

Status Bits

Affected by	C54CM, M40, SATD, SXMD
Affects	ACOVx, ACOV(x + 1)

See Also See the following other related instructions:

❑ Load Accumulator from Memory

❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory

❑ Load Accumulator with Immediate Value

❑ Load Accumulator, Auxiliary, or Temporary Register from Memory

❑ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

❑ Load Auxiliary or Temporary Register Pair from Memory

❑ Multiply and Accumulate with Parallel Load Accumulator from Memory

❑ Multiply and Subtract with Parallel Load Accumulator from Memory

Load Accumulator Pair from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**pair(HI(**ACx**))** = Lmem	No	3	1	X

Opcode `| 1110 1101 | AAAA AAAI | 00DD 1010`

Operands ACx, Lmem

Description This instruction loads the 16 MSBs of data memory operand (Lmem) to the 24 MSBs of the destination accumulator (ACx) and loads the 16 LSBs of the data memory operand (Lmem) to the 24 MSBs of the destination accumulator AC(x+1).

- ❑ The 16 MSBs and 16 LSBs of the source memory operand (Lmem) are sign extended to 24 bits and loaded into the 24 MSBs of the destination accumulator ACx and AC(x+1) according to the SXMD.
- ❑ For the load operation in higher accumulator bits, overflow detection is performed at bit position 31. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ The valid combination of source accumulators are AC0/AC1 and AC2/AC3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, overflow detection, report, and saturation are done after the operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx, ACOV(x + 1)

Repeat This instruction can be repeated.

Example

Syntax	Description
pair(HI(AC2)) = *AR3+	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC2(31–16). The 16 lowest bits of the content at the location addressed by AR3 + 1 when the value in AR3 is even or AR3 − 1 when AR3 is odd are loaded into AC3(31–16). AR3 is incremented by 1.

Execution

```
(Lmem[31:16])  -> ACx[39:16],
(Lmem[15:0]) -> AC(x+1)[39:16]
```

```
Before                         After
AC1         FF FFFF 8000       AC1         00 1234 8000
AC2         00 1234 1234       AC2         FF ABCD 8000
XAR3             00 2000       XAR3             00 2002
Data memory
2000h               1234       2000h               1234
2001h               ABCD       2001h               ABCD
```

Load Accumulator Pair from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**pair(LO(**ACx**))** = Lmem	No	3	1	X

Opcode `1110 1101 | AAAA AAAI | 00DD 1100`

Operands ACx, Lmem

Description This instruction loads the 16 MSBs of data memory operand (Lmem) to the 16 LSBs of the destination accumulator (ACx) and loads the 16 LSBs of the data memory operand (Lmem) to the 16 LSBs of the destination accumulator AC(x+1).

❑ The 16 LSBs of the source accumulator ACx is sign extended to 24 bits and loaded into the 24 MSBs of the destination accumulator ACy according to the SXMD.

❑ For the load operation in higher accumulator bits, overflow detection is performed at bit position 31. If an overflow is detected, the destination accumulator overflow status bit is set.

❑ When an overflow is detected on higher data path at SXMD=0 and ACx[15]=1, a saturation is performed with a saturation value of 00 7FFFh.

❑ The valid combination of source accumulators are AC0/AC1 and AC2/AC3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by C54CM, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
pair(LO(AC0)) = *AR3	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC0(15–0). The 16 lowest bits of the content at the location addressed by AR3 + 1 when the value in AR3 is even or AR3 − 1 when AR3 is odd are loaded into AC1(15–0).

Execution

```
(Lmem[31:16])  -> ACx[15:0],
(Lmem[15:0]) -> AC(x+1)[15:0]
```

```
Before                          After
AC3         00 1234 5678        AC3         00 1234 ABCD
AC0         FF FFFF FFFF        AC0         FF FFFF 1234
XAR5             00 2001        XAR5             00 1FFF
Data memory
2000h               1234        2000h               1234
2001h               ABCD        2001h               ABCD
```

MOV *Load Accumulator with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = K16 **<< #16**	No	4	1	X
[2]	ACx = K16 **<<** #SHFT	No	4	1	X

Description These instructions load a 16-bit signed constant, K16, to a selected accumulator (ACx).

Status Bits

Affected by	C54CM, M40, SATD, SXMD
Affects	ACOVx

See Also See the following other related instructions:

- ❏ Load Accumulator from Memory
- ❏ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- ❏ Load Accumulator Pair from Memory
- ❏ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ❏ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- ❏ Load Auxiliary or Temporary Register Pair from Memory
- ❏ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❏ Multiply and Subtract with Parallel Load Accumulator from Memory

Load Accumulator with Immediate Value

Syntax Characteristics

		Parallel			
No.	**Syntax**	**Enable Bit**	**Size**	**Cycles**	**Pipeline**
[1]	ACx = K16 **<< #16**	No	4	1	X

Opcode | `0111 1010` | `KKKK KKKK` | `KKKK KKKK` | `xxDD 101x`

Operands ACx, K16

Description This instruction loads the 16-bit signed constant, K16, shifted left by 16 bits to the accumulator (ACx):

- The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- The input operand is shifted left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	**Description**
AC0 = #−2 << #16	AC0 is loaded with the signed 16-bit value (−2) shifted left by 16 bits.

Load Accumulator with Immediate Value

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = K16 **<<** #SHFT	No	4	1	X

Opcode `|0111 0101|KKKK KKKK|KKKK KKKK|xxDD SHFT`

Operands ACx, K16, SHFT

Description This instruction loads the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, to the accumulator (ACx):

- ❑ The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- ❑ The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = #−2 << #15	AC0 is loaded with the signed 16-bit value (−2) shifted left by 15 bits.

MOV — *Load Accumulator, Auxiliary, or Temporary Register from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = Smem	No	2	1	X
[2]	dst = uns(**high_byte(**Smem**)**)	No	3	1	X
[3]	dst = uns(**low_byte(**Smem**)**)	No	3	1	X

Description These instructions load the content of a memory (Smem) location to a selected destination (dst) register.

Status Bits

Affected by	M40, SXMD
Affects	none

See Also See the following other related instructions:

- ❑ Load Accumulator from Memory
- ❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator Pair from Memory
- ❑ Load Accumulator with Immediate Value
- ❑ Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- ❑ Load Auxiliary or Temporary Register Pair from Memory
- ❑ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Store Accumulator, Auxiliary, or Temporary Register Content to Memory

Load Accumulator, Auxiliary, or Temporary Register from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = Smem	No	2	1	X

Opcode

```
|1010  FDDD|AAAA  AAAI
```

Operands dst, Smem

Description This instruction loads the content of a memory (Smem) location to the destination (dst) register.

- ❑ When the destination register is an accumulator:
  - ■ The content of the memory location is sign extended to 40 bits according to SXMD.
  - ■ The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the destination register is an auxiliary or temporary register:
  - ■ The content of the memory location is sign extended to 16 bits.
  - ■ The load operation in the destination register uses a dedicated path independent of the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AR1 = *AR3+	AR1 is loaded with the content addressed by AR3. AR3 is incremented by 1.

```
Before                   After
AR1           FC00       AR1           3400
AR3           0200       AR3           0201
200           3400       200           3400
```

Load Accumulator, Auxiliary, or Temporary Register from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = uns(**high_byte(**Smem**)**)	No	3	1	X

Opcode

```
|1101 1111|AAAA AAAI|FDDD 000u
```

Operands dst, Smem

Description This instruction loads the high-byte content of a memory (Smem) location to the destination (dst) register.

- ❑ When the destination register is an accumulator:
  - ■ The memory operand is extended to 40 bits according to uns.
    - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
    - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
  - ■ The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the destination register is an auxiliary or temporary register:
  - ■ The memory operand is extended to 16 bits according to uns.
    - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 16 bits.
    - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 16 bits regardless of SXMD.
  - ■ The load operation in the destination register uses a dedicated path independent of the A-unit ALU.
- ❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	**Description**
AC0 = uns(high_byte(*AR3))	The high-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

Load Accumulator, Auxiliary, or Temporary Register from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = uns(**low_byte(**Smem**)**)	No	3	1	X

Opcode | `1101 1111 | AAAA AAAI | FDDD 001u`

Operands dst, Smem

Description This instruction loads the low-byte content of a memory (Smem) location to the destination (dst) register.

- ❑ When the destination register is an accumulator:
  - The memory operand is extended to 40 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the destination register is an auxiliary or temporary register:
  - The memory operand is extended to 16 bits according to uns.
    - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 16 bits.
    - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 16 bits regardless of SXMD.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.
- ❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(low_byte(*AR3))	The low-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

MOV *Load Accumulator, Auxiliary, or Temporary Register with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = k4	Yes	2	1	X
[2]	dst = –k4	Yes	2	1	X
[3]	dst = K16	No	4	1	X

Description

These instructions load a 4-bit unsigned constant, k4; the 2s complement representation of the 4-bit unsigned constant; or a 16-bit signed constant, K16, to a selected destination (dst) register.

Status Bits

Affected by	M40, SXMD
Affects	none

See Also

See the following other related instructions:

❑ Load Accumulator from Memory

❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory

❑ Load Accumulator Pair from Memory

❑ Load Accumulator with Immediate Value

❑ Load Accumulator, Auxiliary, or Temporary Register from Memory

❑ Load Auxiliary or Temporary Register Pair from Memory

❑ Multiply and Accumulate with Parallel Load Accumulator from Memory

❑ Multiply and Subtract with Parallel Load Accumulator from Memory

Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = k4	Yes	2	1	X

Opcode `| 0011 110E | kkkk FDDD`

Operands dst, k4

Description This instruction loads the 4-bit unsigned constant, k4, to the destination (dst) register.

- When the destination register is an accumulator:
  - The 4-bit constant, k4, is zero extended to 40 bits.
  - The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- When the destination register is an auxiliary or temporary register:
  - The 4-bit constant, k4, is zero extended to 16 bits.
  - The load operation in the destination register uses a dedicated path independent of the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = #2	AC0 is loaded with the unsigned 4-bit value (2).

Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = –k4	Yes	2	1	X

Opcode `0011 111E kkkk FDDD`

Operands dst, k4

Description This instruction loads the 2s complement representation of the 4-bit unsigned constant, k4, to the destination (dst) register.

- ❑ When the destination register is an accumulator:
  - ■ The 4-bit constant, k4, is negated in the I-unit, loaded into the accumulator, and sign extended to 40 bits before being processed by the D-unit as a signed constant.
  - ■ The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the destination register is an auxiliary or temporary register:
  - ■ The 4-bit constant, k4, is zero extended to 16 bits and negated in the I-unit before being processed by the A-unit as a signed K16 constant.
  - ■ The load operation in the destination register uses a dedicated path independent of the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = #−2	AC0 is loaded with a 2s complement representation of the unsigned 4-bit value (2).

Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = K16	No	4	1	X

Opcode `0111 0110 | KKKK KKKK | KKKK KKKK | FDDD 10xx`

Operands dst, K16

Description This instruction loads the 16-bit signed constant, K16, to the destination (dst) register.

- When the destination register is an accumulator, the 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- When the destination register is an auxiliary or temporary register, the load operation in the destination register uses a dedicated path independent of the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = #248	AC1 is loaded with the signed 16-bit value (248).

```
Before                          After
AC1         00 0200 FC00        AC1         00 0000 00F8
```

MOV *Load Auxiliary or Temporary Register Pair from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**pair(**TAx**)** = Lmem	No	3	1	X

Opcode `1110 1101 | AAAA AAAI | FDDD 111x`

Operands Lmem, TAx

Description This instruction loads the 16 highest bits of data memory operand (Lmem) to the temporary or auxiliary register (TAx) and loads the 16 lowest bits of data memory operand (Lmem) to temporary or auxiliary register TA(x + 1):

- The load operation in the temporary or auxiliary register uses a dedicated path independent of the A-unit ALU.
- Valid auxiliary registers are AR0, AR2, AR4, and AR6.
- Valid temporary registers are T0 and T2.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Load Accumulator, Auxiliary, or Temporary Register from Memory
- Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- Modify Auxiliary or Temporary Register Content

Example

Syntax	Description
pair(T0) = *AR2	The 16 highest bits of the content at the location addressed by AR2 are loaded into T0 and the 16 lowest bits of the content at the location addressed by AR2 + 1 are loaded into T1.

MOV *Load CPU Register from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**BK03** = Smem	No	3	1	X
[2]	**BK47** = Smem	No	3	1	X
[3]	**BKC** = Smem	No	3	1	X
[4]	**BSA01** = Smem	No	3	1	X
[5]	**BSA23** = Smem	No	3	1	X
[6]	**BSA45** = Smem	No	3	1	X
[7]	**BSA67** = Smem	No	3	1	X
[8]	**BSAC** = Smem	No	3	1	X
[9]	**BRC0** = Smem	No	3	1	X
[10]	**BRC1** = Smem	No	3	1	X
[11]	**CDP** = Smem	No	3	1	X
[12]	**CSR** = Smem	No	3	1	X
[13]	**DP** = Smem	No	3	1	X
[14]	**DPH** = Smem	No	3	1	X
[15]	**PDP** = Smem	No	3	1	X
[16]	**SP** = Smem	No	3	1	X
[17]	**SSP** = Smem	No	3	1	X
[18]	**TRN0** = Smem	No	3	1	X
[19]	**TRN1** = Smem	No	3	1	X
[20]	**RETA** = **dbl(**Lmem**)**	No	3	5	X

Opcode See Table 5−1 (page 5-212).

Operands Lmem, Smem

Description

Instructions [1] through [19] load the content of a memory (Smem) location to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The content of the memory location is zero extended to the bitwidth of the destination CPU register.

The operation is performed in the execute phase of the pipeline. There is a 3-cycle latency between PDP, DP, SP, SSP, CDP, BSAx, BKx, BRCx, and CSR loads and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

For instruction [10], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.

Instruction [20] loads the content of data memory operand (Lmem) to the 24-bit RETA register (the return address of the calling subroutine) and to the 8-bit CFCT register (active control flow execution context flags of the calling subroutine):

- The 16 highest bits of Lmem are loaded into the CFCT register and into the 8 highest bits of the RETA register.
- The 16 lowest bits of Lmem are loaded into the 16 lowest bits of the RETA register.

  When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.

Status Bits

Affected by	none
Affects	none

Repeat

Instructions [13] and [20] cannot be repeated; all other instructions can be repeated.

See Also

See the following other related instructions:

- Load CPU Register with Immediate Value

Table 5−1. Opcodes for Load CPU Register from Memory Instruction

No.	Syntax	Opcode
[1]	**BK03** = Smem	`1101 1100 AAAA AAAI 1001 xx10`
[2]	**BK47** = Smem	`1101 1100 AAAA AAAI 1010 xx10`
[3]	**BKC** = Smem	`1101 1100 AAAA AAAI 1011 xx10`
[4]	**BSA01** = Smem	`1101 1100 AAAA AAAI 0010 xx10`
[5]	**BSA23** = Smem	`1101 1100 AAAA AAAI 0011 xx10`
[6]	**BSA45** = Smem	`1101 1100 AAAA AAAI 0100 xx10`
[7]	**BSA67** = Smem	`1101 1100 AAAA AAAI 0101 xx10`
[8]	**BSAC** = Smem	`1101 1100 AAAA AAAI 0110 xx10`
[9]	**BRC0** = Smem	`1101 1100 AAAA AAAI x001 xx11`
[10]	**BRC1** = Smem	`1101 1100 AAAA AAAI x010 xx11`
[11]	**CDP** = Smem	`1101 1100 AAAA AAAI 0001 xx10`
[12]	**CSR** = Smem	`1101 1100 AAAA AAAI x000 xx11`
[13]	**DP** = Smem	`1101 1100 AAAA AAAI 0000 xx10`
[14]	**DPH** = Smem	`1101 1100 AAAA AAAI 1100 xx10`
[15]	**PDP** = Smem	`1101 1100 AAAA AAAI 1111 xx10`
[16]	**SP** = Smem	`1101 1100 AAAA AAAI 0111 xx10`
[17]	**SSP** = Smem	`1101 1100 AAAA AAAI 1000 xx10`
[18]	**TRN0** = Smem	`1101 1100 AAAA AAAI x011 xx11`
[19]	**TRN1** = Smem	`1101 1100 AAAA AAAI x100 xx11`
[20]	**RETA** = **dbl(**Lmem**)**	`1110 1101 AAAA AAAI xxxx 011x`

MOV *Load CPU Register with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**BK03** = k12	Yes	3	1	AD
[2]	**BK47** = k12	Yes	3	1	AD
[3]	**BKC** = k12	Yes	3	1	AD
[4]	**BRC0** = k12	Yes	3	1	AD
[5]	**BRC1** = k12	Yes	3	1	AD
[6]	**CSR** = k12	Yes	3	1	AD
[7]	**DPH** = k7	Yes	3	1	AD
[8]	**PDP** = k9	Yes	3	1	AD
[9]	**BSA01** = k16	No	4	1	AD
[10]	**BSA23** = k16	No	4	1	AD
[11]	**BSA45** = k16	No	4	1	AD
[12]	**BSA67** = k16	No	4	1	AD
[13]	**BSAC** = k16	No	4	1	AD
[14]	**CDP** = k16	No	4	1	AD
[15]	**DP** = k16	No	4	1	AD
[16]	**SP** = k16	No	4	1	AD
[17]	**SSP** = k16	No	4	1	AD

Opcode See Table 5−2 (page 5-214).

Operands kx

Description These instructions load the unsigned constant, kx, to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The constant is zero extended to the bitwidth of the destination CPU register.

For instruction [5], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.

The operation is performed in the address phase of the pipeline.

Status Bits	Affected by	none
	Affects	none

Repeat Instruction [15] cannot be repeated; all other instructions can be repeated.

See Also See the following other related instructions:

- ❑ Load CPU Register from Memory

Table 5−2. Opcodes for Load CPU Register with Immediate Value Instruction

No.	Syntax	Opcode
[1]	**BK03** = k12	`0001 011E kkkk kkkk kkkk 0100`
[2]	**BK47** = k12	`0001 011E kkkk kkkk kkkk 0101`
[3]	**BKC** = k12	`0001 011E kkkk kkkk kkkk 0110`
[4]	**BRC0** = k12	`0001 011E kkkk kkkk kkkk 1001`
[5]	**BRC1** = k12	`0001 011E kkkk kkkk kkkk 1010`
[6]	**CSR** = k12	`0001 011E kkkk kkkk kkkk 1000`
[7]	**DPH** = k7	`0001 011E xxxx xkkk kkkk 0000`
[8]	**PDP** = k9	`0001 011E xxxk kkkk kkkk 0011`
[9]	**BSA01** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 011x`
[10]	**BSA23** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 100x`
[11]	**BSA45** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 101x`
[12]	**BSA67** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 110x`
[13]	**BSAC** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 111x`
[14]	**CDP** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 010x`
[15]	**DP** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 000x`
[16]	**SP** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx1 000x`
[17]	**SSP** = k16	`0111 1000 kkkk kkkk kkkk kkkk xxx0 001x`

MOV *Load Extended Auxiliary Register from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = **dbl(**Lmem**)**	No	3	1	X

Opcode | `1110 1101 | AAAA AAAI | XDDD 1111`

Operands Lmem , XAdst

Description This instruction loads the lower 23 bits of the data addressed by data memory operand (Lmem) to the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Load Extended Auxiliary Register with Immediate Value
- ❑ Modify Extended Auxiliary Register Content
- ❑ Move Extended Auxiliary Register Content
- ❑ Store Extended Auxiliary Register Content to Memory

Example

Syntax	Description
XAR1 = dbl(*AR3)	The 7 lowest bits of the content at the location addressed by AR3 and the 16 bits of the content at the location addressed by AR3 + 1 are loaded into XAR1.

```
Before                          After
XAR1            00 0000         XAR1            12 0FD3
AR3                0200         AR3                0200
200                3492         200                3492
201                0FD3         201                0FD3
```

AMOV *Load Extended Auxiliary Register with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = k23	No	6	1	AD

Opcode `| 1110 1100 | AAAA AAAI | 0DDD 1110`

Operands k23, XAdst

Description This instruction loads a 23-bit unsigned constant (k23) into the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address generator. Data memory is not accessed.

The premodification or postmodification of the auxiliary register (ARx), the use of *port(#K), and the use of the readport() or writeport() qualifier is not supported for this instruction. The use of auxiliary register offset operations is supported. If the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management also controls the result stored in XAdst.

Status Bits Affected by ST2_55

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❏ Load Extended Auxiliary Register from Memory

❏ Modify Extended Auxiliary Register Content

❏ Move Extended Auxiliary Register Content

❏ Store Extended Auxiliary Register Content to Memory

Example

Syntax	Description
XAR0 = #7FFFFFh	The 23-bit value (7FFFFFh) is loaded into XAR0.

MOV *Load Memory with Immediate Value*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = K8	No	3	1	X
[2]	Smem = K16	No	4	1	X

Opcode

```
 K8                      |1110  0110 |AAAA  AAAI |KKKK  KKKK
 K16         |1111  1011 |AAAA  AAAI |KKKK  KKKK |KKKK  KKKK
```

Operands Kx, Smem

Description These instructions initialize a data memory location. These instructions store an 8-bit signed constant, K8, or a 16-bit signed constant, K16, to a memory (Smem) location. They use a dedicated datapath to perform the operation.

For instruction [1], the immediate value is always signed extended to 16 bits before being stored in memory.

Status Bits

Affected by	none
Affects	none

Repeat Both instructions [1] and [2] can be repeated.

See Also See the following other related instructions:

❑ Move Memory to Memory

Example

Syntax	Description
*(#0501h) = #248	The signed 16-bit value (248) is loaded to address 501h.

Before		**After**	
0501	FC00	0501	F800

.LK *Lock Access Qualifier*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	lock()	No	2	1	D

Opcode | | 0100 0101 | 1111 0010

Operands none

Description This is an operand qualifier that can be paralleled with any of 13 instructions (listed below) which execute a read-modify-write operation to a specific memory operand. If the lock() qualifier is applied to any of 13 instructions, the lock signal is activated at the same cycle with the read request and the corresponding write request follows this read request. This means any memory request issued by other instructions cannot be located between this locked read and write request due to stall generation. This also provides a suitable interface with the OCP.

This operand qualifier cannot be executed:

- ❑ Alone
- ❑ In parallel with instructions except the 13 lock instructions

Any of the 13 instructions using the lock() qualifier cannot be combined with any other user-defined parallelism instruction.

The 13 lock instructions which can be paralleled with the lock() qualifier are listed in the table below.

Number	Algebraic	Mnemonic
1	TC1 = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TC1
2	TC2 = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TC2
3	TC1 = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TC1
4	TC2 = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TC2
5	TC1 = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TC1
6	TC2 = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TC2
7	bit(Smem, src) = #1	BSET src, Smem
8	bit(Smem, src) = #0	BCLR src, Smem
9	cbit(Smem, src)	BNOT src, Smem
10	Smem = Smem & k16	AND k16, Smem
11	Smem = Smem | k16	OR k16, Smem
12	Smem = Smem ^ k16	XOR k16, Smem
13	Smem = Smem + k16	ADD k16, Smem

Any of the 13 instructions with the lock() qualifier is not allowed in the conditional execution context which is applied by “if(cond) execute(D_unit)” instruction due to OCP compliance. The cases below are illegal and rejected by the code-gen tools:

```
if(cond execute(D_unit)
TC1=bit(*ar2+, #2), bit(*ar2+, #2)=#1 || lock()
```

```
instruction || if(cond) execute(D_unit)
TC1=bit(*ar2+, #2), bit(*ar2+, #2)=#1 || lock()
```

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example `TC1=bit(*ar2+, #2), bit(*ar2+, #2)=#1 || lock()`

Before		**After**	
XAR2	00 1780	XAR2	00 1781
Data memory			
1780h	FE00	1780h	FE04
1781h	3800	1781h	3800
TC1	x		0

DELAY *Memory Delay*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**delay(**Smem**)**	No	2	1	X

Opcode `1011 0110 | AAAA AAAI`

Operands Smem

Description This instruction copies the content of the memory (Smem) location into the next higher address (Smem + 1). When the data is copied, the content of the addressed location remains the same. A dedicated datapath is used to make this memory move.

When this instruction is executed, the two address register arithmetic units ARAU X and Y, of the A-unit data address generator unit, are used to compute the two addresses Smem and Smem + 1. The address generation is not affected by circular addressing; if Smem points to the end of a circular buffer, Smem + 1 will point to an address outside the circular buffer.

The soft dual memory addressing mode mechanism cannot be applied to this instruction. This instruction cannot use the *port(#k16) addressing mode or be paralleled with the readport() or writeport() operand qualifier.

This instruction cannot be used for accesses to I/O space. Any illegal access to I/O space generates a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
delay(*AR1+)	The content addressed by AR1 is copied to the next higher address, AR1 + 1. AR1 is incremented by 1.

Before		**After**	
AR1	0200	AR1	0201
200	3400	200	3400
201	0D80	201	3400
202	2030	202	2030

mmap *Memory-Mapped Register Access Qualifier*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mmap()**	No	1	1	D

Opcode `1001 1000`

Operands none

Description This is an operand qualifier that can be paralleled with any instruction making a Smem or Lmem direct memory access (dma). This operand qualifier allows you to locally prevent the dma access from being relative to the data stack pointer (SP) or the local data page register (DP). It forces the dma access to be relative to the memory-mapped register (MMR) data page start address, 00 0000h.

This operand qualifier cannot be executed:

- ❑ as a stand-alone instruction (assembler generates an error message)
- ❑ in parallel with instructions not embedding an Smem or Lmem data memory operand
- ❑ in parallel with instructions loading or storing a byte to a register (see Load Accumulator, Auxiliary, or Temporary Register from Memory instructions [2] and [3]; Load Accumulator from Memory instructions [2] and [3]; and Store Accumulator, Auxiliary, or Temporary Register Content to Memory instructions [2] and [3])

The MMRs are mapped as 16-bit data entities between addresses 0h and 5Fh. The scratch-pad memory that is mapped between addresses 60h and 7Fh of each main data pages of 64K words cannot be accessed through this mechanism.

Any instruction using the mmap() modifier cannot be combined with any other user-defined parallelism instruction.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
T2 = @(AC0_L)) || mmap()	AC0_L is a keyword representing AC0(15–0). The content of AC0(15–0) is copied into T2.

AMAR *Modify Auxiliary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**Smem**)**	No	2	1	AD

Opcode

```
|1011  0100|AAAA  AAAI
```

Operands Smem

Description This instruction performs, in the A-unit address generation units, the auxiliary register modification specified by Smem as if a word single data memory operand access was made. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits

Affected by	ST2_55
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary or Temporary Register Content
- ❑ Modify Auxiliary or Temporary Register Content by Addition
- ❑ Modify Auxiliary or Temporary Register Content by Subtraction
- ❑ Modify Auxiliary Register Content with Parallel Multiply
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Modify Extended Auxiliary Register Content
- ❑ Modify Extended Auxiliary Register Content by Addition
- ❑ Modify Extended Auxiliary Register Content by Subtraction
- ❑ Parallel Modify Auxiliary Register Contents

Example

Syntax	Description
mar(*AR3+)	The content of AR3 is incremented by 1.

AMAR::MPY *Modify Auxiliary Register Content with Parallel Multiply*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**Xmem**),** ACx = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**)**)))	No	4	1	X

Opcode `|1000 0010|XXXM MMYY|YMMM 11mm|uuxx DDg%`

Operands ACx, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

- ❑ This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
- ❑ For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply

Example

Syntax	Description
mar(*AR3+), AC0 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0.

AMAR::MAC *Modify Auxiliary Register Content with Parallel Multiply and Accumulate*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**Xmem**),** ACx = M40(rnd(ACx + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X
[2]	**mar(**Xmem**),** ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Description These instructions perform two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content
- ❑ Modify Auxiliary Register Content with Parallel Multiply
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Accumulate

Modify Auxiliary Register Content with Parallel Multiply and Accumulate

Syntax Characteristics

		Parallel			
No.	**Syntax**	**Enable Bit**	**Size**	**Cycles**	**Pipeline**
[1]	**mar(**Xmem**),** ACx = M40(rnd(ACx + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode | 1000 0011 | XXXM MMYY | YMMM 11mm | uuxx DDg%

Operands ACx, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.

- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
- ❑ For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	**Description**
mar(*AR3+), AC0 = AC0 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0.

Modify Auxiliary Register Content with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**mar(**Xmem**),** ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode `1000 0100 | XXXM MMYY | YMMM 01mm | uuxx DDg%`

Operands ACx, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

❑ Input operands are extended to 17 bits according to uns.

  ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.

  ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.

❑ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
mar(*AR2+), AC0 = ((AC0 >> #16) + (uns(*AR1) * uns(coef(*CDP))))	Both instructions are performed in parallel. AR2 is incremented by 1. The unsigned content addressed by AR1 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 shifted right by 16 bits and the result is stored in AC0. An overflow is detected in AC0.

```
Before                             After
AC0         00 6900 0000           AC0         00 95C0 9200
AC1         00 0023 0000           AC1         00 0023 0000
*AR1                EF00           *AR1                EF00
AR2                 0201           AR2                 0202
*CDP                A067           *CDP                A067
ACOV0                  0           ACOV0                  1
ACOV1                  0           ACOV1                  0
CARRY                  0           CARRY                  0
M40                    0           M40                    0
FRCT                   0           FRCT                   0
SATD                   0           SATD                   0
```

AMAR::MAS *Modify Auxiliary Register Content with Parallel Multiply and Subtract*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**Xmem**),** ACx = M40(rnd(ACx – **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode `| 1000 0101 | XXXM MMYY | YMMM 00mm | uuxx DDg%`

Operands ACx, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

The second operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.

❑ Input operands are extended to 17 bits according to uns.

- ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
- ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.

❑ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.

- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
- ❑ For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content
- ❑ Modify Auxiliary Register Content with Parallel Multiply
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Multiply and Subtract

Example

Syntax	**Description**
mar(*AR3+), AC0 = AC0 – (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0.

AMOV *Modify Auxiliary or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**TAy = TAx**)**	No	3	1	AD
[2]	**mar(**TAx = P8**)**	No	3	1	AD
[3]	**mar(**TAx = D16**)**	No	4	1	AD

Description

These instructions perform, in the A-unit address generation units:

- a move from auxiliary or temporary register TAx to auxiliary or temporary register TAy
- a load in the auxiliary or temporary registers TAx of a program address defined by a program address label assembled into P8
- a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16

The operation is performed in the address phase of the pipeline, however data memory is not accessed.

Status Bits

Affected by	none
Affects	none

See Also

See the following other related instructions:

- Load Auxiliary or Temporary Register from Memory
- Modify Auxiliary Register Content
- Modify Auxiliary or Temporary Register Content by Addition
- Modify Auxiliary or Temporary Register Content by Subtraction
- Modify Extended Auxiliary Register Content
- Modify Extended Auxiliary Register Content by Addition
- Modify Extended Auxiliary Register Content by Subtraction

Modify Auxiliary or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**TAy = TAx**)**	No	3	1	AD

Opcode

```
|0001  010E|FSSS  xxxx|FDDD  0001

|0001  010E|FSSS  xxxx|FDDD  1001
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, TAy

Description This instruction performs, in the A-unit address generation units, a move from the auxiliary or temporary register TAx to auxiliary or temporary register TAy. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example 1

Syntax	Description
mar(AR0 = AR1)	The content of AR1 is copied to AR0.

Example 2

Syntax	Description
mar(T0 = T1)	The content of T1 is copied to T0.

Modify Auxiliary or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**mar(**TAx = P8**)**	No	3	1	AD

Opcode

```
|0001 010E|PPPP PPPP|FDDD 0101

|0001 010E|PPPP PPPP|FDDD 1101
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, P8

Description This instruction performs, in the A-unit address generation units, a load in the auxiliary or temporary registers TAx of a program address defined by a program address label assembled into P8. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example 1

Syntax	Description
mar(AR0 = #255)	The unsigned 8-bit value (255) is copied to AR0.

Example 2

Syntax	Description
mar(T0 = #255)	The unsigned 8-bit value (255) is copied to T0.

Modify Auxiliary or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	**mar(**TAx = D16**)**	No	4	1	AD

Opcode

```
| 0111  0111 | DDDD  DDDD | DDDD  DDDD | FDDD  xxxx
```

Operands TAx, D16

Description This instruction performs, in the A-unit address generation units, a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Status Bits

Status Bits	Affected by	none
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
mar(T1 = #FFFFh)	The address FFFFh is copied to T1.

AADD *Modify Auxiliary or Temporary Register Content by Addition*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**TAy + TAx**)**	No	3	1	AD
[2]	**mar(**TAx + P8**)**	No	3	1	AD

Description

These instructions perform, in the A-unit address generation units:

- ❑ an addition between two auxiliary or temporary registers, TAx and TAy, and stores the result in TAy
- ❑ an addition between the auxiliary or temporary registers TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx

The operation is performed in the address phase of the pipeline, however data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Status Bits

Affected by	ST2_55
Affects	none

See Also

See the following other related instructions:

- ❑ Modify Auxiliary Register Content
- ❑ Modify Auxiliary or Temporary Register Content
- ❑ Modify Auxiliary or Temporary Register Content by Subtraction
- ❑ Modify Extended Auxiliary Register Content
- ❑ Modify Extended Auxiliary Register Content by Addition
- ❑ Modify Extended Auxiliary Register Content by Subtraction

Modify Auxiliary or Temporary Register Content by Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**TAy + TAx**)**	No	3	1	AD

Opcode

```
|0001 010E|FSSS xxxx|FDDD 0000

|0001 010E|FSSS xxxx|FDDD 1000
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, TAy

Description This instruction performs, in the A-unit address generation units, an addition between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. The content of TAx is considered signed. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits

Affected by	ST2_55
Affects	none

Repeat This instruction can be repeated.

Example 1

Syntax	Description
mar(AR0 + T0)	The content of AR0 is added to the signed content of T0 and the result is stored in AR0.

```
Before                         After
XAR0              01 0000      XAR0              00 8000
T0                   8000      T0                   8000
```

Example 2

Syntax	Description
mar(T0 + T1)	The content of T0 is added to the content of T1 and the result is stored in T0.

Modify Auxiliary or Temporary Register Content by Addition

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**mar(**TAx + P8**)**	No	3	1	AD

Opcode

```
|0001 010E|PPPP PPPP|FDDD 0100
|0001 010E|PPPP PPPP|FDDD 1100
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, P8

Description This instruction performs, in the A-unit address generation units, an addition between the auxiliary or temporary register TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits

Affected by	ST2_55
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
mar(T0 + #255)	The unsigned 8-bit value (255) is added to the content of T0 and the result is stored in T0.

ASUB *Modify Auxiliary or Temporary Register Content by Subtraction*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**TAy – TAx**)**	No	3	1	AD
[2]	**mar(**TAx – P8**)**	No	3	1	AD

Description

These instructions perform, in the A-unit address generation units:

- ❑ a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy
- ❑ a subtraction between the auxiliary or temporary registers TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx

The operation is performed in the address phase of the pipeline, however data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Status Bits

Affected by	ST2_55
Affects	none

See Also

See the following other related instructions:

- ❑ Modify Auxiliary Register Content
- ❑ Modify Auxiliary or Temporary Register Content
- ❑ Modify Auxiliary or Temporary Register Content by Addition
- ❑ Modify Extended Auxiliary Register Content
- ❑ Modify Extended Auxiliary Register Content by Addition
- ❑ Modify Extended Auxiliary Register Content by Subtraction

Modify Auxiliary or Temporary Register Content by Subtraction

Syntax Characteristics

		Parallel			
No.	**Syntax**	**Enable Bit**	**Size**	**Cycles**	**Pipeline**
[1]	**mar(**TAy – TAx**)**	No	3	1	AD

Opcode

```
|0001 010E|FSSS xxxx|FDDD 0010
|0001 010E|FSSS xxxx|FDDD 1010
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, TAy

Description This instruction performs, in the A-unit address generation units, a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. The content of TAx is considered signed. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits Affected by ST2_55

Affects none

Repeat This instruction can be repeated.

Example 1

Syntax	Description
mar(AR0 – T0)	The signed content of T0 is subtracted from the content of AR0 and the result is stored in AR0.

```
Before                           After
XAR0             01 8000         XAR0             01 0000
T0                  8000         T0                  8000
```

Example 2

Syntax	Description
mar(T0 – T1)	The content of T1 is subtracted from the content of T0 and the result is stored in T0.

Modify Auxiliary or Temporary Register Content by Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**mar(**TAx – P8**)**	No	3	1	AD

Opcode

```
|0001 010E|PPPP PPPP|FDDD 0110
|0001 010E|PPPP PPPP|FDDD 1110
```

The assembler selects the opcode depending on the instruction position in a paralleled pair.

Operands TAx, P8

Description This instruction performs, in the A-unit address generation units, a subtraction between the auxiliary or temporary register TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is in paralleled, the circular buffer management controls the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the mar() instruction must be executed with C54CM set to 1.

When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.

Status Bits		
	Affected by	ST2_55
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
mar(AR0 – #255)	The unsigned 8-bit value (255) is subtracted from the signed content of AR0 and the result is stored in AR0.

AADD *Modify Data Stack Pointer*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SP = SP + K8	Yes	2	1	AD

Opcode `| 0100 111E | KKKK KKKK`

Operands K8

Description This instruction performs an addition in the A-unit data-address generation unit (DAGEN) in the address phase of the pipeline. The 8-bit signed constant, K8, is sign extended to 16 bits and added to the data stack pointer (SP). When in 32-bit stack configuration, the system stack pointer (SSP) is also modified. Updates of the SP and SSP (depending on the stack configuration) should not be executed in parallel with this instruction.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
SP = SP + #127	The 8-bit value (127) is sign extended to 16 bits and added to the stack pointer (SP).

AMAR *Modify Extended Auxiliary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = **mar(**Smem**)**	No	3	1	AD
[2]	**mar(**XACdst = XACsrc**)**	Yes	3	1	AD

Description These instructions perform, in the A-unit address generation units:

- The effective address specified by the Smem operand field and modifies the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). Data memory is not accessed.
- A full 23-bit move from one addressing register to another addressing register, from XACsrc to XACdst, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Status Bits

Affected by	ST2_55
Affects	none

See Also See the following other related instructions:

- Load Extended Auxiliary Register from Memory
- Load Extended Auxiliary Register with Immediate Value
- Modify Auxiliary Register Content
- Move Extended Auxiliary Register Content
- Store Extended Auxiliary Register Content to Memory
- Modify Extended Auxiliary Register Content by Addition
- Modify Extended Auxiliary Register Content by Subtraction

Modify Extended Auxiliary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XAdst = **mar(**Smem**)**	No	3	1	AD

Opcode `1110 1100 | AAAA AAAI | XDDD 1110`

Operands Smem, XAdst

Description This instruction computes the effective address specified by the Smem operand field and modifies the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address generator. Data memory is not accessed.

The premodification or postmodification of the auxiliary register (ARx), the use of *port(#K), and the use of the readport() or writeport() qualifier is not supported for this instruction. The use of auxiliary register offset operations is supported. If the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management also controls the result stored in XAdst.

Status Bits

Affected by	ST2_55
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
XAR0 = mar(*AR1)	The content of AR1 is loaded into XAR0.

Modify Extended Auxiliary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**mar(**XACdst = XACsrc**)**	Yes	3	1	AD

Opcode

```
DAG_X:    |0001  010E|XACS  0001|XACD  0001
DAG_Y:    |0001  010E|XACS  0001|XACD  1001
```

Operands XARx, XARy, XCDP

Description This instruction performs, in the A-unit address generation units, a full 23-bit move from one addressing register to another addressing register, from XACsrc to XACdst, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by

Affects

Repeat This instruction can be repeated.

Example 1

Syntax	Description
mar(XAR1 = XAR0)	The content of XAR0 is copied to XAR1.

```
Before                        After
XAR0          12 3456         XAR0          12 3456
XAR1          43 5634         XAR1          12 3456
```

Example 2

Syntax	Description
mar(XCDP = XAR7)	The content of XAR7 is copied to XCDP.

```
Before                        After
XCDP          00 8000         XCDP          01 4000
XAR7          01 4000         XAR7          01 4000
```

```
Execution
(XACsrc)  -> XACdst
```

AADD *Modify Extended Auxiliary Register Content by Addition*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**XACdst + XACsrc**)**	Yes	3	1	AD

Opcode

```
DAG_X:      | 0001  010E | XACS  0001 | XACD  0000
DAG_Y:      | 0001  010E | XACS  0001 | XACD  1000
```

Operands XARx, XARy, XCDP

Description This instruction performs, in the A-unit address generation units, a full 23-bit unsigned addition between two auxiliary or other addressing registers, XACdst and XACsrc, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Since the operation performed is an unsigned operation, if a destination register is an auxiliary register, it is not allowed to use the circular addressing mode; that is, the result of setting the corresponding bit (ARnLC) in status register ST2_55 to 1 is undefined and using the circular addressing qualifier operating in parallel is not allowed.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by

Affects

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Modify Auxiliary or Temporary Register Content
- Modify Auxiliary or Temporary Register Content by Addition
- Modify Auxiliary or Temporary Register Content by Subtraction
- Modify Auxiliary Register Content with Parallel Multiply
- Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- Modify Auxiliary Register Content with Parallel Multiply and Subtract
- Modify Extended Auxiliary Register Content
- Modify Extended Auxiliary Register Content by Subtraction
- Parallel Modify Auxiliary Register Contents

Example 1

Syntax	Description
mar(XAR1 + XAR0)	The content of XAR0 is added to XAR1 and stored in XAR1.

```
Before                          After
XAR0            12 3456         XAR0            12 3456
XAR1            43 5634         XAR1            55 8A8A
```

Example 2

Syntax	Description
mar(XCDP + XAR7)	The content of XAR7 is added to XCDP and stored in XCDP.

```
Before                          After
XCDP            00 8000         XCDP            01 0080
XAR7            00 8080         XAR7            00 8080

Execution
(XACdst) + (XACsrc)  –> XACdst
```

ASUB Modify Extended Auxiliary Register Content by Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**XACdst − XACsrc**)**	Yes	3	1	AD

Opcode

```
DAG_X:    |0001  010E|XACS  0001|XACD  0010
DAG_Y:    |0001  010E|XACS  0001|XACD  1010
```

Operands XARx, XARy, XCDP

Description This instruction performs, in the A-unit address generation units, a full 23-bit subtraction between two auxiliary or other addressing registers, XACdst and XACsrc, and stores the result in XACdst. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1 or the circular addressing qualifier is operating in parallel, the circular buffer management does not control the result stored in the destination register.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by

Affects

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary or Temporary Register Content
- ❑ Modify Auxiliary or Temporary Register Content by Addition
- ❑ Modify Auxiliary or Temporary Register Content by Subtraction
- ❑ Modify Auxiliary Register Content with Parallel Multiply
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Modify Extended Auxiliary Register Content
- ❑ Modify Extended Auxiliary Register Content by Addition
- ❑ Parallel Modify Auxiliary Register Contents

Example 1

Syntax	Description
mar(XAR1 − XAR0)	The content of XAR0 is subtracted from XAR1 and stored in XAR1.

```
Before                           After
XAR0             12 3456         XAR0             12 3456
XAR1             43 5634         XAR1             31 21DE
```

Example 2

Syntax	Description
mar(XCDP − XAR7)	The content of XAR7 is subtracted from XCDP and stored in XCDP.

```
Before                           After
XCDP             00 8000         XCDP             00 7000
XAR7             00 1000         XAR7             00 1000

Execution
(XACdst) – (XACsrc)  –> XACdst
```

MOV *Move Accumulator Content to Auxiliary or Temporary Register*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TAx = **HI(**ACx**)**	Yes	2	1	X

Opcode | 0100 010E | 00SS FDDD

Operands ACx, TAx

Description This instruction moves the high part of the accumulator, ACx(31−16), to the destination auxiliary or temporary register (TAx). The 16-bit move operation is performed in the A-unit ALU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Move Accumulator, Auxiliary, or Temporary Register Content
- ❑ Move Auxiliary or Temporary Register Content to Accumulator

Example

Syntax	Description
AR2 = HI(AC0)	The content of AC0(31–16) is copied to AR2.

```
Before                          After
AC0         01 E500 0030        AC0         01 E500 0030
AR2                 0200        AR2                 E500
```

MOV *Move Accumulator, Auxiliary, or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = src	Yes	2	1	X

Opcode `0010 001E | FSSS FDDD`

Operands dst, src

Description This instruction moves the content of the source (src) register to the destination (dst) register:

- When the destination (dst) register is an accumulator:
  - The 40-bit move operation is performed in the D-unit ALU.
  - During the 40-bit move operation, an overflow is detected according to M40:
    - the destination accumulator overflow status bit (ACOVx) is set.
    - the destination register (ACx) is saturated according to SATD.
  - If the source (src) register is an auxiliary or temporary register, the 16 LSBs of the source register are sign extended to 40 bits according to SXMD.
- When the destination (dst) register is an auxiliary or temporary register:
  - The 16-bit move operation is performed in the A-unit ALU.
  - If the source (src) register is an accumulator, the 16 LSBs of the accumulator are used to perform the operation.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Move Accumulator Content to Auxiliary or Temporary Register
- Move Auxiliary or Temporary Register Content to Accumulator
- Move Auxiliary or Temporary Register Content to CPU Register
- Move Extended Auxiliary Register Content

Example

Syntax	Description
AC1 = AC0	The content of AC0 is copied to AC1. Because an overflow occurred, ACOV1 is set to 1.

```
Before                              After
AC0          01 E500 0030           AC0          01 E500 0030
AC1          00 2800 0200           AC1          01 E500 0030
M40                     0           M40                     0
SATD                    0           SATD                    0
ACOV1                   0           ACOV1                   1
```

MOV *Move Auxiliary or Temporary Register Content to Accumulator*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**HI(**ACx**)** = TAx	Yes	2	1	X

Opcode `| 0101 001E | FSSS 00DD`

Operands ACx, TAx

Description This instruction moves the content of the auxiliary or temporary register (TAx) to the high part of the accumulator, ACx(31−16):

- ❑ The 16-bit move operation is performed in the D-unit ALU.
- ❑ During the 16-bit move operation, an overflow is detected according to M40:
  - ■ the destination accumulator overflow status bit (ACOVx) is set.
  - ■ the destination register (ACx) is saturated according to SATD.
- ❑ If the source (src) register is an auxiliary or temporary register, the 16 LSBs of the source register are sign extended to 40 bits according to SXMD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SATD, SXMD
Affects	ACOVx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Move Accumulator Content to Auxiliary or Temporary Register
- ❑ Move Accumulator, Auxiliary, or Temporary Register Content
- ❑ Move Auxiliary or Temporary Register Content to CPU Register
- ❑ Move Extended Auxiliary Register Content

Example

Syntax	Description
HI(AC0) = T0	The content of T0 is copied to AC0(31–16).

MOV *Move Auxiliary or Temporary Register Content to CPU Register*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**BRC0** = TAx	Yes	2	1	X
[2]	**BRC1** = TAx	Yes	2	1	X
[3]	**CDP** = TAx	Yes	2	1	X
[4]	**CSR** = TAx	Yes	2	1	X
[5]	**SP** = TAx	Yes	2	1	X
[6]	**SSP** = TAx	Yes	2	1	X

Opcode See Table 5−3 (page 5-258).

Operands TAx

Description This instruction moves the content of the auxiliary or temporary register (TAx) to the selected CPU register. All the move operations are performed in the execute phase of the pipeline and the A-unit ALU is used to transfer the content of the registers.

There is a 3-cycle latency between SP, SSP, CDP, TAx, CSR, and BRCx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

For instruction [2] when BRC1 is loaded with the content of TAx, the block repeat save register (BRS1) is also loaded with the same value.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Move Accumulator Content to Auxiliary or Temporary Register

❑ Move Accumulator, Auxiliary, or Temporary Register Content

❑ Move Auxiliary or Temporary Register Content to Accumulator

❑ Move CPU Register Content to Auxiliary or Temporary Register

❑ Move Extended Auxiliary Register Content

Example

Syntax	Description
BRC1 = T1	The content of T1 is copied to the block repeat register (BRC1) and to the block repeat save register (BRS1).

```
Before                  After
T1        0034          T1        0034
BRC1      00EA          BRC1      0034
BRS1      00EA          BRS1      0034
```

Table 5−3. Opcodes for Move Auxiliary or Temporary Register Content to CPU Register Instruction

No.	Syntax	Opcode
[1]	**BRC0** = TAx	`0101 001E FSSS 1110`
[2]	**BRC1** = TAx	`0101 001E FSSS 1101`
[3]	**CDP** = TAx	`0101 001E FSSS 1010`
[4]	**CSR** = TAx	`0101 001E FSSS 1100`
[5]	**SP** = TAx	`0101 001E FSSS 1000`
[6]	**SSP** = TAx	`0101 001E FSSS 1001`

MOV *Move CPU Register Content to Auxiliary or Temporary Register*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TAx = **BRC0**	Yes	2	1	X
[2]	TAx = **BRC1**	Yes	2	1	X
[3]	TAx = **CDP**	Yes	2	1	X
[4]	TAx = **SP**	Yes	2	1	X
[5]	TAx = **SSP**	Yes	2	1	X
[6]	TAx = **RPTC**	Yes	2	1	X

Opcode See Table 5−4 (page 5-260).

Operands TAx

Description This instruction moves the content of the selected CPU register to the auxiliary or temporary register (TAx). All the move operations are performed in the execute phase of the pipeline and the A-unit ALU is used to transfer the content of the registers.

For instructions [1] and [2], BRCx is decremented in the address phase of the last instruction of a loop. These instructions have a 3-cycle latency requirement versus the last instruction of a loop.

For instructions [3], [4], and [5], there is a 3-cycle latency between SP, SSP, CDP, and TAx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

Status Bits

Affected by	none
Affects	none

Repeat Instruction [6] cannot be repeated; all other instructions can be repeated.

See Also See the following other related instructions:

- ❑ Move Accumulator Content to Auxiliary or Temporary Register
- ❑ Move Auxiliary or Temporary Register Content to CPU Register
- ❑ Store CPU Register Content to Memory

Example

Syntax	Description
T1 = BRC1	The content of block repeat register (BRC1) is copied to T1.

```
Before                      After
T1              0034        T1              00EA
BRC1            00EA        BRC1            00EA
```

Table 5−4. Opcodes for Move CPU Register Content to Auxiliary or Temporary Register Instruction

No.	Syntax	Opcode
[1]	TAx = **BRC0**	`0100 010E 1100 FDDD`
[2]	TAx = **BRC1**	`0100 010E 1101 FDDD`
[3]	TAx = **CDP**	`0100 010E 1010 FDDD`
[4]	TAx = **SP**	`0100 010E 1000 FDDD`
[5]	TAx = **SSP**	`0100 010E 1001 FDDD`
[6]	TAx = **RPTC**	`0100 010E 1110 FDDD`

MOV *Move Extended Auxiliary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	xdst = xsrc	No	2	1	X

Opcode `1001 0000 XSSS XDDD`

Operands xdst, xsrc

Description This instruction moves the content of the source register (xsrc) to the destination register (xdst):

- ❑ When the destination register (xdst) is an accumulator (ACx) and the source register (xsrc) is a 23-bit register (XARx, XSP, XSSP, XDP, or XCDP):
  - ■ The 23-bit move operation is performed in the D-unit ALU.
  - ■ The upper bits of ACx are filled with 0.
- ❑ When the source register (xsrc) is an accumulator (ACx) and the destination register (xdst) is a 23-bit register (XARx, XSP, XSSP, XDP, or XCDP):
  - ■ The 23-bit move operation is performed in the A-unit ALU.
  - ■ The lower 23 bits of ACx are loaded into xdst.
- ❑ When both the source register (xsrc) and the destination register (xdst) are accumulators, the Move Accumulator Content instruction (dst = src) is assembled.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Load Extended Auxiliary Register from Memory
- ❑ Load Extended Auxiliary Register with Immediate Value
- ❑ Modify Extended Auxiliary Register Content
- ❑ Store Extended Auxiliary Register Content to Memory

Example

Syntax	Description
XAR1 = AC0	The lower 23 bits of AC0 are loaded into XAR1.

MOV *Move Memory to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = **coef(**Cmem**)**	No	3	1	X
[2]	**coef(**Cmem**)** = Smem	No	3	1	X
[3]	Lmem = **dbl(coef(**Cmem**))**	No	3	1	X
[4]	**dbl(coef(**Cmem**))** = Lmem	No	3	1	X
[5]	**dbl(**Ymem**)** = **dbl(**Xmem**)**	No	3	1	X
[6]	Ymem = Xmem	No	3	1	X

Description

These instructions store the content of a memory location to a memory location. They use a dedicated datapath to perform the operation.

Status Bits

Affected by none

Affects none

See Also

See the following other related instructions:

❑ Store Accumulator Content to Memory

❑ Store Accumulator, Auxiliary, or Temporary Register Content to Memory

❑ Store Auxiliary or Temporary Register Pair Content to Memory

❑ Store CPU Register Content to Memory

❑ Store Extended Auxiliary Register Content to Memory

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = **coef(**Cmem**)**	No	3	1	X

Opcode

```
|1110  1111|AAAA  AAAI|xxxx  00mm
```

Operands Cmem, Smem

Description This instruction stores the content of a data memory operand Cmem, addressed using the coefficient addressing mode, to a memory (Smem) location.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*(#0500h) = coef(*CDP)	The content addressed by the coefficient data pointer register (CDP) is copied to address 0500h.

```
Before                 After
*CDP         3400      *CDP         3400
500          0000      500          3400
```

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**coef(**Cmem**)** = Smem	No	3	1	X

Opcode `| 1110 1111 | AAAA AAAI | xxxx 01mm`

Operands Cmem, Smem

Description This instruction stores the content of a memory (Smem) location to a data memory (Cmem) location addressed using the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
coef(*CDP) = *AR3	The content addressed by AR3 is copied in the location addressed by the coefficient data pointer register (CDP).

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	Lmem = **dbl(coef(**Cmem**))**	No	3	1	X

Opcode | 1110 1111 | AAAA AAAI | xxxx 10mm

Operands Cmem, Lmem

Description This instruction stores the content of two consecutive data memory (Cmem) locations, addressed using the coefficient addressing mode, to two consecutive data memory (Lmem) locations.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AR1 = dbl(coef((CDP + T0)))	The content (long word) addressed by the coefficient data pointer register (CDP) and CDP + 1 is copied in the location addressed by AR1 and AR1 + 1, respectively. After the memory store, CDP is incremented by the content of T0 (5).

```
Before                  After
T0          0005        T0          0005
CDP         0200        CDP         0205
AR1         0300        AR1         0300
200         3400        200         3400
201         0FD3        201         0FD3
300         0000        300         3400
301         0000        301         0FD3
```

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	**dbl(coef(**Cmem**))** = Lmem	No	3	1	X

Opcode | 1110 1111 | AAAA AAAI | xxxx 11mm

Operands Cmem, Lmem

Description This instruction stores the content of two consecutive data memory (Lmem) locations to two consecutive data memory (Cmem) locations addressed using the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus. On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
dbl(coef(*CDP)) = *AR3+	The content (long word) addressed by AR3 and AR3 + 1 is copied in the location addressed by the coefficient data pointer register (CDP) and CDP + 1, respectively. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	**dbl(**Ymem**)** = **dbl(**Xmem**)**	No	3	1	X

Opcode `1000 0000 | XXXM MMYY | YMMM 00xx`

Operands Xmem, Ymem

Description This instruction stores the content of two consecutive data memory (Xmem) locations, addressed using the dual addressing mode, to two consecutive data memory (Ymem) locations.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
dbl(*AR1) = dbl(*AR0)	The content addressed by AR0 is copied in the location addressed by AR1 and the content addressed by AR0 + 1 is copied in the location addressed by AR1 + 1.

```
Before                  After
AR0         0300        AR0         0300
AR1         0400        AR1         0400
300         3400        300         3400
301         0FD3        301         0FD3
400         0000        400         3400
401         0000        401         0FD3
```

Move Memory to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	Ymem = Xmem	No	3	1	X

Opcode `|1000 0000|XXXM MMYY|YMMM 01xx`

Operands Xmem, Ymem

Description This instruction stores the content of data memory (Xmem) location, addressed using the dual addressing mode, to data memory (Ymem) location.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = *AR5	The content addressed by AR5 is copied in the location addressed by AR3.

MPY *Multiply*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy * ACx)	Yes	2	1	X
[2]	ACy = rnd(ACx * Tx)	Yes	2	1	X
[3]	ACy = rnd(ACx * K8)	Yes	3	1	X
[4]	ACy = rnd(ACx * K16)	No	4	1	X
[5]	ACx = rnd(Smem * **coef(**Cmem**)**)[, T3 = Smem]	No	3	1	X
[6]	ACy = rnd(Smem * ACx)[, T3 = Smem]	No	3	1	X
[7]	ACx = rnd(Smem * K8)[, T3 = Smem]	No	4	1	X
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	No	4	1	X
[9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	No	3	1	X
[10]	ACx = rnd(Smem * uns(**coef(**Cmem**)**))	No	3	1	X

Description These instructions perform a multiplication in the D-unit MAC.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply
- ❑ Multiply and Accumulate
- ❑ Multiply and Accumulate with Parallel Multiply
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply with Parallel Multiply and Accumulate
- ❑ Multiply with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiplies
- ❑ Square

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy * ACx)	Yes	2	1	X

Opcode `0101 010E DDSS 011%`

Operands ACx, ACy

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and ACy(32−16).

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 * AC0	The product of the content of AC1 and the content of AC0 is stored in AC1.

```
Before                          After
AC0        02 6000 3400         AC0        02 6000 3400
AC1        00 C000 0000         AC1        00 4800 0000
M40                   1         M40                   1
FRCT                  0         FRCT                  0
ACOV1                 0         ACOV1                 0
```

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACx * Tx)	Yes	2	1	X

Opcode `0101 100E | DDSS ss0%`

Operands ACx, ACy, Tx

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of Tx, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * T0	The product of the content of AC1 and the content of T0 is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACx * K8)	Yes	3	1	X

Opcode | 0001 111E | KKKK KKKK | SSDD xx0% |

Operands ACx, ACy, K8

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the 8-bit signed constant, K8, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits		
	Affected by	FRCT, M40, RDM
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * #−2	The product of the content of AC1 and a signed 8-bit value (−2) is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx * K16)	No	4	1	X

Opcode | `0111 1001 | KKKK KKKK | KKKK KKKK | SSDD xx0%`

Operands ACx, ACy, K16

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the 16-bit signed constant, K16, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * #−64	The product of the content of AC1 and a signed 16-bit value (−64) is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = rnd(Smem * **coef(**Cmem**))**[, T3 = Smem]	No	3	1	X

Opcode

```
|1101  0001|AAAA  AAAI|U%DD  00mm
```

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * coef(*CDP)	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = rnd(Smem * ACx) [,T3 = Smem]	No	3	1	X

Opcode `| 1101 0011 | AAAA AAAI | U%DD 00SS`

Operands ACx, ACy, Smem

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of a memory location (Smem), sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * AC1	The product of the content addressed by AR3 and the content of AC1 is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACx = rnd(Smem * K8) [,T3 = Smem]	No	4	1	X

Opcode | `1111 1000 | AAAA AAAI | KKKK KKKK | xxDD x0U%`

Operands ACx, K8, Smem

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * #−2	The product of the content addressed by AR3 and a signed 8-bit value (−2) is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	No	4	1	X

Opcode `|1000 0110|XXXM MMYY|YMMM xxDD|000g uuU%`

Operands ACx, Xmem, Ymem

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ❑ Input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(*AR3) * uns(*AR4)	The product of the unsigned content addressed by AR3 and the unsigned content addressed by AR4 is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]	No	3	1	X

Opcode `| 1101 0011 | AAAA AAAI | U%DD u1ss`

Operands ACx, Smem, Tx

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory location (Smem), sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the instruction, the 32-bit result is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the instruction, the 32-bit result is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = uns(T0 * *AR3)	The unsigned product of the content addressed by AR3 and the content of T0 is stored in AC0.

Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACx = rnd(Smem * uns(**coef(**Cmem**)**))	No	3	1	X

Opcode

```
1101  0000 | AAAA  AAAI | 0%DD  01mm
```

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits.

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result of a double precision multiplication and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	**Description**
AC0 = *AR3− * uns(coef(*CDP+))	The product of the content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0. AR3 is decremented by 1 and CDP is incremented by 1.

```
Execution
rnd((Smem)[16:0]*uns(Cmem)[16:0]) -> ACx

Before                          After
AC0         FF 8000 0000        AC0         FF FF00 0000
XAR3             00 1001        XAR3             00 1000
Data memory
1001h               FE00        1001h               FE00
XCDP             00 2000        XCDP             00 2000
Coeff memory
2000h               8000        2000h               8000
```

MPY::MAC *Multiply with Parallel Multiply and Accumulate*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe-line
[1]	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[2]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[3]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[4]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X

Description These instructions perform two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Multiply
- ❑ Multiply and Accumulate
- ❑ Parallel Multiply and Accumulates

Multiply With Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(**coef(**Cmem**)**))), ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode | `1000 0100` | `XXXM MMYY` | `YMMM 10mm` | `uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ❑ Input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).

- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	**Description**

AC0 = uns(*AR3) * uns(coef(*CDP)),
AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

Multiply with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx + **(**uns(Smem) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode

```
|1111  1101|AAAA  AAAI|0000  01mm|DDDD  uug%
```

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = AC0 + (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                          After
AC0          00 0000 8000       AC0          00 3F80 8000
XAR3              00 10FF       XAR3              00 10FE
Data memory
10FFh                FE00       10FFh                FE00
XCDP              00 2000       XCDP              00 2002
Coeff memory
2001h                4000       2001h                4000

AC1          FF 8000 0000       AC1          00 7F00 0000
Coeff memory
2000h                8000       2000h                8000
```

Multiply with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe-line
[3]	ACy = M40(rnd(uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**)))**))), ACx = M40(rnd(ACx + **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0100 01mm | DDDD uug%

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(HI(*AR3−)) * uns(HI(coef(*CDP+))),
AC0 = AC0 + (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         00 3F80 8000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Multiply With Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx + uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001 0010|XXXM MMYY|YMMM 01mm|uuDD DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = AC0 + (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(ACx + uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

```
Before                         After
AC0         00 0000 8000       AC0         00 3F80 8000
XAR2             00 10FE       XAR2             00 10FD
XAR3             00 20FE       XAR3             00 20FD
Data memory
10FEh               FE00       10FEh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
20FEh               FF00       20FFh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MPY::MAS *Multiply With Parallel Multiply and Subtract*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe-line
[1]	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[2]	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[3]	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X

Description These instructions perform two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Multiply
- ❑ Multiply and Subtract
- ❑ Parallel Multiply and Subtract

Multiply with Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − **(**uns(Smem) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0000 11mm | DDDD uug%

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = AC0 − (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx−M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         FF C080 8000
XAR3             00 10FF       XAR3             00 10FE
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F00 0000
Coeff memory
2000h               8000       2000h               8000
```

Multiply with Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipe-line
[2]	ACy = M40(rnd(uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)**)**))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0100 11mm | DDDD uug%

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(HI(*AR3−)) * uns(HI(coef(*CDP+))),
AC0 = AC0 − (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx–M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         FF C080 8000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Multiply with Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001  0010|XXXM  MMYY|YMMM  10mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply, and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the contents of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC 1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ❑ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 key word is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = AC0 − (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(ACx - uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

```
Before                         After
AC0         00 0000 8000       AC0         FF C080 8000
XAR2              00 10FE      XAR2              00 10FD
XAR3              00 20FE      XAR3              00 20FD
Data memory
10FEh                FE00      10FEh                FE00
XCDP              00 2000      XCDP              00 2002
Coeff memory
2001h                4000      2001h                4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
20FEh                FF00      20FFh                FF00
Coeff memory
2000h                8000      2000h                8000
```

MPYM::MOV *Multiply with Parallel Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(Tx * Xmem), Ymem = **HI(**ACx **<< T2)** [,T3 = Xmem]	No	4	1	X

Opcode `|1000 0111|XXXM MMYY|YMMM SSDD|000x ssU%`

Operands ACx, ACy, Tx, Xmem, Ymem

Description This instruction performs two operations in parallel: multiply and store.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- ❑ The input operand is shifted in the D-unit shifter according to SXMD.
- ❑ After the shift, the high part of the accumulator, ACx(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = rnd(Tx * Xmem),
  Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem]

- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = rnd(Tx * Xmem),
  Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem]

Status Bits

Affected by	C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD
Affects	ACOVy

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- ❑ Addition with Parallel Store Accumulator Content to Memory
- ❑ Multiply
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Store Accumulator Content to Memory
- ❑ Subtraction with Parallel Store Accumulator Content to Memory

Example

Syntax	Description
AC1 = rnd(T0 * *AR0+), *AR1+ = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR0 is multiplied by the content of T0. Since FRCT = 1, the result is multiplied by 2, rounded, and stored in AC1. The content of AC0 is shifted by the content of T2, and AC0(31−16) is stored at the address of AR1. AR0 and AR1 are both incremented by 1.

Before		**After**	
AC0	FF 8421 1234	AC0	FF 8421 1234
AC1	00 0000 0000	AC1	00 2000 0000
AR0	0200	AR0	0201
AR1	0300	AR1	0301
T0	4000	T0	4000
T2	0004	T2	0004
200	4000	200	4000
300	1111	300	4211
FRCT	1	FRCT	1
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0

MAC *Multiply and Accumulate (MAC)*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + **(**ACx * Tx**)**)	Yes	2	1	X
[2]	ACy = rnd(**(**ACy * Tx**)** + ACx)	Yes	2	1	X
[3]	ACy = rnd(ACx + **(**Tx * K8**)**)	Yes	3	1	X
[4]	ACy = rnd(ACx + **(**Tx * K16**)**)	No	4	1	X
[5]	ACx = rnd(ACx + **(**Smem * **coef(**Cmem**))**)[, T3 = Smem]	No	3	1	X
[6]	ACy = rnd(ACy + **(**Smem * ACx**)**)[, T3 = Smem]	No	3	1	X
[7]	ACy = rnd(ACx + **(**Tx * Smem**)**)[, T3 = Smem]	No	3	1	X
[8]	ACy = rnd(ACx + **(**Smem * K8**)**)[, T3 = Smem]	No	4	1	X
[9]	ACy = M40(rnd(ACx + **(**uns(Xmem) * uns(Ymem))**)** [, T3 = Xmem]	No	4	1	X
[10]	ACy = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(Ymem))**)** [, T3 = Xmem]	No	4	1	X
[11]	ACx = rnd(ACx + **(**Smem * uns(**coef(**Cmem**)))**)	No	3	1	X

Description These instructions perform a multiplication and an accumulation in the D-unit MAC.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

See Also See the following other related instructions:

❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate

❑ Multiply and Accumulate with Parallel Delay

❑ Multiply and Accumulate with Parallel Load Accumulator from Memory

❑ Multiply and Accumulate with Parallel Multiply

❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory

❑ Multiply and Subtract

❑ Multiply and Subtract with Parallel Multiply and Accumulate

❑ Multiply with Parallel Multiply and Accumulate

❑ Parallel Multiply and Accumulates

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + **(**ACx * Tx**)**)	Yes	2	1	X

Opcode

```
|0101 011E|DDSS ss0%
```

Operands ACx, ACy, Tx

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of Tx, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (AC1 * T0)	The product of the content of AC1 and the content of T0 is added to the content of AC0. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	X

Opcode | 0101 100E | DDSS ss1%

Operands ACx, ACy, Tx

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACy(32−16) and the content of Tx, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = rnd((AC1 * T1) + AC0)	The product of the content of AC1 and the content of T1 is added to the content of AC0. The result is rounded and stored in AC1.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	X

Opcode

```
| 0001  111E | KKKK  KKKK | SSDD  ss1%
```

Operands ACx, ACy, K8, Tx

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (T0 * K8)	The product of the content of T0 and a signed 8-bit value is added to the content of AC1. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx + **(**Tx * K16**)**)	No	4	1	X

Opcode

```
| 0111  1001 | KKKK  KKKK | KKKK  KKKK | SSDD  ss1% 
```

Operands ACx, ACy, K16, Tx

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the 16-bit signed constant, K16, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits		
	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (T0 * #FFFFh)	The product of the content of T0 and a signed 16-bit value (FFFFh) is added to the content of AC1. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACx = rnd(ACx + **(**Smem * **coef(**Cmem**)))**[, T3 = Smem]	No	3	1	X

Opcode `1101 0001 | AAAA AAAI | U%DD 01mm`

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC2 = rnd(AC2 + (*AR1 * coef(*CDP)))	The product of the content addressed by AR1 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC2. The result is rounded and stored in AC2. The result generated an overflow.

```
Before                          After
AC2         00 EC00 0000        AC2         00 EC00 0000
AR1                 0302        AR2                 0302
CDP                 0202        CDP                 0202
302                 FE00        302                 FE00
202                 0040        202                 0040
ACOV2                  0        ACOV2                  1
```

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = rnd(ACy + **(**Smem * ACx**)**)[, T3 = Smem]	No	3	1	X

Opcode | 1101 0010 | AAAA AAAI | U%DD 00SS

Operands ACx, ACy, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of a memory location (Smem), sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 + (*AR3 * AC0)	The product of the content addressed by AR3 and the content of AC0 is added to the content of AC1. The result is stored in AC1.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	No	3	1	X

Opcode `|1101 0100|AAAA AAAI|U%DD ssSS`

Operands ACx, ACy, Smem, Tx

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory location (Smem), sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (T0 * *AR3)	The product of the content addressed by AR3 and the content of T0 is added to the content of AC1. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = rnd(ACx + **(**Smem * K8**)**)[, T3 = Smem]	No	4	1	X

Opcode `| 1111 1000 | AAAA AAAI | KKKK KKKK | SSDD x1U%`

Operands ACx, ACy, K8, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.

❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (*AR3 * #FFh)	The product of the content addressed by AR3 and a signed 8-bit value (FFh) is added to the content of AC1. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = M40(rnd(ACx + **(**uns(Xmem) * uns(Ymem))**)** [, T3 = Xmem]	No	4	1	X

Opcode `|1000 0110|XXXM MMYY|YMMM SSDD|001g uuU%`

Operands ACx, ACy, Xmem, Ymem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

❑ Input operands are extended to 17 bits according to uns.

- If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
- If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.

❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVy
Repeat	This instruction can be repeated.	

Example

Syntax	Description
AC3 = rnd(AC3 + (uns(*AR2+) * uns(*AR3+)))	The product of the unsigned content addressed by AR2 and the unsigned content addressed by AR3 is added to the content of AC3. The result is rounded and stored in AC3. The result generated an overflow. AR2 and AR3 are both incremented by 1.

```
Before                              After
AC3          00 2300 EC00          AC3          00 9221 0000
AR2                   302          AR2                   303
AR3                   202          AR3                   203
ACOV3                   0          ACOV3                   1
302                  FE00          302                  FE00
202                  7000          202                  7000
M40                     0          M40                     0
SATD                    0          SATD                    0
FRCT                    0          FRCT                    0
```

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = M40(rnd((ACx **>> #16)** + **(**uns(Xmem) * uns(Ymem)))**)** [, T3 = Xmem]	No	4	1	X

Opcode `1000 0110 | XXXM MMYY | YMMM SSDD | 010g uuU%`

Operands ACx, ACy, Xmem, Ymem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ❑ Input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = (AC1 >> #16) + (uns(*AR3) * uns(*AR4))	The product of the unsigned content addressed by AR3 and the unsigned content addressed by AR4 is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC0.

Multiply and Accumulate (MAC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACx = rnd(ACx + **(**Smem * uns(**coef(**Cmem**))**))	No	3	1	X

Opcode

```
|1101  0000|AAAA  AAAI|0%DD  10mm
```

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result and accumulation to the other partial result of double precision multiplication, and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (*AR3− * uns(coef(*CDP+)))	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP in incremented by 1.

```
Execution
rnd(ACx+(Smem)[16:0]*uns(Cmem)[16:0]) -> ACx

Before                          After
AC0         00 0000 8000        AC0         FF FF00 8000
XAR3             00 1001        XAR3             00 1000
Data memory
1001h               FE00        1001h               FE00
XCDP             00 2000        XCDP             00 2001
Coeff memory
2000h               8000        2000h               8000
```

MACMZ *Multiply and Accumulate with Parallel Delay*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx + **(**Smem * **coef(**Cmem**)))**[, T3 = Smem]**,** **delay(**Smem**)**	No	3	1	X

Opcode `| 1101 0000 | AAAA AAAI | U%DD xxmm`

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC in parallel with the delay memory instruction. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.

❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.

❑ When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

The soft dual memory addressing mode mechanism cannot be applied to this instruction. This instruction cannot use the *port(#k16) addressing mode or be paralleled with the readport() or writeport() operand qualifier.

This instruction cannot be used for accesses to I/O space. Any illegal access to I/O space generates a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 set to 0, compatibility is ensured.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

Repeat

This instruction can be repeated.

See Also

See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Multiply and Accumulate
- ❑ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❑ Multiply and Accumulate with Parallel Multiply
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply with Parallel Multiply and Accumulate
- ❑ Parallel Multiply and Accumulates

Example

Syntax	Description
AC0 = AC0 + (*AR3 * coef(*CDP)), delay(*AR3)	The product of the content addressed by AR3 and the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The content addressed by AR3 is copied into the next higher address.

MACM::MOV *Multiply and Accumulate with Parallel Load Accumulator from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx + **(**Tx * Xmem**)**), ACy = Ymem **<< #16** [,T3 = Xmem]	No	4	1	X

Opcode `|1000 0110|XXXM MMYY|YMMM DDDD|101x ssU%`

Operands ACx, ACy, Tx, Xmem, Ymem

Description This instruction performs two operations in parallel: multiply and accumulate (MAC) and load.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ When an addition overflow is detected, the accumulator is saturated according to SATD.
- ❑ This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation loads the content of data memory operand Ymem, which has been shifted to the left by 16 bits, into accumulator ACy.

- ❑ The input operand is sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ The input operand is shifted to the left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Multiply and Accumulate
- ❑ Multiply and Accumulate with Parallel Delay
- ❑ Multiply and Accumulate with Parallel Multiply
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply with Parallel Multiply and Accumulate
- ❑ Parallel Multiply and Accumulates

Example

Syntax	Description

AC0 = AC0 + (T0 * *AR3),
AC1 = *AR4 << #16
	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is added to the content of AC0. The result is stored in AC0. The content addressed by AR4, which has been shifted to the left by 16 bits, is stored in AC1.

MAC::MPY *Multiply and Accumulate with Parallel Multiply*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	No	4	1	X
[2]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	X
[3]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))	No	4	1	X
[4]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	No	4	1	X
[5]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))	No	4	1	X
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X

Description

These instructions perform two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

See Also

See the following other related instructions:

❏ Modify Auxiliary Register Content with Parallel Multiply and Accumulate

❏ Multiply and Accumulate

❏ Multiply and Accumulate with Parallel Delay

❏ Multiply and Accumulate with Parallel Load Accumulator from Memory

❏ Multiply and Accumulate with Parallel Store Accumulator Content to Memory

❏ Multiply and Subtract with Parallel Multiply

❏ Multiply with Parallel Multiply and Accumulate

❏ Parallel Multiply and Accumulates

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**)**)))	No	4	1	X

Opcode `1000 0010 | XXXM MMYY | YMMM 01mm | uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	**Description**

AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))),
AC1 = uns(*AR4) * uns(coef(*CDP))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy + **(**uns(Smem) * uns(**HI(coef(**Cmem**))))**)), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode | `1111 1101` | `AAAA AAAI` | `0000 10mm` | `DDDD uug%`

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = uns(*AR3−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                          After
AC0         FF 8000 0000        AC0         00 3F80 0000
XAR3             00 10FF        XAR3             00 10FE
Data memory
10FFh               FE00        10FFh               FE00
XCDP             00 2000        XCDP             00 2002
Coeff memory
2001h               4000        2001h               4000

AC1         00 0000 8000        AC1         00 7F00 8000
Coeff memory
2000h               8000        2000h               8000
```

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Smem) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode `| 1111 1101 | AAAA AAAI | 0010 10mm | DDDD uug%`

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = uns(*AR3−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
(ACy>>#16)+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FF       XAR3             00 10FE
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0800 0000       AC1         00 7F00 0800
Coeff memory
2000h               8000       2000h               8000
```

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy + **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode

```
|1111 1101|AAAA AAAI|0100 10mm|DDDD uug%
```

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = uns(LO(*AR3−)) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         00 7F80 8000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd((ACy**>>#16**) + **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0110 10mm | DDDD uug%

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = uns(LO(*AR3−)) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
(ACy>>#16)+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before                            After
AC0          FF 8000 0000         AC0          00 3F80 0000
XAR3              00 10FE         XAR3              00 10FC
Data memory
10FFh                FE00         10FFh                FE00
XCDP              00 2000         XCDP              00 2002
Coeff memory
2001h                4000         2001h                4000

AC1          00 0800 0000         AC1          00 7F80 0800
Data memory
10FEh                FF00         10FEh                FF00
Coeff memory
2000h                8000         2000h                8000
```

Multiply and Accumulate With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode `|1001 0100|XXXM MMYY|YMMM 10mm|uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ❑ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = uns(*AR2−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(uns(Xmem)[16:0]  * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd((ACy >> #16) + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before		**After**	
AC0	FF 8000 0000	AC0	00 3F80 0000
XAR2	00 10FE	XAR2	00 10FD
XAR3	00 20FE	XAR3	00 20FD
Data memory			
10FEh	FE00	10FEh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0800 0000	AC1	00 7F80 8000
Data memory			
20FEh	FF00	20FFh	FF00
Coeff memory			
2000h	8000	2000h	8000

MAC::MAS *Multiply and Accumulate With Parallel Multiply and Subtract*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[2]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[3]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[4]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5	1	X

Description These instructions perform two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

See Also

See the following other related instructions:

- Modify Auxiliary Register Content with Parallel Multiply and Subtract
- Multiply and Subtract
- Multiply and Subtract with Parallel Load Accumulator from Memory
- Multiply and Subtract with Parallel Store Accumulator Content to Memory
- Multiply and Subtract with Parallel Multiply
- Parallel Multiply and Subtracts

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = M40(rnd(ACy + **(**uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − **(**uns(Smem) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0001 10mm | DDDD uug%

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx−M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0          00 0000 8000      AC0          FF C080 8000
XAR3              00 10FF      XAR3              00 10FE
Data memory
10FFh                FE00      10FFh                FE00
XCDP              00 2000      XCDP              00 2002
Coeff memory
2001h                4000      2001h                4000

AC1          00 0000 8000      AC1          00 7F00 8000
Coeff memory
2000h                8000      2000h                8000
```

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Smem) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(ACx − **(**uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode `|1111 1101|AAAA AAAI|0010 01mm|DDDD uug%`

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
(ACy>>#16)+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx−M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before                        After
AC0         00 0000 8000      AC0         FF C080 8000
XAR3             00 10FF      XAR3             00 10FE
Data memory
10FFh               FE00      10FFh               FE00
XCDP             00 2000      XCDP             00 2002
Coeff memory
2001h               4000      2001h               4000

AC1         00 0800 0000      AC1         00 7F00 0800
Coeff memory
2000h               8000      2000h               8000
```

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy + **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**)))**))), ACx = M40(rnd(ACx − **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X

Opcode `1111 1101 | AAAA AAAI | 0101 10mm | DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0]))  -> ACy
ACx–M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0]))  -> ACx
```

Before		**After**	
AC0	00 0000 8000	AC0	FF C080 8000
XAR3	00 10FE	XAR3	00 10FC
Data memory			
10FFh	FE00	10FFh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0000 8000	AC1	00 7F80 0000
Data memory			
10FEh	FF00	10FEh	FF00
Coeff memory			
2000h	8000	2000h	8000

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode `|1111 1101|AAAA AAAI|0110 01mm|DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1>>#16) + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
(ACy>>#16)+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx-M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before                          After
AC0          00 0000 8000       AC0          FF C080 8000
XAR3              00 10FE       XAR3              00 10FC
Data memory
10FFh                FE00       10FFh                FE00
XCDP              00 2000       XCDP              00 2002
Coeff memory
2001h                4000       2001h                4000

AC1          00 0800 0000       AC1          00 7F80 0800
Data memory
10FEh                FF00       10FEh                FF00
Coeff memory
2000h                8000       2000h                8000
```

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACy + uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode | 1001 0011 | XXXM MMYY | YMMM 01mm | uuDD DDg%

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ❑ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(ACx – uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(ACy + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before		**After**	
AC0	00 0000 8000	AC0	FF C080 8000
XAR2	00 10FE	XAR2	00 10FD
XAR3	00 20FE	XAR3	00 20FD
Data memory			
10FEh	FE00	10FEh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0000 8000	AC1	00 7F80 8000
Data memory			
20FEh	FF00	20FFh	FF00
Coeff memory			
2000h	8000	2000h	8000

Multiply and Accumulate With Parallel Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001  0100|XXXM  MMYY|YMMM  00mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC) and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ❑ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB bus are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(ACx – uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd((ACy >> #16) + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy

Before                         After
AC0         00 0000 8000       AC0         FF C080 8000
XAR2             00 10FE       XAR2             00 10FD
XAR3             00 20FE       XAR3             00 20FD
Data memory
10FEh               FE00       10FEh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0800 0000       AC1         00 7F80 8000
Data memory
20FEh               FF00       20FFh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MACM::MOV *Multiply and Accumulate with Parallel Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + **(**Tx * Xmem**)**), Ymem = **HI(**ACx **<< T2)** [,T3 = Xmem]	No	4	1	X

Opcode `|1000 0111|XXXM MMYY|YMMM SSDD|001x ssU%`

Operands ACx, ACy, Tx, Xmem, Ymem

Description This instruction performs two operations in parallel: multiply and accumulate (MAC) and store.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.
- This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACx(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
    ACy = rnd(ACy + (Tx * Xmem)),
    Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem]

❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
    ACy = rnd(ACy + (Tx * Xmem)),
    Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem]

Status Bits

Affected by	C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD
Affects	ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate

❑ Multiply and Accumulate

❑ Multiply and Accumulate with Parallel Delay

❑ Multiply and Accumulate with Parallel Load Accumulator from Memory

❑ Multiply and Accumulate with Parallel Multiply

❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory

❑ Multiply with Parallel Multiply and Accumulate

❑ Parallel Multiply and Accumulates

Example

Syntax	Description
AC0 = AC0 + (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is added to the content of AC0. The result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31−16) is stored at the address of AR4.

MAS *Multiply and Subtract*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – **(**ACx * Tx**)**)	Yes	2	1	X
[2]	ACx = rnd(ACx – **(**Smem * **coef(**Cmem**))**)[, T3 = Smem]	No	3	1	X
[3]	ACy = rnd(ACy – **(**Smem * ACx**)**)[, T3 = Smem]	No	3	1	X
[4]	ACy = rnd(ACx – **(**Tx * Smem**)**)[, T3 = Smem]	No	3	1	X
[5]	ACy = M40(rnd(ACx – **(**uns(Xmem) * uns(Ymem)))**)** [, T3 = Xmem]	No	4	1	X
[6]	ACx = rnd(ACx − **(**Smem * uns(**coef(**Cmem)))**)**)	No	3	1	X

Description These instructions perform a multiplication and a subtraction in the D-unit MAC.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Accumulate
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiply and Subtracts

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – (ACx * Tx))	Yes	2	1	X

Opcode | 0101 011E | DDSS ss1%

Operands ACx, ACy, Tx

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of Tx, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = rnd(AC1 – (AC0 * T1))	The product of the content of AC0 and the content of T1 is subtracted from the content of AC1. The result is rounded and stored in AC1.

Before		**After**	
AC0	00 EC00 0000	AC0	00 EC00 0000
AC1	00 3400 0000	AC1	00 1680 0000
T1	2000	T1	2000
M40	0	M40	0
ACOV1	0	ACOV1	0
FRCT	0	FRCT	0

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = rnd(ACx – **(**Smem * **coef(**Cmem**))**)[, T3 = Smem]	No	3	1	X

Opcode

```
|1101  0001|AAAA  AAAI|U%DD  10mm
```

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of a memory location (Smem), sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC2 = rnd(AC2 – (*AR1 * coef(*CDP)))	The product of the content addressed by AR1 and the content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC2. The result is rounded and stored in AC2.

```
Before                          After
AC2         00 EC00 0000        AC2         00 EC01 0000
AR1                 0302        AR2                 0302
CDP                 0202        CDP                 0202
302                 FE00        302                 FE00
202                 0040        202                 0040
ACOV2                  0        ACOV2                  1
SATD                   0        SATD                   0
RDM                    0        RDM                    0
FRCT                   0        FRCT                   0
```

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = rnd(ACy – **(**Smem * ACx**)**)[, T3 = Smem]	No	3	1	X

Opcode

```
|1101  0010|AAAA  AAAI|U%DD  01SS
```

Operands ACx, ACy, Smem

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32−16) and the content of a memory location (Smem), sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (*AR3 * AC1)	The product of the content addressed by AR3 and the content of AC1 is subtracted from the content of AC0. The result is stored in AC0.

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = rnd(ACx – (Tx * Smem))[, T3 = Smem]	No	3	1	X

Opcode

```
|1101  0101|AAAA  AAAI|U%DD  ssSS
```

Operands ACx, ACy, Smem, Tx

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory location (Smem), sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (T0 * *AR3)	The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC1. The result is stored in AC0.

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACx – **(**uns(Xmem) * uns(Ymem))**)** [, T3 = Xmem]	No	4	1	X

Opcode

```
|1000  0110|XXXM  MMYY|YMMM  SSDD|011g  uuU%
```

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.

- ❑ Input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC3 = AC3 – (uns(*AR2+) * uns(*AR3+))	The product of the unsigned content addressed by AR2 and the unsigned content addressed by AR3 is subtracted from the content of AC3. The result is stored in AC3. AR2 and AR3 are both incremented by 1.

```
Before                           After
AC3          00 2300 EC00        AC3          FF B3E0 EC00
AR2                   302        AR2                   303
AR3                   202        AR3                   203
ACOV3                   0        ACOV3                   0
302                  FE00        302                  FE00
202                  7000        202                  7000
FRCT                    0        FRCT                    0
```

Multiply and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACx = rnd(ACx − **(**Smem * uns(**coef(**Cmem**))**))	No	3	1	X

Opcode | 1101 0000 | AAAA AAAI | 0%DD 11mm

Operands ACx, Cmem, Smem

Description This instruction performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of a data memory location (Smem) and the content of a data memory operand (Cmem).

Note:

The uns keyword is mandatory for this instruction.

The data memory operand Smem is addressed by DAGEN path X by using the Smem addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The another data memory operand Cmem is addressed by DAGEN path C by using the coefficient addressing mode, driven on data bus BDB, and sign extended to 17 bits with filling zeros in the MAC1.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

This instruction can be applied to compute the intermediate multiplication result and subtraction from the other partial result of double precision arithmetic, and to free up one DAGEN operator (DAGEN path Y) for storing an instruction with enabling parallelism.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 − (*AR3− * uns(coef(*CDP+)))	The product of the content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP is incremented by 1.

```
Execution
rnd(ACx+(Smem)[16:0]*uns(Cmem)[16:0]) -> ACx

Before                         After
AC0         00 0000 8000       AC0         00 0100 8000
XAR3             00 1001       XAR3             00 1000
Data memory
1001h               FE00       1001h               FE00
XCDP             00 2000       XCDP             00 2001
Coeff memory
2000h               8000       2000h               8000
```

MASM::MOV *Multiply and Subtract with Parallel Load Accumulator from Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx – **(**Tx * Xmem**)**), ACy = Ymem **<< #16** [,T3 = Xmem]	No	4	1	X

Opcode `1000 0110 | XXXM MMYY | YMMM DDDD | 100x ssU%`

Operands ACx, ACy, Tx, Xmem, Ymem

Description This instruction performs two operations in parallel: multiply and subtract (MAS), and load.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation loads the content of data memory operand Ymem, which has been shifted to the left by 16 bits, into accumulator ACy.

- The input operand is sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- The input operand is shifted to the left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiply and Subtracts

Example

Syntax	**Description**

AC0 = AC0 – (T0 * *AR3),
AC1 = *AR4 << #16
	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC0. The result is stored in AC0. The content addressed by AR4, which has been shifted to the left by 16 bits, is stored in AC1.

MAS::MPY *Multiply and Subtract with Parallel Multiply*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X
[2]	ACy = M40(rnd(ACy − **(**uns(Smem) * uns(**HI(coef(**Cmem**))**)**)**)), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X
[3]	ACy = M40(rnd(ACy − **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)**)**)), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Description These instructions perform two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Accumulate with Parallel Multiply
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiply and Subtracts

Multiply and Subtract With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**)**)))	No	4	1	X

Opcode `|1000 0010|XXXM MMYY|YMMM 10mm|uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = AC0 – (uns(*AR3) * uns(coef(*CDP))),
AC1 = uns(*AR4) * uns(coef(*CDP))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

Multiply and Subtract With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy − **(**uns(Smem) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0001 00mm | DDDD uug%

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = uns(*AR3−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy-M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FF       XAR3             00 10FE
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         FF 8100 8000
Coeff memory
2000h               8000       2000h               8000
```

Multiply and Subtract With Parallel Multiply

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy − **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode `1111 1101 | AAAA AAAI | 0101 00mm | DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = uns(LO(*AR3−)) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy–M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         FF 8080 8000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MAS::MAC *Multiply and Subtract with Parallel Multiply and Accumulate*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(ACy + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X
[2]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X
[3]	ACy = M40(rnd(ACy − **(**uns(Smem) * uns(**HI(coef(**Cmem**)))**))), ACx = M40(rnd(ACx + **(**uns(Smem) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X
[4]	ACy = M40(rnd(ACy − **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**)))**))), ACx = M40(rnd(ACx + **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X

Description

These instructions perform two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also

See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiply and Subtracts

Multiply and Subtract with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(ACy + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode `|1000 0011|XXXM MMYY|YMMM 01mm|uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = M40(rnd(AC0 – (uns(*AR0) * uns(coef(*CDP))))),
AC1 = M40(rnd(AC1 + (uns(*AR1) * uns(coef(*CDP)))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR0 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is rounded and stored in AC0. The product of the unsigned content addressed by AR1 and the unsigned content addressed by CDP is added to the content of AC1. The result is rounded and stored in AC1.

```
Before                          After
AC0        00 6900 0000         AC0        00 486B 0000
AC1        00 0023 0000         AC1        00 95E3 0000
*AR0               3400         *AR0               3400
*AR1               EF00         *AR1               EF00
*CDP               A067         *CDP               A067
ACOV0                 0         ACOV0                 0
ACOV1                 0         ACOV1                 0
CARRY                 0         CARRY                 0
FRCT                  0         FRCT                  0
```

Multiply and Subtract with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode

```
|1000  0100|XXXM  MMYY|YMMM  00mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).

- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = AC0 – (uns(*AR3) * uns(coef(*CDP))),
AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

Multiply and Subtract with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy − **(**uns(Smem) * uns(**HI(coef(**Cmem**))**)**)**)), ACx = M40(rnd(ACx + **(**uns(Smem) * uns(**LO(coef(**Cmem**))**)**)**))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0001 11mm | DDDD uug%

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 + (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy−M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                          After
AC0         00 0000 8000        AC0         00 3F80 8000
XAR3             00 10FF        XAR3             00 10FE
Data memory
10FFh               FE00        10FFh               FE00
XCDP             00 2000        XCDP             00 2002
Coeff memory
2001h               4000        2001h               4000

AC1         00 0000 8000        AC1         FF 8100 8000
Coeff memory
2000h               8000        2000h               8000
```

Multiply and Subtract with Parallel Multiply and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy − (uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**)))**)))), ACx = M40(rnd(ACx + (uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**)))**))))	No	4	1	X

Opcode | 1111 1101 | AAAA AAAI | 0101 11mm | DDDD uug%

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS) and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = AC0 + (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy–M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) –> ACy
ACx+M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) –> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         00 3F80 8000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         FF 8080 8000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MASM:MOV *Multiply and Subtract with Parallel Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – **(**Tx * Xmem**)**), Ymem = **HI(**ACx **<< T2)** [,T3 = Xmem]	No	4	1	X

Opcode `1000 0111 | XXXM MMYY | YMMM SSDD | 010x ssU%`

Operands ACx, ACy, Tx, Xmem, Ymem

Description This instruction performs two operations in parallel: multiply and subtract (MAS) and store.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- This instruction provides the option to store the 16-bit data memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- The input operand is shifted in the D-unit shifter according to SXMD.
- After the shift, the high part of the accumulator, ACx(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = rnd(ACy – (Tx * Xmem)),
  Ymem = HI(saturate(uns(ACx << T2))) [,T3 = Xmem]
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = rnd(ACy – (Tx * Xmem)),
  Ymem = HI(saturate(ACx << T2)) [,T3 = Xmem]

Status Bits

Affected by	C54CM, FRCT, M40, RDM, SATD, SMUL, SST, SXMD
Affects	ACOVy

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Parallel Multiply and Subtracts

Example

Syntax	Description
AC0 = AC0 – (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The product of the content addressed by AR3 and the content of T0 is subtracted from the content of AC0. The result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31−16) is stored at the address of AR4.

NEG Negate Accumulator, Auxiliary, or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = – src	Yes	2	1	X

Opcode

```
| 0011  010E | FSSS  FDDD
```

Operands dst, src

Description This instruction computes the 2s complement of the content of the source register (src). This instruction clears the CARRY status bit to 0 for all nonzero values of src. If src equals 0, the CARRY status bit is set to 1.

❑ When the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
- Overflow detection and CARRY status bit depends on M40.
- When an overflow is detected, the accumulator is saturated according to SATD.

❑ When the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
- Overflow detection is done at bit position 15.
- When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	M40, SATA, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Complement Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Complement Accumulator, Auxiliary, or Temporary Register Content

Example

Syntax	**Description**
AC0 = –AC1	The 2s complement of the content of AC1 is stored in AC0.

NOP *No Operation (nop)*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**nop**	Yes	1	1	D
[2]	**nop_16**	Yes	2	1	D

Opcode `0010 000E`

Operands none

Description Instruction [1] increments the program counter register (PC) by 1 byte. Instruction [2] increments the PC by 2 bytes.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
nop	The program counter (PC) is incremented by 1 byte.

AMAR *Parallel Modify Auxiliary Register Contents*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**mar(**Xmem**), mar(**Ymem**), mar(coef(**Cmem**))**	No	4	1	X

Opcode

```
|1000 0101|XXXM MMYY|YMMM 10mm|xxxx xxxx
```

Operands Cmem, Xmem, Ymem

Description This instruction performs three parallel modify auxiliary register (MAR) operations in one cycle. The auxiliary register modification is specified by:

- The content of data memory operand Xmem
- The content of data memory operand Ymem
- The content of a data memory operand Cmem, addressed using the coefficient addressing mode

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Modify Auxiliary Register Content
- Modify Extended Auxiliary Register Content

Example

Syntax	Description
mar(*AR3+), mar(*AR4−), mar(coef(*CDP))	AR3 is incremented by 1. AR4 is decremented by 1. CDP is not modified.

MPY::MPY *Parallel Multiplies*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(**coef(**Cmem**)**))), ACy = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**)**)))	No	4	1	X
[2]	ACy = M40(rnd(uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X
[3]	ACy = M40(rnd(uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X
[4]	ACy = M40(rnd(uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5	1	X

Description These instructions perform two parallel multiply operations in one cycle. The operations are executed in the two D-unit MACs.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

See Also See the following other related instructions:

- Modify Auxiliary Register Content with Parallel Multiply
- Multiply
- Multiply and Accumulate with Parallel Multiply
- Multiply and Subtract with Parallel Multiply
- Parallel Multiply and Accumulates
- Parallel Multiply and Subtracts

Parallel Multiplies

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(uns(Xmem) * uns(**coef(**Cmem**)**))), ACy = M40(rnd(uns(Ymem) * uns(**coef(**Cmem**)**)))	No	4	1	X

Opcode `| 1000 0010 | XXXM MMYY | YMMM 00mm | uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = uns(*AR3) * uns(coef(*CDP)),
AC1 = uns(*AR4) * uns(coef(*CDP))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is stored in AC1.

Parallel Multiplies

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode

```
|1111  1101|AAAA  AAAI|0000  00mm|DDDD  uug%
```

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.

- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = uns(*AR3−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FF       XAR3             00 10FE
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F00 0000
Coeff memory
2000h               8000       2000h               8000
```

Parallel Multiplies

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode `1111 1101 | AAAA AAAI | 0100 00mm | DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(HI(*AR3−)) * uns(HI(coef(*CDP+))),
AC0 = uns(LO(*AR3−)) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Parallel Multiplies

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001  0010|XXXM  MMYY|YMMM  00mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply operations in one cycle. The operations are executed in the D-unit MACs.

The first operation performs a multiplication in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits.

- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- ❑ The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = uns(*AR3−) * uns(HI(coef(*CDP+))),
AC0 = uns(*AR2−) * uns(LO(coef(*CDP+)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
M40(rnd(uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

```
Before                         After
AC0         FF 8000 0000       AC0         00 3F80 0000
XAR2             00 10FE       XAR2             00 10FD
XAR3             00 20FE       XAR3             00 20FD
Data memory
10FEh               FE00       10FEh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         FF 8000 0000       AC1         00 7F80 0000
Data memory
20FEh               FF00       20FFh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MAC::MAC Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[2]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[3]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[4]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[5]	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[6]	ACy = M40(rnd((ACy >> #16) + (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[7]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[8]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[9]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[10]	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))	No	5	1	X

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = M40(rnd(ACy + **(**uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(**LO(coef(**Cmem**))**))))	No	5	1	X
[12]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(**LO(coef(**Cmem**))**))))	No	5	1	X

Description

These instructions perform two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also

See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- ❑ Multiply and Accumulate
- ❑ Multiply and Accumulate with Parallel Delay
- ❑ Multiply and Accumulate with Parallel Load Accumulator from Memory
- ❑ Multiply and Accumulate with Parallel Multiply
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply with Parallel Multiply and Accumulate
- ❑ Parallel Multiplies
- ❑ Parallel Multiply and Subtracts

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx + **(**uns(Xmem) * uns(**coef(**Cmem**)**)))), ACy = M40(rnd(ACy + **(**uns(Ymem) * uns(**coef(**Cmem**)**))))	No	4	1	X

Opcode `1000 0011 | XXXM MMYY | YMMM 00mm | uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits

	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	**Description**

AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))),
AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1. The result is stored in AC1.

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M4(rnd(ACy + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode | `1000 0011` | `XXXM MMYY` | `YMMM 10mm` | `uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).

- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))),
AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1. The result is stored in AC1.

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode `|1000 0100|XXXM MMYY|YMMM 11mm|uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- Input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.

- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- ■ mar(Xmem)
- ■ mar(Ymem)
- ■ mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))),
AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1.

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy + **(**uns(Smem) * uns(**HI(coef(**Cmem**))))**)), ACx = M40(rnd(ACx + **(**uns(Smem) * uns(**LO(coef(**Cmem**))))**))	No	4	1	X

Opcode

```
|1111 1101|AAAA AAAI|0001 01mm|DDDD uug%
```

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 + (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. AR3 is decremented by 1 and CDP. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0          00 0000 8000      AC0          00 3F80 8000
XAR3              00 10FF      XAR3              00 10FE
Data memory
10FFh                FE00      10FFh                FE00
XCDP              00 2000      XCDP              00 2002
Coeff memory
2001h                4000      2001h                4000

AC1          00 0000 8000      AC1          00 7F00 8000
Coeff memory
2000h                8000      2000h                8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = M40(rnd(ACy + **(**uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Smem) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode

```
|1111  1101 |AAAA  AAAI |0010  00mm |DDDD  uug%
```

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	**Description**

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
(ACx>>#16)+M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx

Before                          After
AC0         00 0800 0000        AC0         00 3F80 0800
XAR3             00 10FF        XAR3             00 10FE
Data memory
10FFh               FE00        10FFh               FE00
XCDP             00 2000        XCDP             00 2002
Coeff memory
2001h               4000        2001h               4000

AC1         00 0000 8000        AC1         00 7F80 8000
Coeff memory
2000h               8000        2000h               8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = M40(rnd((ACy **>> #16)** + **(**uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd((ACx **>> #16)** + **(**uns(Smem) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode

```
|1111  1101|AAAA  AAAI|0010  11mm|DDDD  uug%
```

Operands ACx, ACy, Cmem, Smem

Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

❑ Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
(ACy>>#16)+M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
(ACx>>#16)+M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                          After
AC0         00 0800 0000        AC0         00 3F80 0800
XAR3             00 10FF        XAR3             00 10FE
Data memory
10FFh               FE00        10FFh               FE00
XCDP             00 2000        XCDP             00 2002
Coeff memory
2001h               4000        2001h               4000

AC1         00 0800 0000        AC1         00 7F00 0800
Coeff memory
2000h               8000        2000h               8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = M40(rnd(ACy + (uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**))**)))), ACx = M40(rnd(ACx + (uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**))**))))	No	4	1	X

Opcode | `1111 1101` | `AAAA AAAI` | `0101 01mm` | `DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB, and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = AC0 + (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx+M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         00 3F80 8000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         00 7F80 8000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X

Opcode `1111 1101 | AAAA AAAI | 0110 00mm | DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB, and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- ❑ For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
ACy+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
(ACx>>#16)+M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                        After
AC0         00 0800 8000      AC0         00 3F80 0800
XAR3             00 10FE      XAR3             00 10FC
Data memory
10FFh               FE00      10FFh               FE00
XCDP             00 2000      XCDP             00 2002
Coeff memory
2001h               4000      2001h               4000

AC1         00 0000 8000      AC1         00 7F80 8000
Data memory
10FEh               FF00      10FEh               FF00
Coeff memory
2000h               8000      2000h               8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = M40(rnd((ACy >> #16) + (uns(HI(Lmem))* uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(LO(Lmem))* uns(LO(coef(Cmem))))))	No	4	1	X

Opcode

```
|1111 1101|AAAA AAAI|0110 11mm|DDDD uug%
```

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator, which has been shifted to the right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = (AC1 >> #16) + (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1, which has been shifted to the right by 16 bits. The result is stored in AC1. The product of the unsigned content addressed by lower part of AR3 and the unsigned content addressed by the lower part of CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
(ACy>>#16)+M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
(ACx>>#16)+M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0800 0000       AC0         00 3F80 0800
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0800 0000       AC1         00 7F80 0800
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = M40(rnd(ACy + uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx + uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001  0011|XXXM  MMYY|YMMM  00mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 + (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(ACx + uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(ACy + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before		**After**	
AC0	00 0000 8000	AC0	00 3F80 8000
XAR2	00 10FE	XAR2	00 10FD
XAR3	00 20FE	XAR3	00 20FD
Data memory			
10FEh	FE00	10FEh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0000 8000	AC1	00 7F80 8000
Data memory			
20FEh	FF00	20FFh	FF00
Coeff memory			
2000h	8000	2000h	8000

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001 0011|XXXM MMYY|YMMM 10mm|uuDD DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the contents of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an accumulation in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the contents of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd((ACx >> #16) + uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(ACy + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

Before		**After**	
AC0	00 0800 8000	AC0	00 3F80 0800
XAR2	00 10FE	XAR2	00 10FD
XAR3	00 20FE	XAR3	00 20FD
Data memory			
10FEh	FE00	10FEh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0000 8000	AC1	00 7F80 8000
Data memory			
20FEh	FF00	20FFh	FF00
Coeff memory			
2000h	8000	2000h	8000

Parallel Multiply and Accumulates

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = M40(rnd(**(**ACy **>> #16)** + **(**uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(**(**ACx **>> #16)** + **(**uns(Xmem) * uns(**LO(coef(**Cmem**))**))))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode `|1001 0011|XXXM MMYY|YMMM 11mm|uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and an accumulation in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and an addition in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The input operands are extended to 17 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 + (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = (AC0 >> #16) + (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is added to the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is added to the content of AC0, which has been shifted to the right by 16 bits. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd((ACx >> #16) + uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd((ACy >> #16) + uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

```
Before                         After
AC0         00 0800 8000       AC0         00 3F80 0800
XAR2             00 10FE       XAR2             00 10FD
XAR3             00 20FE       XAR3             00 20FD
Data memory
10FEh               FE00       10FEh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 8000 0000       AC1         00 7F80 8000
Data memory
20FEh               FF00       20FFh               FF00
Coeff memory
2000h               8000       2000h               8000
```

MAS::MAS *Parallel Multiply and Subtracts*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(coef(Cmem)))))	No	4	1	X
[2]	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))	No	4	1	X
[3]	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))	No	4	1	X
[4]	ACy = M40(rnd(ACy − (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))	No	5	1	X

Description These instructions perform two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Modify Auxiliary Register Content with Parallel Multiply and Subtract
- ❑ Multiply and Subtract
- ❑ Multiply and Subtract with Parallel Load Accumulator from Memory
- ❑ Multiply and Subtract with Parallel Multiply
- ❑ Multiply and Subtract with Parallel Multiply and Accumulate
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Parallel Multiplies
- ❑ Parallel Multiply and Accumulates

Parallel Multiply and Subtracts

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = M40(rnd(ACx – **(**uns(Xmem) * uns(**coef(**Cmem**))**))), ACy = M40(rnd(ACy – **(**uns(Ymem) * uns(**coef(**Cmem**))**)))	No	4	1	X

Opcode `| 1000 0101 | XXXM MMYY | YMMM 01mm | uuDD DDg%`

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

The second operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode and extended to 17 bits.

- ❑ Input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	**Description**

AC0 = AC0 – (uns(*AR3) * uns(coef(*CDP))),
AC1 = AC1 – (uns(*AR4) * uns(coef(*CDP)))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0. The result is stored in AC0. The product of the unsigned content addressed by AR4 and the unsigned content addressed by CDP is subtracted from the content of AC1. The result is stored in AC1.

Parallel Multiply and Subtracts

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = M40(rnd(ACy − **(**uns(Smem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − **(**uns(Smem) * uns(**LO(coef(**Cmem**))**)))	No	4	1	X

Opcode `| 1111 1101 | AAAA AAAI | 0011 00mm | DDDD uug%`

Operands ACx, ACy, Cmem, Smem

Description This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand HI(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC2 (this data is shared to MAC1 and MAC2). The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA (effective address); the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand Smem and the content of data memory operand LO(coef(Cmem)). The data memory operand Smem is addressed by DAGEN path X with the corresponding addressing mode, driven on data bus DDB, and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- Multiplication overflow detection depends on SMUL.

- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR3−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 is decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy−M40(rnd(uns(Smem)[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx−M40(rnd(uns(Smem)[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

Before		**After**	
AC0	00 0000 8000	AC0	FF C080 8000
XAR3	00 10FF	XAR3	00 10FE
Data memory			
10FFh	FE00	10FFh	FE00
XCDP	00 2000	XCDP	00 2002
Coeff memory			
2001h	4000	2001h	4000
AC1	00 0000 8000	AC1	FF 8100 8000
Coeff memory			
2000h	8000	2000h	8000

Parallel Multiply and Subtracts

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = M40(rnd(ACy − **(**uns(**HI(**Lmem**)**) * uns(**HI(coef(**Cmem**)))**))), ACx = M40(rnd(ACx − **(**uns(**LO(**Lmem**)**) * uns(**LO(coef(**Cmem**)))**)))	No	4	1	X

Opcode | `1111 1101` | `AAAA AAAI` | `0111 00mm` | `DDDD uug%`

Operands ACx, ACy, Cmem, Lmem

Description This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the content of data memory operand HI(Lmem) and the content of data memory operand HI(coef(Cmem)). The data memory operand HI(Lmem) is addressed by DAGEN path X with the EA (effective address); the data, which can be assumed to be the higher part of long word memory data, is driven on data bus CDB and sign extended to 17 bits in the MAC2. The other data memory operand HI(coef(Cmem)) is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the content of data memory operand LO(Lmem) and the content of data memory operand LO(coef(Cmem)). The data memory operand LO(Lmem) is addressed by DAGEN path X with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word memory data, is driven on data bus DDB and sign extended to 17 bits in the MAC1. The other data memory operand LO(coef(Cmem)) is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.

❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.

- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(HI(*AR3−)) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(LO(*AR3−)) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by the higher part of AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by the lower part of AR3 and the unsigned content addressed by the lower part of CDP is subtracted from the content of AC0. The result is stored in AC0. When AR3− is used with HI/LO, AR3 is decremented by 2. When CDP+ is used with HI/LO, CDP is incremented by 2.

```
Execution
ACy−M40(rnd(uns(HI(Lmem))[16:0]*uns(HI(coef(Cmem)))[16:0])) -> ACy
ACx−M40(rnd(uns(LO(Lmem))[16:0]*uns(LO(coef(Cmem)))[16:0])) -> ACx
```

```
Before                         After
AC0         00 0000 8000       AC0         FF C080 8000
XAR3             00 10FE       XAR3             00 10FC
Data memory
10FFh               FE00       10FFh               FE00
XCDP             00 2000       XCDP             00 2002
Coeff memory
2001h               4000       2001h               4000

AC1         00 0000 8000       AC1         FF 8080 8000
Data memory
10FEh               FF00       10FEh               FF00
Coeff memory
2000h               8000       2000h               8000
```

Parallel Multiply and Subtracts

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = M40(rnd(ACy − (uns(Ymem) * uns(**HI(coef(**Cmem**))**))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(**LO(coef(**Cmem**))**)))	No	5 (*)	1	X

(*) 1 LSB is allocated to instruction slot #2.

Opcode

```
|1001  0101|XXXM  MMYY|YMMM  01mm|uuDD  DDg%
```

Operands ACx, ACy, Cmem, Xmem, Ymem

Description This instruction performs two parallel multiply and subtraction (MAS) operations in one cycle. The operations are executed in the two D-unit MACs.

The first operation performs a multiplication and a subtraction in the D-unit MAC2. The input operands of the multiplier are the contents of data memory operand Ymem, extended to 17 bits, and the content of data memory operand HI(coef(Cmem)) which is addressed by DAGEN path C with the EA; the data, which can be assumed to be the higher part of long word coefficient data, is driven on data bus B2DB and sign extended to 17 bits in the MAC2.

The second operation performs a multiplication and a subtraction in the D-unit MAC1. The input operands of the multiplier are the contents of data memory operand Xmem, extended to 17 bits, and the content of data memory operand LO(coef(Cmem)) which is addressed by DAGEN path C with the next address of EA (EA+1 when EA is even, EA−1 when EA is odd); the data, which can be assumed to be the lower part of long word coefficient data, is driven on data bus BDB and sign extended to 17 bits in the MAC1.

- ❑ The input operands are extended to 17 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.
- ❑ If FRCT = 1, the output of the multiplier is shifted to the left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.

- The 32-bit result of the multiplication is sign extended to 40 bits.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, the accumulator is saturated according to SATD.
- Because this instruction occupies both instruction slots #1 and #2, this can not be executed in parallel with other instructions.
- The Xmem operand can access the MMRs but the Ymem operand can not.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BAB, BDB, and B2DB buses; on some C55xx-based devices, the BAB, BDB, and B2DB buses are only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Compatibility with C54x devices (C54CM = 1)

None.

Status Bits

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description

AC1 = AC1 − (uns(*AR3−) * uns(HI(coef(*CDP+)))),
AC0 = AC0 − (uns(*AR2−) * uns(LO(coef(*CDP+))))
	Both instructions are performed in parallel. The product of the unsigned content addressed by AR3 and the unsigned content addressed by the higher part of the coefficient data pointer register (CDP) is subtracted from the content of AC1. The result is stored in AC1. The product of the unsigned content addressed by AR2 and the unsigned content addressed by the lower part of the CDP is subtracted from the content of AC0. The result is stored in AC0. AR3 and AR2 are decremented by 1. When CDP+ is used with HI/LO, CDP is incremented by 2.

Execution

```
M40(rnd(ACx – uns(Xmem)[16:0] * uns(LO(coef(Cmem)))[16:0])) -> ACx
M40(rnd(ACy – uns(Ymem)[16:0] * uns(HI(coef(Cmem)))[16:0])) -> ACy
```

```
Before                         After
AC0          00 0000 8000      AC0          FF C080 8000
XAR2              00 10FE      XAR2              00 10FD
XAR3              00 20FE      XAR3              00 20FD
Data memory
10FEh                FE00      10FEh                FE00
XCDP              00 2000      XCDP              00 2002
Coeff memory
2001h                4000      2001h                4000

AC1          00 0000 8000      AC1          FF 8080 8000
Data memory
20FEh                FF00      20FFh                FF00
Coeff memory
2000h                8000      2000h                8000
```

port — *Peripheral Port Register Access Qualifiers*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**readport()**	No	1	1	D
[2]	**writeport()**	No	1	1	D

Opcode

```
 readport                               |1001 1001
writeport                               |1001 1010
```

Operands none

Description These operand qualifiers allow you to locally disable access toward the data memory and enable access to the 64K-word I/O space. The I/O data location is specified by the Smem, Xmem, or Ymem fields.

- ❑ A readport() operand qualifier may be included in any instruction making a word single data memory access Smem or Xmem that is used in a read operation, except instructions using delay().
- ❑ A readport() operand qualifier cannot be used in any instruction making a dual memory access Xmem or Ymem that is used in read operation. There is an exception for the instructions making a dual read/write memory access of the type Ymem = Xmem, or Smem = coeff, where readport() qualifier can be used.
- ❑ A writeport() operand qualifier may be included in any instruction making a word single data memory access Smem or Ymem that is used in a write operation, except instructions using the delay().
- ❑ A writeport() operand qualifier cannot be used in any instruction making a dual memory access Xmem or Ymem that is used in write operation. There is an exception for the instructions making a dual read/write memory access of the type Ymem = Xmem, or coeff = Smem, where writeport() qualifier can be used.
- ❑ A readport() or writeport() operand qualifier cannot be used as a stand-alone instruction (the assembler generates an error message).

Any instruction making a word single data memory access Smem (except those listed above) can use the *port(#k16) addressing mode to access the 64K-word I/O space with an immediate address. When an instruction uses *port(#k16), the 16-bit unsigned constant, k16, is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using *port(#k16) cannot be executed in parallel with another instruction.

The following indirect operands cannot be used for accesses to I/O space. An instruction using one of these operands requires a 2-byte extension to the instruction. Because of the extension, an instruction using one of the following indirect operands cannot be executed with these operand qualifiers.

- ❑ *ARn(#K16)
- ❑ *+ARn(#K16)
- ❑ *CDP(#K16)
- ❑ *+CDP(#K16)

Status Bits Affected by none

Affects none

Repeat An instruction using this operand qualifier can be repeated.

Example 1

Syntax	Description
T2 = *AR3 || readport()	The content addressed by AR3 (I/O address) is loaded into T2.

Example 2

Syntax	Description
*AR3 = T2 || writeport()	The content of T2 is written to the location addressed by AR3 (I/O address).

POPBOTH *Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	xdst = **popboth()**	Yes	2	1	X

Opcode | 0101 000E | XDDD 0100

Operands xdst

Description This instruction moves the content of two 16-bit data memory locations addressed by the data stack pointer (SP) and system stack pointer (SSP) to accumulator ACx or to the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

The content of xdst(15−0) is loaded from the location addressed by SP and the content of xdst(31−16) is loaded from the location addressed by SSP. The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process.

When xdst is a 23-bit register, the upper 9 bits of the data memory addressed by SSP are discarded and only the 7 lower bits of the data memory are loaded into the high part of xdst(22−16).

When xdst is an accumulator, the guard bits, ACx(39−32), are reloaded (unchanged) with the current value and are not modified by this instruction.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Pop Top of Stack
- Push to Top of Stack
- Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

POP *Pop Top of Stack*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = **pop()**	Yes	2	1	X
[2]	dst = **pop()**	Yes	2	1	X
[3]	dst, Smem = **pop()**	No	3	1	X
[4]	ACx = **dbl(pop())**	Yes	2	1	X
[5]	Smem = **pop()**	No	2	1	X
[6]	**dbl(**Lmem**)** = **pop()**	No	2	1	X

Description These instructions move the content of the data memory location addressed by the data stack pointer (SP) to:

- an accumulator, auxiliary, or temporary register
- a data memory location

The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process.

When the destination register is an accumulator, the guard bits and the 16 higher bits of the accumulator, ACx(39−16), are reloaded (unchanged) with the current value and are not modified by these instructions.

The increment operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.

Status Bits Affected by none

Affects none

See Also See the following other related instructions:

- Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers
- Push to Top of Stack
- Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = **pop()**	Yes	2	1	X

Opcode

```
| 0011  101E | FSSS  FDDD
```

Note: FSSS = dst1, FDDD = dst2

Operands dst1, dst2

Description This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst1 and moves the content of the 16-bit data memory location pointed by SP + 1 to destination register dst2.

When the destination register, dst1 or dst2, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15−0). The guard bits and the 16 higher bits of the accumulator, ACx(39−16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0, AC1 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0) and the content of the memory location pointed by SP + 1 is copied to AC1(15–0). Bits 39−16 of the accumulators are unchanged. The SP is incremented by 2.

```
Before                         After
AC0         00 4500 0000       AC0         00 4500 4890
AC1         F7 5678 9432       AC1         F7 5678 2300
SP                  0300       SP                  0302
300                 4890       300                 4890
301                 2300       301                 2300
```

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = **pop()**	Yes	2	1	X

Opcode `|0101 000E|FDDD x010`

Operands dst

Description This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst.

When the destination register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15−0). The guard bits and the 16 higher bits of the accumulator, ACx(39−16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 1.

Status Bits

Affected by none

Affects none

Repeat This instruction cannot be repeated with a single conditional or unconditional repeat instruction. It can be repeated in other repeat instructions.

Example

Syntax	Description
AC0 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0). Bits 39−16 of AC0 are unchanged. The SP is incremented by 1.

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst, Smem = **pop()**	No	3	1	X

Opcode `1110 0100 | AAAA AAAI | FDDD x1xx`

Operands dst, Smem

Description This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst and moves the content of the 16-bit data memory location pointed by SP + 1 to data memory (Smem) location.

When the destination register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15−0). The guard bits and the 16 higher bits of the accumulator, ACx(39−16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0, *AR3 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0) and the content of the memory location pointed by SP + 1 is copied to the location addressed by AR3. Bits 39−16 of AC0 are unchanged. The SP is incremented by 2.

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = **dbl(pop())**	Yes	2	1	X

Opcode

```
|0101  000E|xxDD  x011
```

Operands ACx

Description This instruction moves the content of the 16-bit data memory location pointed by SP to the accumulator high part ACx(31–16) and moves the content of the 16-bit data memory location pointed by SP + 1 to the accumulator low part ACx(15–0).

The guard bits of the accumulator, ACx(39−32), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated with a single conditional or unconditional repeat instruction. It can be repeated in other repeat instructions.

Example

Syntax	Description
AC1 = dbl(pop())	The content of the memory location pointed by the data stack pointer (SP) is copied to AC1(31–16) and the content of the memory location pointed by SP + 1 is copied to AC1(15–0). Bits 39−32 of AC1 are unchanged. The SP is incremented by 2.

```
Before                          After
AC1         03 3800 FC00        AC1         03 5644 F800
SP                  0304        SP                  0306
304                 5644        304                 5644
305                 F800        305                 F800
```

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = **pop()**	No	2	1	X

Opcode `| 1011 1011 | AAAA AAAI`

Operands Smem

Description This instruction moves the content of the 16-bit data memory location pointed by SP to data memory (Smem) location. SP is incremented by 1.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR1 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to the location addressed by AR1. The SP is incremented by 1.

```
Before                      After
AR1            0200         AR1            0200
SP             0300         SP             0301
200            3400         200            6903
300            6903         300            6903
```

Pop Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	**dbl(**Lmem**)** = **pop()**	No	2	1	X

Opcode `1011 1000 AAAA AAAI`

Operands Lmem

Description This instruction moves the content of the 16-bit data memory location pointed by SP to the 16 highest bits of data memory location Lmem and moves the content of the 16-bit data memory location pointed by SP + 1 to the 16 lowest bits of data memory location Lmem.

When Lmem is at an even address, the two 16-bit values popped from the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values popped from the stack are stored at memory location Lmem in the reverse order.

SP is incremented by 2.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
dbl(*AR3–) = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to the 16 highest bits of the location addressed by AR3 and the content of the memory location pointed by SP + 1 is copied to the 16 lowest bits of the location addressed by AR3. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution. The SP is incremented by 2.

PSHBOTH *Push Accumulator or Extended Auxiliary Register Content to Stack Pointers*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**pshboth(**xsrc**)**	Yes	2	1	X

Opcode |0101 000E|XSSS 0101

Operands xsrc

Description This instruction moves the lower 32 bits of ACx or the content of the 23-bit source register (XARx, XSP, XSSP, XDP, or XCDP) to the two 16-bit memory locations addressed by the data stack pointer (SP) and system stack pointer (SSP). The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process.

The content of xsrc(15−0) is moved to the location addressed by SP and the content of xsrc(31−16) is moved to the location addressed by SSP.

When xsrc is a 23-bit register, the upper 9 bits of the location addressed by SSP are filled with 0.

Status Bits		
	Affected by	none
	Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers
- ❑ Pop Top of Stack
- ❑ Push to Top of Stack

PSH *Push to Top of Stack*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**push(**src1, src2**)**	Yes	2	1	X
[2]	**push(**src**)**	Yes	2	1	X
[3]	**push(**src, Smem**)**	No	3	1	X
[4]	**dbl(push(**ACx**))**	Yes	2	1	X
[5]	**push(**Smem**)**	No	2	1	X
[6]	**push(dbl(**Lmem**))**	No	2	1	X

Description

These instructions move one or two operands to the data memory location addressed by the data stack pointer (SP). The return address register (RETA) and the control-flow context register (CFCT) are not accessed by this instruction even in the fast-return process. The operands may be:

- an accumulator, auxiliary, or temporary register
- a data memory location

The decrement operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.

Status Bits

Affected by none

Affects none

See Also

See the following other related instructions:

- Pop Top of Stack
- Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers
- Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**push(**src1, src2**)**	Yes	2	1	X

Opcode

```
|0011  100E|FSSS  FDDD
```

Note: FSSS = src1, FDDD = src2

Operands src1, src2

Description This instruction decrements SP by 2, then moves the content of the source register src1 to the 16-bit data memory location pointed by SP and moves the content of the source register src2 to the 16-bit data memory location pointed by SP + 1.

When the source register, src1 or src2, is an accumulator, the source accumulator low part, ACx(15−0), is moved to the 16-bit data memory operand.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
push(AR0, AC1)	The data stack pointer (SP) is decremented by 2. The content of AR0 is copied to the memory location pointed by SP and the content of AC1(15–0) is copied to the memory location pointed by SP + 1.

```
Before                          After
AR0               0300          AR0               0300
AC1        03 5644 F800         AC1        03 5644 F800
SP                0300          SP                02FE
2FE               0000          2FE               0300
2FF               0000          2FF               F800
300               5890          300               5890
```

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**push(**src**)**	Yes	2	1	X

Opcode | `0101 000E | FSSS x110`

Operands src

Description This instruction decrements SP by 1, then moves the content of the source register (src) to the 16-bit data memory location pointed by SP. When the source register is an accumulator, the source accumulator low part, ACx(15−0), is moved to the 16-bit data memory operand.

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated with a single conditional or unconditional repeat instruction. It can be repeated in other repeat instructions.

Example

Syntax	Description
push(AC0)	The data stack pointer (SP) is decremented by 1. The content of AC0(15–0) is copied to the memory location pointed by SP.

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	**push(**src, Smem**)**	No	3	1	X

Opcode `1110 0100 | AAAA AAAI | FSSS x0xx`

Operands Smem, src

Description This instruction decrements SP by 2, then moves the content of the source register (src) to the 16-bit data memory location pointed by SP and moves the content of the data memory (Smem) location to the 16-bit data memory location pointed by SP + 1.

When the source register is an accumulator, the source accumulator low part, ACx(15−0), is moved to the 16-bit data memory operand.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
push(AC0, *AR3)	The data stack pointer (SP) is decremented by 2. The content of AC0(15–0) is copied to the memory location pointed by SP and the content addressed by AR3 is copied to the memory location pointed by SP + 1.

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	**dbl(push(**ACx**))**	Yes	2	1	X

Opcode `| 0101 000E | xxSS x111`

Operands ACx

Description This instruction decrements SP by 2, then moves the content of the accumulator high part ACx(31–16) to the 16-bit data memory location pointed by SP and moves the content of the accumulator low part ACx(15–0) to the 16-bit data memory location pointed by SP + 1.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated with a single conditional or unconditional repeat instruction. It can be repeated in other repeat instructions.

Example

Syntax	Description
dbl(push(AC0))	The data stack pointer (SP) is decremented by 2. The content of AC0(31–16) is copied to the memory location pointed by SP and the content of AC0(15–0) is copied to the memory location pointed by SP + 1.

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	**push(**Smem**)**	No	2	1	X

Opcode | 1011 0101 | AAAA AAAI

Operands Smem

Description This instruction decrements SP by 1, then moves the content of the data memory (Smem) location to the 16-bit data memory location pointed by SP.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
push(*AR1)	The data stack pointer (SP) decremented by 1. The content addressed by AR1 is copied to the memory location pointed by SP.

```
Before                    After
*AR1          6903        *AR1          6903
SP            0305        SP            0304
304           0000        304           6903
305           0300        305           0300
```

Push to Top of Stack

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	**push(dbl(**Lmem**))**	No	2	1	X

Opcode

```
|1011 0111|AAAA AAAI
```

Operands Lmem

Description This instruction decrements SP by 2, then moves the 16 highest bits of data memory location Lmem to the 16-bit data memory location pointed by SP and moves the 16 lowest bits of data memory location Lmem to the 16-bit data memory location pointed by SP + 1.

When Lmem is at an even address, the two 16-bit values pushed onto the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values pushed onto the stack are stored at memory location Lmem in the reverse order.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
push(dbl(*AR3–))	The data stack pointer (SP) is decremented by 2. The 16 highest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP and the 16 lowest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP + 1. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution.

RPTB *Repeat Block of Instructions Unconditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**localrepeat{}**	Yes	2	1	AD
[2]	**blockrepeat{}**	Yes	3	1	AD

Description

These instructions repeat a block of instructions the number of times specified by:

- The content of BRC0 + 1, if no loop has already been detected.
- The content of BRS1 + 1, if one level of the loop has already been detected.

Loop structures defined by these instructions must have the following characteristics:

- The minimum number of instructions executed within one loop iteration is 2.
- The minimum number of cycles executed within one loop iteration is 2.
- Since the result of updating BRCx (and BRAF in C54CM = 1) within 3 instruction cycles from the end of the loop is uncertain (effective in the same iteration or the next iteration depending on the pipeline state), this operation is prohibited.
- The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.
- C54CM bit in ST1_55 cannot be modified within a block-repeat loop.

These instructions cannot be repeated.

See section 1.5 for a list of instructions that cannot be used in a repeat block mechanism.

Status Bits

Affected by	none
Affects	none

See Also

See the following other related instructions:

- Repeat Single Instruction Conditionally
- Repeat Single Instruction Unconditionally
- Repeat Single Instruction Unconditionally and Decrement CSR
- Repeat Single Instruction Unconditionally and Increment CSR

Repeat Block of Instructions Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**localrepeat{}**	Yes	2	1	AD

Opcode `0100 101E | 1lll lll1`

Operands none

Description This instruction repeats a block of instructions the number of times specified by:

❑ The content of BRC0 + 1, if no loop has already been detected. In this case:

- In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
- The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
- BRC0 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
- BRC0 contains 0 after the block-repeat operation has ended.

❑ The content of BRS1 + 1, if one level of the loop has already been detected. In this case:

- BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
- In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
- The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
- BRC1 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
- BRC1 contains 0 after the block-repeat operation has ended.
- BRS1 content is not impacted by the block-repeat operation.

Loop structures defined by this instruction must have the following characteristics:

- ❑ The minimum number of instructions executed within one loop iteration is 2.
- ❑ The minimum number of cycles executed within one loop iteration is 2.
- ❑ The maximum loop size is 128 bytes.
- ❑ Since the result of updating BRCx (and BRAF in C54CM = 1) within 3 instruction cycles from the end of the loop is uncertain (effective in the same iteration or the next iteration depending on the pipeline state), this operation is prohibited.
- ❑ C54CM bit in ST1_55 cannot be modified within a block-repeat loop.
- ❑ The following instructions cannot be used as the last instruction in the loop structure:

while (cond && (RPTC < k8)) repeat	repeat(k8)	repeat(CSR), CSR += k4
if (cond) execute(AD_Unit)	repeat(k16)	repeat(CSR), CSR += TAx
if (cond) execute(D_Unit)	repeat(CSR)	repeat(CSR), CSR –= k4

Note:

Instructions if (cond) execute (AD_Unit), or if (cond) execute (D_Unit) can be used as the last instruction in the loop structure if the instruction is executed with the instruction with which it is paralleled (if (cond) execute (AD_Unit) || instruction_executes conditionally)

A local loop is defined as when all the code of the loop is repeatedly executed from within the instruction buffer queue (IBQ):

- ❑ All the code of the local loop must fit within the 128-byte, 4-byte-aligned IBQ; therefore, local repeat blocks are limited to 128 bytes minus the 0 to 3 bytes of first-instruction misalignment. The 128th byte of the IBQ can only occur in a paralleled instruction. See Figure 5−2 for legal uses of the localrepeat instruction.
- ❑ The following instructions cannot be used in any form in a local loop code:

  blockrepeat call goto

  idle intr reset

  return trap

- ❑ Nested local repeat block instructions are allowed.

- ❑ The only branch instructions allowed in a localrepeat structure are the branch conditionally instructions (if (cond) goto) with a target branch address pointing to an instruction included within the loop code and being at a higher address than the branching instruction. In this case, the branch conditionally instruction is executed in 3 cycles and the condition is evaluated in the read phase of the pipeline (there is a 1-cycle latency on the condition setting).

Compatibility with C54x devices (C54CM = 1)

When C54CM =1:

- ❑ This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.
- ❑ The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.
- ❑ You can stop an active block-repeat operation by clearing BRAF to 0.
- ❑ Block-repeat control registers for level 1 are not used. Nested block-repeat operations are supported using the C54x convention with context save/restore and BRAF. When an interrupt is acknowledged, unlike C54x device, BRAF is captured into CFCT register, and saved to the stack. You can use a block/local loop instruction in an interrupt without preserving BRAF (while preserving BRC0, RSA0 and REA0).
- ❑ BRAF is automatically cleared to 0 when a far branch (FB) or far call (FCALL) instruction is executed.

Status Bits	Affected by	none
	Affects	none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
localrepeat	A block of instructions is repeated as defined by the content of BRC0 + 1.

	Address	BRC0	RSA0	REA0	BRS1
BRC0 = #3		0003	0000	0000	0000
localrepeat {	004003	?*	4005	400D	?
… …	004005	?	?	?	?
… …	00400D	DTZ**	?	?	?
}		0000	4005	400D	0000

*?: Unchanged

**DTZ: Decrease till zero

Figure 5−2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction

(a) 128-Byte Unaligned Loop—Legal Use

```
          … …                                ; no alignment directive
   localrepeat {
                 1st instruction
          … …                                } 128-byte loop body
                 Last instruction
            }
  next instruction
          … …
```

The entire localrepeat block and the *next instruction* reside in the IBQ, this code is accepted by the assembler.

Figure 5−2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(b) 129-Byte Unaligned Loop with Single Instruction at End of Loop—Illegal Use

```
        … …                                    ; no alignment directive
   localrepeat {
                1st instruction
        … …                                    } 129-byte loop body
                Last instruction
                (nonparalleled = single)
          }
  next instruction
        … …
```

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a nonparalleled (single) instruction, the CPU must confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is rejected by the assembler.

(c) 129-Byte Unaligned Loop with Paralleled Instruction at End of Loop—Legal Use

```
        … …                                    ; no alignment directive
   localrepeat {
                1st instruction
        … …                                    } 129-byte loop body
                Last instruction (paralleled)
          }
  next instruction
        … …
```

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

Figure 5−2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(d) 129-Byte Aligned Loop with Single Instruction at End of Loop—Legal Use

```
          align 4                                ; alignment directive
     localrepeat {
                   1st instruction
             … …                                 } 129-byte loop body
                   Last instruction
                   (nonparalleled = single)
              }
  next instruction
             … …
```

The localrepeat instruction is aligned, so the entire localrepeat block and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

(e) 130-Byte Unaligned Loop—Illegal Use

```
             … …                                 ; no alignment directive
     localrepeat {
                   1st instruction
             … …                                 } 130-byte loop body
                   Last instruction
              }
  next instruction
             … …
```

The localrepeat instruction is not aligned; the entire localrepeat block may not reside in the IBQ. Because the last instruction of the localrepeat block may not reside in the IBQ, this code is rejected by the assembler.

Figure 5−2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(f) 130-Byte Aligned Loop with Single Instruction at End of Loop—Legal Use

```
         align 4                                  ; alignment directive
     nop_16||nop                                  ; 3-byte instruction
   localrepeat {
                  1st instruction
           … …                                    } 130-byte loop body
                  Last instruction
                  (nonparalleled = single)
               }
next instruction
           … …
```

The nop instructions are aligned so the localrepeat instruction, the entire localrepeat block, and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

(g) 132-Byte Aligned Loop with Paralleled Instruction at End of Loop—Legal Use

```
         align 4                                  ; alignment directive
          nop_16                                  ; 2-byte instruction
   localrepeat {
                  1st instruction
           … …                                    } 132-byte loop body
                  Last instruction (paralleled)
               }
next instruction
           … …
```

The nop instruction is aligned, so the localrepeat instruction and the entire localrepeat block reside in the IBQ; the *next instruction* is not fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

Repeat Block of Instructions Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**blockrepeat{}**	Yes	3	1	AD

Opcode | 0000 111E | llll llll | llll llll

Operands none

Description This instruction repeats a block of instructions the number of times specified by:

- ❑ the content of BRC0 + 1, if no loop has already been detected. In this case:
  - ■ In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
  - ■ The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
  - ■ BRC0 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - ■ BRC0 contains 0 after the block-repeat operation has ended.
- ❑ the content of BRS1 + 1, if one level of the loop has already been detected. In this case:
  - ■ BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
  - ■ In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
  - ■ The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
  - ■ BRC1 is decremented at the decode phase of the last instruction of the loop when its content is not equal to 0.
  - ■ BRC1 contains 0 after the block-repeat operation has ended.
  - ■ BRS1 content is not impacted by the block-repeat operation.

Loop structures defined by these instructions must have the following characteristics:

- ❑ The minimum number of instructions executed within one loop iteration is 2.
- ❑ The minimum number of cycles executed within one loop iteration is 2.
- ❑ The maximum loop size is 64K bytes.
- ❑ The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.
- ❑ Since the result of updating BRCx (and BRAF in C54CM = 1) within 3 instruction cycles from the end of the loop is uncertain (effective in the same iteration or the next iteration depending on the pipeline state), this operation is prohibited.
- ❑ C54CM bit in ST1_55 cannot be modified within a block-repeat loop.
- ❑ The following instructions cannot be used as the last instruction in the loop structure:

while (cond && (RPTC < k8)) repeat	repeat(k8)	repeat(CSR), CSR += k4
if (cond) execute(AD_Unit)	repeat(k16)	repeat(CSR), CSR += TAx
if (cond) execute(D_Unit)	repeat(CSR)	repeat(CSR), CSR –= k4

- ❑ See section 1.5 for a list of instructions that cannot be used in the block-repeat loop code.

Compatibility with C54x devices (C54CM = 1)

When C54CM =1:

- ❑ This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.
- ❑ The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.
- ❑ You can stop an active block-repeat operation by clearing BRAF to 0.
- ❑ Block-repeat control registers for level 1 are not used. Nested block-repeat operations are supported using the C54x convention with context save/restore and BRAF. The control-flow context register (CFCT) values are not used.
- ❑ BRAF is automatically cleared to 0 when a far branch (FB) or far call (FCALL) instruction is executed.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
blockrepeat	A block of instructions is repeated as defined by the content of BRC0 + 1. A second loop of instructions is repeated as defined by the content of BRS1 + 1 (BRC1 is loaded with the content of BRS1).

	Address	BRC0	RSA0	REA0	BRS1	BRC1	RSA1	REA1
BRC0 = #3		0003	0000	0000	0000	0000	0000	0000
BRC1 = #1		?*	?	?	0001	0001	?	?
blockrepeat {	004006	?	4009	4017	?	?	?	?
… …	004009	?	?	?	?	?	?	?
localrepeat {	00400B	?	?	?	?	(BRS1)	400D	4015
… …	00400D	?	?	?	?	?	?	?
… …	004015	?	?	?	?	DTZ**	?	?
}								
… …	004017	DTZ**	?	?	?	?	?	?
}		0000	4009	4017	0001	0000	400D	4015

*?: Unchanged

**DTZ: Decrease till zero

RPTCC *Repeat Single Instruction Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**while (**cond **&& (RPTC <** k8**)) repeat**	Yes	3	1	AD

Opcode `0000 000E | xCCC CCCC | kkkk kkkk`

Operands cond, k8

Description This instruction evaluates a single condition defined by the cond field and as long as the condition is true, the next instruction or the next two paralleled instructions is repeated the number of times specified by an 8-bit immediate value, k8 + 1. The maximum number of executions of a given instruction or paralleled instructions is 2^8 –1 (255). See Table 1−3 for a list of conditions.

The 8 LSBs of the repeat counter register (RPTC):

- Are loaded with the immediate value at the address phase of the pipeline.
- Are decremented by 1 in the decode phase of the repeated instruction.
- Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

The 8 MSBs of RPTC:

- Are loaded with the cond code at the address phase of the pipeline.
- Are untouched during the while/repeat structure execution.

At each step of the iteration, the condition defined by the cond field is tested in the execute phase of the pipeline. When the condition becomes false, the instruction repetition stops.

- If the condition becomes false at any execution of the repeated instruction, the 8 LSBs of RPTC are corrected to indicate exactly how many iterations were not performed.
- Since the condition is evaluated in the execute phase of the repeated instruction, when the condition is tested false, some of the succeeding iterations of that repeated instruction may have gone through the address, access, and read phases of the pipeline. Therefore, they may have modified the pointer registers used in the DAGEN units to generate data memory operands addresses in the address phase.

When the while/repeat structure is exited, reading the computed single-repeat register (CSR) content enables you to determine how many instructions have gone through the address phase of the pipeline. You may then use the Repeat Single Instruction Unconditionally instruction [3] to rewind the pointer registers. Note that this must only be performed when a false condition has been met inside the while/repeat structure.

- ❑ The following table provides the 8 LSBs of RPTC and CSR once the while/repeat structure is exited.

If the condition is not met	**RPTC[7:0] content after exiting loop**	**CSR content after exiting loop**
At 1st iteration	RPTCinit + 1	4
At 2nd iteration	RPTCinit	4
At 3rd iteration	RPTC – 1	4
…	…	…
At RPTCinit – 2 iteration	4	3
At RPTCinit – 1 iteration	3	2
At RPTCinit iteration	2	1
At RPTCinit + 1 iteration	1	0
Never	0	0

RPTCinit is the number of requested iterations minus 1.

The repeat single mechanism triggered by this instruction is interruptible. Saving and restoring the RPTC content in ISRs enables you to preserve the while/repeat structure context.

Instead of programming a number of iterations (minus 1) equal to 0, it is recommended that you use the conditional execute() structure.

This instruction cannot be used as the last or the second to last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

In addition, any store−to−memory instruction including push instructions cannot be used in a conditional repeat single mechanism.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

Repeat

This instruction cannot be repeated.

See Also

See the following other related instructions:

- ❑ Repeat Block of Instructions Unconditionally
- ❑ Repeat Single Instruction Unconditionally
- ❑ Repeat Single Instruction Unconditionally and Decrement CSR
- ❑ Repeat Single Instruction Unconditionally and Increment CSR

Example

Syntax	Description
while (AC1 > #0 && (RPTC < #7)) repeat	As long as the content of AC1 is greater than 0 and the repeat counter is not equal to 0, the next single instruction is repeated as defined by the unsigned 8-bit value (7) + 1. At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

while (AC1 > #0 && (RPTC < #7)) repeat	address:	004004
AC1 = AC1 − (T0 * *AR1)		004008
… …		00400B

```
Before                         After
AC1         00 2359 0340       AC1         00 1FC2 7B40
T0                  0340       T0                  0340
*AR1                2354       *AR1                2354
RPTC                4106†      RPTC                0000
```

† At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

RPT *Repeat Single Instruction Unconditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**repeat(**k8**)**	Yes	2	1	AD
[2]	**repeat(**k16**)**	Yes	3	1	AD
[3]	**repeat(CSR)**	Yes	2	1	AD

Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1 or an immediate value, kx + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is 2^{16} –1 (65535).

The repeat single mechanism triggered by these instructions is interruptible.

These instructions cannot be repeated.

These instructions cannot be used as the last instruction in a repeat loop structure.

Two paralleled instructions can be repeated when following the parallelism general rules.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits

Affected by	none
Affects	none

See Also

See the following other related instructions:

❑ Repeat Block of Instructions Unconditionally

❑ Repeat Single Instruction Conditionally

❑ Repeat Single Instruction Unconditionally and Decrement CSR

❑ Repeat Single Instruction Unconditionally and Increment CSR

Repeat Single Instruction Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**repeat(**k8**)**	Yes	2	1	AD
[2]	**repeat(**k16**)**	Yes	3	1	AD

Opcode

k8		0100 110E	kkkk kkkk
k16	0000 110E	kkkk kkkk	kkkk kkkk

Operands kx

Description This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by an immediate value, kx + 1. The repeat counter register (RPTC):

❑ Is loaded with the immediate value in the address phase of the pipeline.

❑ Is decremented by 1 in the decode phase of the repeated instruction.

❑ Contains 0 at the end of the repeat single mechanism.

❑ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated.

Example 1

Syntax	Description

repeat(#3)
AC1 = AC1 + *AR3+ * *AR4+
	The single instruction following the repeat instruction is repeated four times.

```
Before                           After
AC1          00 0000 0000        AC1          00 3376 AD10
AR3                  0200        AR3                  0204
AR4                  0400        AR4                  0404
200                  AC03        200                  AC03
201                  3468        201                  3468
202                  FE00        202                  FE00
203                  23DC        203                  23DC
400                  D768        400                  D768
401                  6987        401                  6987
402                  3400        402                  3400
403                  7900        403                  7900
```

Example 2

Syntax	Description
repeat(#513)	A single instruction is repeated as defined by the unsigned 16-bit value + 1 (513 + 1).

Repeat Single Instruction Unconditionally

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	**repeat(CSR)**	Yes	2	1	AD

Opcode | `0100 100E | xxxx x000`

Operands none

Description This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

❑ Is loaded with CSR content in the address phase of the pipeline.

❑ Is decremented by 1 in the decode phase of the repeated instruction.

❑ Contains 0 at the end of the repeat single mechanism.

❑ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description

repeat(CSR)
AC1 = AC1 + *AR3+ * *AR4+
	The single instruction following the repeat instruction is repeated as defined by the content of CSR + 1.

```
Before                          After
AC1         00 0000 0000        AC1         00 3376 AD10
CSR                 0003        CSR                 0003
AR3                 0200        AR3                 0204
AR4                 0400        AR4                 0404
200                 AC03        200                 AC03
201                 3468        201                 3468
202                 FE00        202                 FE00
203                 23DC        203                 23DC
400                 D768        400                 D768
401                 6987        401                 6987
402                 3400        402                 3400
403                 7900        403                 7900
```

RPTSUB *Repeat Single Instruction Unconditionally and Decrement CSR*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**repeat(CSR), CSR** –= k4	Yes	2	1	X

Opcode

```
|0100 100E|kkkk x011
```

Operands k4

Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

- Is loaded with CSR content in the address phase of the pipeline.
- Is decremented by 1 in the decode phase of the repeated instruction.
- Contains 0 at the end of the repeat single mechanism.
- Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

With the A-unit ALU, this instruction allows the content of CSR to be decremented by k4. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated.

See Also

See the following other related instructions:

- ❑ Repeat Block of Instructions Unconditionally
- ❑ Repeat Single Instruction Conditionally
- ❑ Repeat Single Instruction Unconditionally
- ❑ Repeat Single Instruction Unconditionally and Increment CSR

Example

Syntax	**Description**
repeat(CSR), CSR –= #2	A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is decremented by the unsigned 4-bit value (2).

RPTADD *Repeat Single Instruction Unconditionally and Increment CSR*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**repeat(CSR), CSR** += TAx	Yes	2	1	X
[2]	**repeat(CSR), CSR** –= k4	Yes	2	1	X

Description

These instructions repeat the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is 2^{16} –1 (65535).

With the A-unit ALU, these instructions allow the content of CSR to be incremented. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by these instructions is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

These instructions cannot be repeated.

These instructions cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits

Affected by	none
Affects	none

See Also

See the following other related instructions:

❑ Repeat Block of Instructions Unconditionally

❑ Repeat Single Instruction Conditionally

❑ Repeat Single Instruction Unconditionally

❑ Repeat Single Instruction Unconditionally and Decrement CSR

Repeat Single Instruction Unconditionally and Increment CSR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**repeat(CSR), CSR** += TAx	Yes	2	1	X

Opcode `0100 100E FSSS x001`

Operands TAx

Description This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

❑ Is loaded with CSR content in the address phase of the pipeline.

❑ Is decremented by 1 in the decode phase of the repeated instruction.

❑ Contains 0 at the end of the repeat single mechanism.

❑ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by the content of TAx. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
repeat(CSR), CSR += T1	A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is incremented by the content of temporary register T1.

Repeat Single Instruction Unconditionally and Increment CSR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**repeat(CSR), CSR** += k4	Yes	2	1	X

Opcode `0100 100E kkkk x010`

Operands k4

Description This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):

- ❑ Is loaded with CSR content in the address phase of the pipeline.
- ❑ Is decremented by 1 in the decode phase of the repeated instruction.
- ❑ Contains 0 at the end of the repeat single mechanism.
- ❑ Must not be accessed when it is being decremented in the repeat single mechanism or in parallel with the repeat instruction itself.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by k4. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

Example

Syntax	Description
repeat(CSR), CSR += #2	A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is incremented by the unsigned 4-bit value (2).

RETCC *Return Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles[†]	Pipeline
[1]	**if (**cond**) return**	Yes	3	5/5	R

[†] x/y cycles: x cycles = condition true, y cycles = condition false

Opcode | 0000 001E | xCCC CCCC | xxxx xxxx

Operands cond

Description

This instructions evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a return occurs to the return address of the calling subroutine. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1−3 for a list of conditions.

After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

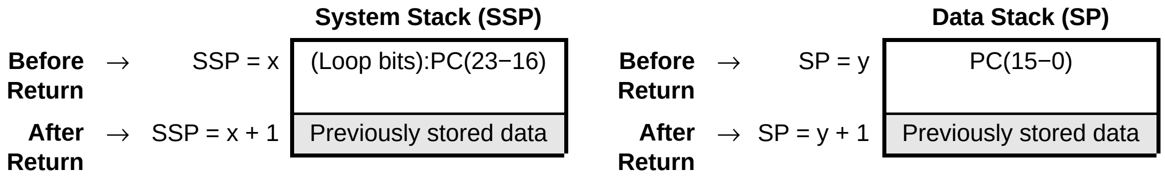
In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

When a return from a subroutine occurs:

❑ The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the read phase of the pipeline.

❑ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the read phase of the pipeline.



Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits

Affected by	ACOVx, CARRY, C54CM, M40, TCx
Affects	ACOVx

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- ❑ Call Conditionally
- ❑ Call Unconditionally
- ❑ Return from Interrupt
- ❑ Return Unconditionally

Example

Syntax	Description
if (ACOV0 = #0) return	The AC0 overflow bit is equal to 0, the program counter (PC) is loaded with the return address of the calling subroutine.

```
Before                          After
ACOV0                   0       ACOV0                  0
PC                              PC         (return address)
SP                              SP
```

RET *Return Unconditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**return**	Yes	2	5	D

Opcode `|0100 100E|xxxx x100`

Operands none

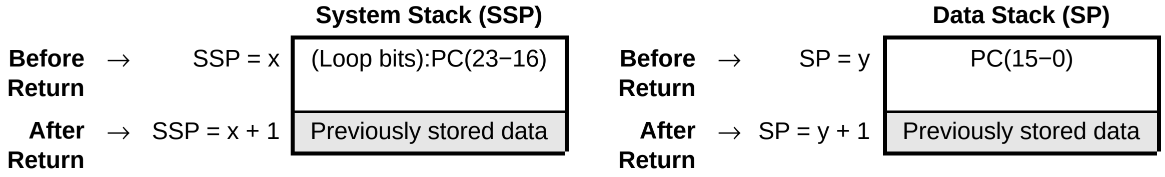
Description This instruction passes control back to the calling subroutine.

After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- ❑ The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the address phase of the pipeline.
- ❑ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.



System Stack (SSP)

Before Return → SSP = x: (Loop bits):PC(23−16)

After Return → SSP = x + 1: Previously stored data

Data Stack (SP)

Before Return → SP = y: PC(15−0)

After Return → SP = y + 1: Previously stored data

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- ❑ Call Conditionally
- ❑ Call Unconditionally
- ❑ Return Conditionally
- ❑ Return from Interrupt

Example

Syntax	**Description**
return	The program counter is loaded with the return address of the calling subroutine.

RETI *Return from Interrupt*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**return_int**	No	2	5	D

Opcode `0100 100E xxxx x101`

Operands none

Description This instruction passes control back to the interrupted task.

After returning from an interrupt service routine (ISR), the CPU automatically restores the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are restored from the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are restored from the stacks (in memory). For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

- The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the address phase of the pipeline.
- The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- The debug status register (DBSTAT) content is popped from the top of SSP. The SSP is incremented by 1 word in the access phase of the pipeline.
- The status register 1 (ST1_55) content is popped from the top of SP. The SP is incremented by 1 word in the access phase of the pipeline.
- The 7 higher bits of status register 0 (ST0_55) concatenated with 9 zeroes are popped from the top of SSP. The SSP is incremented by 1 word in the read phase of the pipeline.

- The status register 2 (ST2_55) content is popped from the top of SP. The SP is incremented by 1 word in the read phase of the pipeline.

			System Stack (SSP)
Before Return	→	SSP = x	(Loop bits):PC(23−16)
		SSP = x + 1	DBSTAT
		SSP = x + 2	ST0_55(15−9)
After Return	→	SSP = x + 3	Previously stored data

			Data Stack (SP)
Before Return	→	SP = y	PC(15−0)
		SP = y + 1	ST1_55
		SP = y + 2	ST2_55
After Return	→	SP = y + 3	Previously stored data

Status Bits Affected by none

Affects none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- Return Conditionally
- Return Unconditionally
- Software Interrupt
- Software Trap

Example

Syntax	Description
return_int	The program counter (PC) is loaded with the return address of the interrupted task.

ROL *Rotate Left Accumulator, Auxiliary, or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	dst = BitOut \\ src \\ BitIn				
[1]	dst = **TC2 \\** src **\\ TC2**	Yes	3	1	X
[2]	dst = **TC2 \\** src **\\ CARRY**	Yes	3	1	X
[3]	dst = **CARRY \\** src **\\ TC2**	Yes	3	1	X
[4]	dst = **CARRY \\** src **\\ CARRY**	Yes	3	1	X

Opcode `| 0001 001E | FSSS xx11 | FDDD 0xvv`

Operands dst, src

Description This instruction performs a bitwise rotation to the MSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at bit position 0
  - BitOut is extracted at a bit position according to M40
- When the destination (dst) operand is an auxiliary or temporary register:
  - if an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation
  - the operation is performed on 16 bits in the A-unit ALU
  - BitIn is inserted at bit position 0
  - BitOut is extracted at bit position 15

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	CARRY, M40, TC2
Affects	CARRY, TC2

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Rotate Right Accumulator, Auxiliary, or Temporary Register Content

Example

Syntax	Description
AC1 = CARRY \\ AC1 \\ TC2	The value of TC2 (1) before the execution of the instruction is shifted into the LSB of AC1 and bit 31 shifted out from AC1 is stored in the CARRY status bit. The rotated value is stored in AC1. Because M40 = 0, the guard bits (39−32) are cleared.

```
Before                              After
AC1          0F E340 5678           AC1          00 C680 ACF1
TC2                     1           TC2                     1
CARRY                   1           CARRY                   1
M40                     0           M40                     0
```

ROR *Rotate Right Accumulator, Auxiliary, or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	dst = BitIn // src // BitOut				
[1]	dst = **TC2 //** src **// TC2**	Yes	3	1	X
[2]	dst = **TC2 //** src **// CARRY**	Yes	3	1	X
[3]	dst = **CARRY //** src **// TC2**	Yes	3	1	X
[4]	dst = **CARRY //** src **// CARRY**	Yes	3	1	X

Opcode

```
|0001 001E|FSSS xx11|FDDD 1xvv
```

Operands dst, src

Description This instruction performs a bitwise rotation to the LSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- ❑ When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at a bit position according to M40
  - BitOut is extracted at bit position 0
- ❑ When the destination (dst) operand is an auxiliary or temporary register:
  - if an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation
  - the operation is performed on 16 bits in the A-unit ALU
  - BitIn is inserted at bit position 15
  - BitOut is extracted at bit position 0

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	CARRY, M40, TC2
Affects	CARRY, TC2

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Rotate Left Accumulator, Auxiliary, or Temporary Register Content

Example

Syntax	Description
AC1 = TC2 // AC0 // TC2	The value of TC2 (1) before the execution of the instruction is shifted into bit 31 of AC0 and the LSB shifted out from AC0 is stored in TC2. The rotated value is stored in AC1. Because M40 = 0, the guard bits (39−32) are cleared.

```
Before                          After
AC0          5F B000 1234       AC0          5F B000 1234
AC1          00 C680 ACF1       AC1          00 D800 091A
TC2                     1       TC2                     0
M40                     0       M40                     0
```

ROUND *Round Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **rnd(**ACx**)**	Yes	2	1	X

Opcode `| 0101 010E | DDSS 101% `

Operands ACx, ACy

Description This instruction performs a rounding of the source accumulator ACx in the D-unit ALU.

❑ The rounding operation depends on RDM:

- When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) is added to the 40-bit source accumulator ACx.
- When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (2^{15}) is added:

```
if( 8000h < bit(15–0) < 10000h)
   add 8000h to the 40-bit source accumulator ACx
else if( bit(15–0) == 8000h)
   if( bit(16) == 1)
   add 8000h to the 40-bit source accumulator ACx
```

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

❑ Addition overflow detection depends on M40.

❑ No addition carry report is stored in CARRY status bit.

❑ If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

Status Bits	Affected by	C54CM, M40, RDM, SATD
	Affects	ACOVy

Repeat This instruction cannot be repeated.

Example

Syntax	Description
AC1 = rnd(AC0)	The content of AC0 is added to 8000h, the 16 LSBs are cleared to 0, and the result is stored in AC1. M40 is cleared to 0, so overflow is detected at bit 31; SATD is cleared to 0, so AC1 is not saturated.

```
Before                         After
AC0          EF 0FF0 8023      AC0          EF 0FF0 8023
AC1          00 0000 0000      AC1          EF 0FF1 0000
RDM                     1      RDM                     1
M40                     0      M40                     0
SATD                    0      SATD                    0
ACOV1                   0      ACOV1                   1
```

SAT *Saturate Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **saturate**(rnd(ACx))	Yes	2	1	X

Opcode | 0101 010E | DDSS 110%

Operands ACx, ACy

Description This instruction performs a saturation of the source accumulator ACx to the 32-bit width frame in the D-unit ALU.

- A rounding is performed if the optional rnd keyword is applied to the instruction. The rounding operation depends on RDM:
  - When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) is added to the 40-bit source accumulator ACx.
  - When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (2^{15}) is added:

    ```
    if( 8000h < bit(15–0) < 10000h)
       add 8000h to the 40-bit source accumulator ACx
    else if( bit(15–0) == 8000h)
       if( bit(16) == 1)
       add 8000h to the 40-bit source accumulator ACx
    ```

    If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.
- An overflow is detected at bit position 31.
- No addition carry report is stored in CARRY status bit.
- If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the destination register is saturated. Saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

Status Bits	Affected by	C54CM, RDM
	Affects	ACOVy

Repeat This instruction can be repeated.

Example 1

Syntax	Description
AC1 = saturate(AC0)	The 32-bit width content of AC0 is saturated and the saturated value, FF 8000 0000, is stored in AC1.

```
Before                          After
AC0          EF 0FF0 8023       AC0          EF 0FF0 8023
AC1          00 0000 0000       AC1          FF 8000 0000
ACOV1                   0       ACOV1                   1
```

Example 2

Syntax	Description
AC1 = saturate(rnd(AC0))	The 32-bit width content of AC0 is saturated. The saturated value, 00 7FFF FFFFh, is rounded, 16 LSBs are cleared, and stored in AC1.

```
Before                          After
AC0          00 7FFF 8000       AC0          00 7FFF 8000
AC1          00 0000 0000       AC1          00 7FFF 0000
RDM                     0       RDM                     0
ACOV1                   0       ACOV1                   1
```

BSET *Set Accumulator, Auxiliary, or Temporary Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(**src, Baddr**)** = **#1**	No	3	1	X

Opcode `|1110 1100|AAAA AAAI|FSSS 000x`

Operands Baddr, src

Description This instruction performs a bit manipulation:

- ❑ In the D-unit ALU, if the source (src) register operand is an accumulator.
- ❑ In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction sets to 1 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

The generated bit address must be within:

- ❑ 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- ❑ 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Clear Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Complement Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Set Memory Bit
- ❑ Set Status Register Bit

Example

Syntax	Description
bit(AC0, AR3) = #1	The bit at the position defined by the content of AR3(4–0) in AC0 is set to 1.

BSET *Set Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(**Smem, src**)** = **#1**	No	3	1	X

Opcode `|1110 0011|AAAA AAAI|FSSS 1100`

Operands Smem, src

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction sets to 1 a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Clear Memory Bit
- ❑ Complement Memory Bit
- ❑ Set Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Set Status Register Bit

Example

Syntax	Description
bit(*AR3, AC0) = #1	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is set to 1.

BSET — *Set Status Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(ST0,** k4**)** = **#1**	Yes	2	1	X
[2]	**bit(ST1,** k4**)** = **#1**	Yes	2	1	X
[3]	**bit(ST2,** k4**)** = **#1**	Yes	2	1	X
[4]	**bit(ST3,** k4**)** = **#1**	Yes	2	1[†]	X

[†] When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Opcode

ST0	0100 011E	kkkk 0001
ST1	0100 011E	kkkk 0011
ST2	0100 011E	kkkk 0101
ST3	0100 011E	kkkk 0111

Operands k4, STx

Description These instructions perform a bit manipulation in the A-unit ALU.

These instructions set to 1 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0, ST1, ST2, or ST3).

It is not allowed to access DP register mapped in ST0 register with bit(ST0, k4) = #1 instruction. Therefore, k4 cannot have a value of 0−8.

It is not allowed to access ASM bit field in ST1 with bit(ST1, k4) = #1 instruction. Therefore, k4 cannot have a value of 0−4.

Compatibility with C54x devices (C54CM = 1)

C55x DSP status registers bit mapping (Figure 5−3, page 5-526) does not correspond to C54x DSP status register bits.

Status Bits

Affected by	none
Affects	Selected status bits

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

❑ Clear Status Register Bit

❑ Set Accumulator, Auxiliary, or Temporary Register Bit

❑ Set Memory Bit

Example

Syntax	Description
bit(ST0, ST0_CARRY) = #1; ST0_CARRY = bit 11	The ST0 bit position defined by the label (ST0_CARRY, bit 11) is set to 1.

```
Before                  After
ST0          0000       ST0          0800
```

Figure 5−3. Status Registers Bit Mapping

ST0_55

15	14	13	12	11	10	9
ACOV2†	**ACOV3**†	**TC1**†	TC2	CARRY	ACOV0	ACOV1
R/W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−0	R/W−0

8	0
DP	
R/W−0	

ST1_55

15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	**M40**†	SATD	SXMD
R/W−0	R/W−0	R/W−1	R/W−0	R/W−1	R/W−0	R/W−0	R/W−1

7	6	5	4	0
C16	FRCT	**C54CM**†	ASM	
R/W−0	R/W−0	R/W−1	R/W−0	

ST2_55

15	14	13	12	11	10	9	8
ARMS	Reserved		DBGM	EALLOW	RDM	Reserved	CDPLC
R/W−0			R/W−1	R/W−0	R/W−0		R/W−0

7	6	5	4	3	2	1	0
AR7LC	AR6LC	AR5LC	AR4LC	AR3LC	AR2LC	AR1LC	AR0LC
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

ST3_55

15	14	13	12	11	8
CAFRZ†	**CAEN**†	**CACLR**†	**HINT**‡	Reserved (always write 1100b)	
R/W−0	R/W−0	R/W−0	R/W−1		

7	6	5	4	3	2	1	0
CBERR†	MPNMC§	**SATA**†	Reserved		CLKOFF	SMUL	SST
R/W−0	R/W−pins	R/W−0			R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; -*n* = Value after reset

† Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

‡ The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

§ The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

SFTCC *Shift Accumulator Content Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = **sftc(**ACx**, TC1)**	Yes	2	1	X
[2]	ACx = **sftc(**ACx**, TC2)**	Yes	2	1	X

Opcode

```
TC1        0101 101E DDxx xx10
TC2        0101 101E DDxx xx11
```

Operands ACx, TCx

Description If the source accumulator ACx(39–0) is equal to 0, this instruction sets the TCx status bit to 1.

If the source accumulator ACx(31–0) has two sign bits:

- this instruction shifts left the 32-bit accumulator ACx by 1 bit
- the TCx status bit is cleared to 0

If the source accumulator ACx(31–0) does not have two sign bits, this instruction sets the TCx status bit to 1.

The sign bits are extracted at bit positions 31 and 30.

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Shift Accumulator Content Logically
- Shift Accumulator, Auxiliary, or Temporary Register Content Logically
- Signed Shift of Accumulator Content
- Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Example 1

Syntax	Description
AC0 = sftc(AC0, TC1)	Because AC0(31) XORed with AC0(30) equals 1, the content of AC0 is not shifted left and TC1 is set to 1.

```
Before                          After
AC0         FF 8765 0055        AC0         FF 8765 0055
TC1                    0        TC1                    1
```

Example 2

Syntax	Description
AC0 = sftc(AC0, TC2)	Because AC0(31) XORed with AC0(30) equals 0, the content of AC0 is shifted left by 1 bit and TC2 is cleared to 0.

```
Before                          After
AC0         00 1234 0000        AC0         00 2468 0000
TC2                    0        TC2                    0
```

SFTL *Shift Accumulator Content Logically*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx **<<<** Tx	Yes	2	1	X
[2]	ACy = ACx **<<<** #SHIFTW	Yes	3	1	X

Description These instructions perform an unsigned shift by an immediate value, SHIFTW, or the content of a temporary register (Tx) in the D-unit shifter.

Status Bits

Affected by	C54CM, M40
Affects	CARRY

See Also See the following other related instructions:

❑ Shift Accumulator Content Conditionally

❑ Shift Accumulator, Auxiliary, or Temporary Register Content Logically

❑ Signed Shift of Accumulator Content

❑ Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Shift Accumulator Content Logically

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx **<<<** Tx	Yes	2	1	X

Opcode | 0101 110E | DDSS ss00

Operands ACx, ACy, Tx

Description This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value. However, no overflow is reported when such saturation occurs.

❑ The operation is performed on 40 bits in the D-unit shifter.

❑ The shift operation is performed according to M40.

❑ The CARRY status bit contains the shifted-out bit. When the shift count is zero, Tx = 0, the CARRY status bit is cleared to 0.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the 6 LSBs of Tx define the shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits	Affected by	C54CM, M40
	Affects	CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 >>> T0	The content of AC0 is logically shifted right by the content of T0 and the result is stored in AC1. There is a right shift because the content of T0 is negative (−6). Because M40 = 0, the guard bits (39−32) are cleared.

```
Before                          After
AC0        5F B000 1234         AC0        5F B000 1234
AC1        00 C680 ACF0         AC1        00 02C0 0048
T0                 FFFA         T0                 FFFA
M40                   0         M40                   0
```

Shift Accumulator Content Logically

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = ACx **<<<** #SHIFTW	Yes	3	1	X

Opcode

```
|0001 000E|DDSS 0111|xxSH IFTW
```

Operands ACx, ACy, SHIFTW

Description This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit.

❑ The operation is performed on 40 bits in the D-unit shifter.

❑ The shift operation is performed according to M40.

❑ The CARRY status bit contains the shifted-out bit. When the shift count is zero, SHIFTW = 0, the CARRY status bit is cleared to 0.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 <<< #31	The content of AC1 is logically shifted left by 31 bits and the result is stored in AC0.

SFTL *Shift Accumulator, Auxiliary, or Temporary Register Content Logically*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst **<<< #1**	Yes	2	1	X
[2]	dst = dst **>>> #1**	Yes	2	1	X

Description These instructions perform an unsigned shift by 1 bit:

❑ In the D-unit shifter, if the destination operand is an accumulator (ACx).

❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).

Status Bits

Affected by	C54CM, M40
Affects	CARRY

See Also See the following other related instructions:

❑ Shift Accumulator Content Conditionally

❑ Shift Accumulator Content Logically

❑ Signed Shift of Accumulator Content

❑ Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Shift Accumulator, Auxiliary, or Temporary Register Content Logically

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst **<<< #1**	Yes	2	1	X

Opcode | 0101 000E | FDDD x000

Operands dst

Description This instruction shifts left by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

❑ When the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit shifter.
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted at a bit position according to M40.

❑ When the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 <<< #1	The content of AC1 is logically shifted left by 1 bit and the result is stored in AC1. Because M40 = 0, the CARRY status bit is extracted at bit 31 and the guard bits (39−32) are cleared.

```
Before                           After
AC1          8F E340 5678        AC1          00 C680 ACF0
CARRY                   0        CARRY                   1
M40                     0        M40                     0
```

Shift Accumulator, Auxiliary, or Temporary Register Content Logically

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst **>>> #1**	Yes	2	1	X

Opcode `0101 000E FDDD x001`

Operands dst

Description This instruction shifts right by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

- ❑ When the destination operand (dst) is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit shifter.
  - ■ 0 is inserted at a bit position according to M40.
  - ■ The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.
- ❑ When the destination operand (dst) is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ 0 is inserted at bit position 15.
  - ■ The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40

Affects CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 >>> #1	The content of AC0 is logically shifted right by 1 bit and the result is stored in AC0.

SFTS *Signed Shift of Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx **<<** Tx	Yes	2	1	X
[2]	ACy = ACx **<<C** Tx	Yes	2	1	X
[3]	ACy = ACx **<<** #SHIFTW	Yes	3	1	X
[4]	ACy = ACx **<<C** #SHIFTW	Yes	3	1	X

Description These instructions perform a signed shift by an immediate value, SHIFTW, or by the content of a temporary register (Tx) in the D-unit shifter.

Status Bits

Affected by	C54CM, M40, SATA, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also See the following other related instructions:

- ❑ Shift Accumulator Content Conditionally
- ❑ Shift Accumulator Content Logically
- ❑ Shift Accumulator, Auxiliary, or Temporary Register Content Logically
- ❑ Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Signed Shift of Accumulator Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx **<<** Tx	Yes	2	1	X

Opcode `|0101 110E|DDSS ss01`

Operands ACx, ACy, Tx

Description This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - ■ if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
  - ■ if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- ❑ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - ■ if M40 =0, comparison is performed versus bit 31
  - ■ if M40 =1, comparison is performed versus bit 39
- ❑ 0 is inserted at bit position 0.
- ❑ The shifted-out bit is extracted according to M40.
- ❑ After shifting, unless otherwise noted, when M40 = 0:
  - ■ overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)

- After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

- These instructions are executed as if M40 status bit was locally set to 1.
- There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.
- The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits

Affected by	C54CM, M40, SATD, SXMD
Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 << T0	The content of AC1 is shifted by the content of T0 and the result is stored in AC0.

Signed Shift of Accumulator Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = ACx **<<C** Tx	Yes	2	1	X

Opcode `0101 110E DDSS ss10`

Operands ACx, ACy, Tx

Description This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - ■ if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
  - ■ if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- ❑ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - ■ if M40 =0, comparison is performed versus bit 31
  - ■ if M40 =1, comparison is performed versus bit 39
- ❑ 0 is inserted at bit position 0.
- ❑ The shifted-out bit is extracted according to M40 and stored in the CARRY status bit. When the shift count is zero, Tx = 0, the CARRY status bit is cleared to 0.
- ❑ After shifting, unless otherwise noted, when M40 = 0:
  - ■ overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)

- ❑ After shifting, unless otherwise noted, when M40 = 1:
  - ■ overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1:

- ❑ These instructions are executed as if M40 status bit was locally set to 1.
- ❑ There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.
- ❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits

Affected by	C54CM, M40, SATD, SXMD
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC2 = AC2 <<C T1	The content of AC2 is shifted left by the content of T1 and the saturated result is stored in AC2. The shifted out bit is stored in the CARRY status bit. Since SATD = 1 and M40 = 0, AC2 = FF 8000 0000 (saturation).

```
Before                           After
AC2          80 AA00 1234        AC2          FF 8000 0000
T1                   0005        T1                   0005
CARRY                   0        CARRY                   1
M40                     0        M40                     0
ACOV2                   0        ACOV2                   1
SXMD                    1        SXMD                    1
SATD                    1        SATD                    1
```

Signed Shift of Accumulator Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = ACx **<<** #SHIFTW	Yes	3	1	X

Opcode `0001 000E | DDSS 0101 | xxSH IFTW`

Operands ACx, ACy, SHIFTW

Description This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content.

❑ The operation is performed on 40 bits in the D-unit shifter.

❑ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the 6-bit value, SHIFTW:

- ■ if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- ■ if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter

❑ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:

- ■ if M40 =0, comparison is performed versus bit 31
- ■ if M40 =1, comparison is performed versus bit 39

❑ 0 is inserted at bit position 0.

❑ The shifted-out bit is extracted according to M40.

❑ After shifting, unless otherwise noted, when M40 = 0:

- ■ overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
- ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)

❑ After shifting, unless otherwise noted, when M40 = 1:

- ■ overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
- ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy

Repeat This instruction can be repeated.

Example 1

Syntax	Description
AC0 = AC1 << #31	The content of AC1 is shifted left by 31 bits and the result is stored in AC0.

Example 2

Syntax	Description
AC0 = AC1 << #–32	The content of AC1 is shifted right by 32 bits and the result is stored in AC0.

Signed Shift of Accumulator Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACy = ACx **<<C** #SHIFTW	Yes	3	1	X

Opcode `|0001 000E|DDSS 0110|xxSH IFTW`

Operands ACx, ACy, SHIFTW

Description This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the 6-bit value, SHIFTW:
  - ■ if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
  - ■ if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- ❑ The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - ■ if M40 =0, comparison is performed versus bit 31
  - ■ if M40 =1, comparison is performed versus bit 39
- ❑ 0 is inserted at bit position 0.
- ❑ The shifted-out bit is extracted according to M40 and stored in the CARRY status bit. When the shift count is zero, SHIFTW = 0, the CARRY status bit is cleared to 0.
- ❑ After shifting, unless otherwise noted, when M40 = 0:
  - ■ overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVy bit is set)
  - ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)

- ❑ After shifting, unless otherwise noted, when M40 = 1:
  - ■ overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
  - ■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 <<C #–5	The content of AC0 is shifted right by 5 bits and the result is stored in AC1. The shifted out bit is stored in the CARRY status bit.

```
Before                        After
AC0       FF 8765 0055        AC0       FF 8765 0055
AC1       00 4321 1234        AC1       FF FC3B 2802
CARRY                0        CARRY                1
SXMD                 1        SXMD                 1
```

SFTS *Signed Shift of Accumulator, Auxiliary, or Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst **>> #1**	Yes	2	1	X
[2]	dst = dst **<< #1**	Yes	2	1	X

Description These instructions perform a shift of 1 bit:

- ❑ In the D-unit shifter, if the destination operand is an accumulator (ACx).
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).

Status Bits

Affected by	C54CM, M40, SATA, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also See the following other related instructions:

- ❑ Shift Accumulator Content Conditionally
- ❑ Shift Accumulator Content Logically
- ❑ Shift Accumulator, Auxiliary, or Temporary Register Content Logically
- ❑ Signed Shift of Accumulator Content

Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst **>> #1**	Yes	2	1	X

Opcode `0100 010E 01x0 FDDD`

Operands dst

Description This instruction shifts right by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

❑ The operation is performed on 40 bits in the D-unit shifter.

❑ When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted right by 1 bit:

- ■ if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- ■ if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter

❑ Bit 39 is extended according to SXMD

❑ The shifted-out bit is extracted at bit position 0.

If the destination operand (dst) is an auxiliary or temporary register:

❑ The operation is performed on 16 bits in the A-unit ALU.

❑ Bit 15 is sign extended.

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits

Affected by	C54CM, M40, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 >> #1	The content of AC0 is shifted right by 1 bit and the result is stored in AC0.

Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst **<< #1**	Yes	2	1	X

Opcode `| 0100 010E | 01x1 FDDD`

Operands dst

Description This instruction shifts left by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted left by 1 bit:
  - if SXMD = 0, 0 is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
  - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39−32) as the input, instead of ACx(39−32), to the shifter
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 =0, comparison is performed versus bit 31
  - if M40 =1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

If the destination operand (dst) is an auxiliary or temporary register:

- ❑ The operation is performed on 16 bits in the A-unit ALU.
- ❑ 0 is inserted at bit position 0.
- ❑ After shifting, unless otherwise noted:
  - ■ overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)
  - ■ if SATA = 1, when an overflow is detected, the destination register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits Affected by C54CM, M40, SATA, SATD, SXMD

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
T2 = T2 << #1	The content of T2 is shifted left by 1 bit and the result is stored in T2.

```
Before                   After
T2             EF27      T2             DE4E
SATA              1      SATA              1
```

INTR *Software Interrupt*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**intr(**k5**)**	No	2	3	D

Opcode

```
|1001 0101|0xxk kkkk
```

Operands k5

Description

This instruction passes control to a specified interrupt service routine (ISR) and interrupts are globally disabled (INTM bit is set to 1 after ST1_55 content is pushed onto the data stack pointer). The ISR address is stored at the interrupt vector address defined by the content of an interrupt vector pointer (IVPD or IVPH) combined with the 5-bit constant, k5. This instruction is executed regardless of the value of INTM bit.

Note:

DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.

Before beginning an ISR, the CPU automatically saves the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the ISR is done.

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are stored to the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are saved to the stacks (in memory). For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

When control is passed to the ISR:

❑ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The status register 2 (ST2_55) content is pushed to the top of SP.

- The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0_55) concatenated with 9 zeroes are pushed to the top of SSP.
- The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1_55) content is pushed to the top of SP.
- The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

When the software interrupt is acknowledged, the corresponding bits in IFR0 and IFR1 are cleared.

			System Stack (SSP)
After Save	→	SSP = x − 3	(Loop bits):PC(23−16)
		SSP = x − 2	DBSTAT
		SSP = x − 1	ST0_55(15−9)
Before Save	→	SSP = x	Previously saved data

			Data Stack (SP)
After Save	→	SP = y − 3	PC(15−0)
		SP = y − 2	ST1_55
		SP = y − 1	ST2_55
Before Save	→	SP = y	Previously saved data

Status Bits

Affected by none

Affects INTM, IFR0, IFR1

Repeat

This instruction cannot be repeated.

See Also

See the following other related instructions:

- Return from Interrupt
- Software Trap

Example

Syntax	Description
intr(#3)	Program control is passed to the specified interrupt service routine. The interrupt vector address is defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value (3).

RESET *Software Reset*

Syntax Characteristics

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	**reset**	No	2	?	D

Opcode `1001 0100 | xxxx xxxx`

Operands none

Description This instruction performs a nonmaskable software reset that can be used any time to put the device in a known state.

The reset instruction affects ST0_55, ST1_55, ST2_55, IFR0, IFR1, and T2 (Table 5−5 and Figure 5−4); status register ST3_55 and interrupt vectors pointer registers (IVPD and IVPH) are not affected. When the reset instruction is acknowledged, the INTM is set to 1 to disable maskable interrupts. All pending interrupts in IFR0 and IFR1 are cleared. The initialization of the system control register, the interrupt vectors pointer, and the peripheral registers is different from the initialization performed by a hardware reset.

Status Bits

Affected by	none
Affects	IFR0, IFR1, ST0_55, ST1_55, ST2_55

Repeat This instruction cannot be repeated.

Table 5−5. Effects of a Software Reset on DSP Registers

Register	Bit	Reset Value	Comment
T2	All	0	All bits are cleared. To ensure TMS320C54x DSP compatibility, instructions affected by ASM bit will use a shift count of 0 (no shift).
IFR0	All	0	All pending interrupt flags are cleared.
IFR1	All	0	All pending interrupt flags are cleared.
ST0_55	ACOV2	0	AC2 overflow flag is cleared.
	ACOV3	0	AC3 overflow flag is cleared.
	TC1	1	Test control flag 1 is cleared.
	TC2	1	Test control flag 2 is cleared.
	CARRY	1	CARRY bit is cleared.
	ACOV0	0	AC0 overflow flag is cleared.
	ACOV1	0	AC1 overflow flag is cleared.
	DP	0	All bits are cleared, data page 0 is selected.
ST1_55	BRAF	0	This flag is cleared.
	CPL	0	The DP (rather than SP) direct addressing mode is selected. Direct accesses to data space are made relative to the data page register (DP).
	XF	1	External flag is set.
	HM	0	When an active $\overline{\text{HOLD}}$ signal forces the DSP to place its external interface in the high-impedance state, the DSP continues executing code from internal memory.
	INTM	1	Maskable interrupts are globally disabled.
	M40	0	32-bit (rather than 40-bit) computation mode is selected for the D unit.
	SATD	0	CPU will not saturate overflow results in the D unit.
	SXMD	1	Sign-extension mode is on.
	C16	0	Dual 16-bit mode is off. For an instruction that is affected by C16, the D-unit ALU performs one 32-bit operation rather than two parallel 16-bit operations.
	FRCT	0	Results of multiply operations are not shifted.
	C54CM	1	TMS320C54x-compatibility mode is on.
	ASM	0	Instructions affected by ASM will use a shift count of 0 (no shift).

Table 5−5. Effects of a Software Reset on DSP Registers (Continued)

Register	Bit	Reset Value	Comment
ST2_55	ARMS	0	When you use the AR indirect addressing mode, the DSP mode (rather than control mode) operands are available.
	DBGM	1	Debug events are disabled.
	EALLOW	0	A program cannot write to the non-CPU emulation registers.
	RDM	0	When an instruction specifies that an operand should be rounded, the CPU uses rounding to the infinite (rather than rounding to the nearest).
	CDPLC	0	CDP is used for linear addressing (rather than circular addressing).
	AR7LC	0	AR7 is used for linear addressing.
	AR6LC	0	AR6 is used for linear addressing.
	AR5LC	0	AR5 is used for linear addressing.
	AR4LC	0	AR4 is used for linear addressing.
	AR3LC	0	AR3 is used for linear addressing.
	AR2LC	0	AR2 is used for linear addressing.
	AR1LC	0	AR1 is used for linear addressing.
	AR0LC	0	AR0 is used for linear addressing.

Figure 5−4. Effects of a Software Reset on Status Registers

ST0_55

15	14	13	12	11	10	9
ACOV2	ACOV3	TC1	TC2	CARRY	ACOV0	ACOV1
0	0	1	1	1	0	0

8 0
DP
0

ST1_55

15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	M40	SATD	SXMD
0	0	1	0	1	0	0	1

7	6	5	4 0
C16	FRCT	C54CM	ASM
0	0	1	0

ST2_55

15	14 13	12	11	10	9	8
ARMS	Reserved	DBGM	EALLOW	RDM	Reserved	CDPLC
0		1	0	0		0

7	6	5	4	3	2	1	0
AR7LC	AR6LC	AR5LC	AR4LC	AR3LC	AR2LC	AR1LC	AR0LC
0	0	0	0	0	0	0	0

TRAP *Software Trap*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**trap(**k5**)**	No	2	?	D

Opcode `1001 0101 1xxk kkkk`

Operands k5

Description This instruction passes control to a specified interrupt service routine (ISR) and this instruction does not affect INTM bit in ST1_55 and DBGM bit in ST2_55. The ISR address is stored at the interrupt vector address defined by the content of an interrupt vector pointer (IVPD or IVPH) combined with the 5-bit constant, k5. This instruction is executed regardless of the value of INTM bit . This instruction is not maskable.

Note:

DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.

Before beginning an ISR, the CPU automatically saves the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the ISR is done.

In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are stored to the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are saved to the stacks (in memory). For fast-return mode operation, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

When control is passed to the ISR:

- ❑ The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The status register 2 (ST2_55) content is pushed to the top of SP.
- ❑ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0_55) concatenated with 9 zeroes are pushed to the top of SSP.
- ❑ The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1_55) content is pushed to the top of SP.
- ❑ The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- ❑ The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- ❑ The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ❑ The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

			System Stack (SSP)
After Save	→	SSP = x − 3	(Loop bits):PC(23−16)
		SSP = x − 2	DBSTAT
		SSP = x − 1	ST0_55(15−9)
Before Save	→	SSP = x	Previously saved data

			Data Stack (SP)
After Save	→	SP = y − 3	PC(15−0)
		SP = y − 2	ST1_55
		SP = y − 1	ST2_55
Before Save	→	SP = y	Previously saved data

Status Bits

Affected by	none
Affects	none

Repeat This instruction cannot be repeated.

See Also See the following other related instructions:

- ❑ Return from Interrupt
- ❑ Software Interrupt

Example

Syntax	Description
trap(5)	Program control is passed to the specified interrupt service routine. The interrupt vector address is defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value (5).

SQR *Square*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx * ACx)	Yes	2	1	X
[2]	ACx = rnd(Smem * Smem)[, T3 = Smem]	No	3	1	X

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are:

- ❑ ACx(32−16)
- ❑ the content of a memory (Smem) location, sign extended to 17 bits

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Multiply
- ❑ Square and Accumulate
- ❑ Square and Subtract
- ❑ Square Distance

Square

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx * ACx)	Yes	2	1	X

Opcode | 0101 010E | DDSS 100% |

Operands ACx, ACy

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32−16).

- ❑ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 * AC1	The content of AC1 is squared and the result is stored in AC0.

Square

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = rnd(Smem * Smem)[, T3 = Smem]	No	3	1	X

Opcode `1101 0011 | AAAA AAAI | U%DD 10xx`

Operands ACx, Smem

Description This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.

- ❑ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- ❑ Multiplication overflow detection depends on SMUL.
- ❑ The 32-bit result of the multiplication is sign extended to 40 bits.
- ❑ Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVx

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = *AR3 * *AR3	The content addressed by AR3 is squared and the result is stored in AC0.

SQA *Square and Accumulate*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + **(**ACx * ACx**)**)	Yes	2	1	X
[2]	ACy = rnd(ACx + **(**Smem * Smem**)**) [,T3 = Smem]	No	3	1	X

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are:

- ❑ ACx(32−16)
- ❑ the content of a memory (Smem) location, sign extended to 17 bits

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Multiply and Accumulate
- ❑ Square
- ❑ Square Distance
- ❑ Square and Subtract

Square and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + **(**ACx * ACx**)**)	Yes	2	1	X

Opcode | 0101 010E | DDSS 001%

Operands ACx, ACy

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32−16).

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 + (AC1 * AC1)	The content of AC1 squared is added to the content of AC0 and the result is stored in AC0.

Square and Accumulate

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]	No	3	1	X

Opcode `| 1101 0010 | AAAA AAAI | U%DD 10SS`

Operands ACx, ACy, Smem

Description This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 + (*AR3 * *AR3)	The content addressed by AR3 squared is added to the content of AC1 and the result is stored in AC0.

SQS *Square and Subtract*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – **(**ACx * ACx**)**)	Yes	2	1	X
[2]	ACy = rnd(ACx – **(**Smem * Smem**)**)[, T3 = Smem]	No	3	1	X

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are:

- ❑ ACx(32−16)
- ❑ the content of a memory (Smem) location, sign extended to 17 bits

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVx, ACOVy

See Also See the following other related instructions:

- ❑ Multiply and Subtract
- ❑ Square
- ❑ Square and Accumulate
- ❑ Square Distance

Square and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy – (ACx * ACx))	Yes	2	1	X

Opcode `|0101 010E|DDSS 010%`

Operands ACx, ACy

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32−16).

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	FRCT, M40, RDM, SATD, SMUL
Affects	ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC1 – (AC0 * AC0)	The content of AC0 squared is subtracted from the content of AC1 and the result is stored in AC1.

Square and Subtract

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd(ACx – (Smem * Smem))[, T3 = Smem]	No	3	1	X

Opcode | 1101 0010 | AAAA AAAI | U%DD 11SS

Operands ACx, ACy, Smem

Description This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.
- Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the accumulator is saturated according to SATD.

This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL

Affects ACOVy

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (*AR3 * *AR3)	The content addressed by AR3 squared is subtracted from the content of AC1 and the result is stored in AC0.

SQDST *Square Distance*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**sqdst(**Xmem, Ymem, ACx, ACy**)**	No	4	1	X

Opcode `| 1000 0110 | XXXM MMYY | YMMM DDDD | 1110 xxn% |`

Operands ACx, ACy, Xmem, Ymem

Description This instruction performs two parallel operations: multiply and accumulate (MAC), and subtract:

```
ACy = ACy + (ACx * ACx),
ACx = (Xmem << #16) − (Ymem << #16)
```

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACx(32−16).

- If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
- Multiplication overflow detection depends on SMUL.
- The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, during the subtraction an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits	Affected by	C54CM, FRCT, M40, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Absolute Distance
- Square
- Square and Accumulate
- Square and Subtract

Example

Syntax	Description
sqdst(*AR0, *AR1, AC0, AC1)	The content of AC0 squared is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0.

```
Before                          After
AC0         FF ABCD 0000        AC0         FF FFAB 0000
AC1         00 0000 0000        AC1         00 1BB1 8229
*AR0                0055        *AR0                0055
*AR1                00AA        *AR1                00AA
ACOV0                  0        ACOV0                  0
ACOV1                  0        ACOV1                  0
CARRY                  0        CARRY                  0
FRCT                   0        FRCT                   0
```

MOV *Store Accumulator Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = **HI(**ACx**)**	No	2	1	X
[2]	Smem = **HI**(rnd**(**ACx**)**)	No	3	1	X
[3]	Smem = **LO(**ACx **<<** Tx**)**	No	3	1	X
[4]	Smem = **HI**(rnd**(**ACx **<<** Tx**)**)	No	3	1	X
[5]	Smem = **LO(**ACx **<<** #SHIFTW**)**	No	3	1	X
[6]	Smem = **HI(**ACx **<<** #SHIFTW**)**	No	3	1	X
[7]	Smem = **HI**(rnd**(**ACx **<<** #SHIFTW**)**)	No	4	1	X
[8]	Smem = **HI**(saturate(uns(rnd**(**ACx**)**)))	No	3	1	X
[9]	Smem = **HI**(saturate(uns(rnd**(**ACx **<<** Tx**)**)))	No	3	1	X
[10]	Smem = **HI**(saturate(uns(rnd**(**ACx **<<** #SHIFTW**)**)))	No	4	1	X
[11]	**dbl(**Lmem**)** = ACx	No	3	1	X
[12]	**dbl(**Lmem**)** = **saturate**(uns**(**ACx**)**)	No	3	1	X
[13]	**HI(**Lmem**)** = **HI(**ACx**) >> #1,** **LO(**Lmem**)** = **LO(**ACx**) >> #1**	No	3	1	X
[14]	Xmem = **LO(**ACx**),** Ymem = **HI(**ACx**)**	No	3	1	X

Description This instruction stores the content of the selected accumulator (ACx) to a memory (Smem) location, to a data memory operand (Lmem), or to dual data memory operands (Xmem and Ymem).

Status Bits

Affected by	C54CM, RDM, SXMD
Affects	none

See Also

See the following other related instructions:

- ❑ Addition with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Multiply with Parallel Store Accumulator Content to Memory
- ❑ Store Accumulator Pair Content to Memory
- ❑ Store Accumulator, Auxiliary, or Temporary Register Content to Memory
- ❑ Store Auxiliary or Temporary Register Pair Content to Memory
- ❑ Subtraction with Parallel Store Accumulator Content to Memory

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = **HI(**ACx**)**	No	2	1	X

Opcode `1011 11SS | AAAA AAAI`

Operands ACx, Smem

Description This instruction stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location. The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(AC0)	The content of AC0(31–16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	Smem = **HI**(rnd**(**ACx**)**)	No	3	1	X

Opcode `| 1110 1000 | AAAA AAAI | SSxx x0x%`

Operands ACx, Smem

Description This instruction stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(uns(rnd(ACx))))
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx)))
- ❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(rnd(AC0))	The content of AC0(31–16) is rounded and stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	Smem = **LO(**ACx **<<** Tx**)**	No	3	1	X

Opcode | `1110 0111 | AAAA AAAI | SSss 00xx`

Operands ACx, Smem, Tx

Description This instruction shifts the accumulator, ACx, by the content of Tx and stores the low part of the accumulator, ACx(15−0), to the memory (Smem) location. If the 16-bit value in Tx is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value. The input operand is shifted in the D-unit shifter according to SXMD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, the 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the 16-bit value in Tx is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
Smem = LO(saturate(uns(ACx << Tx)))

❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
Smem = LO(saturate(ACx << Tx))

Status Bits	Affected by	C54CM, RDM, SST, SXMD
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = LO(AC0 << T0)	The content of AC0 is shifted by the content of T0 and AC0(15−0) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	Smem = **HI**(rnd**(**ACx **<<** Tx**)**)	No	3	1	X

Opcode | 1110 0111 | AAAA AAAI | SSss 10x%

Operands ACx, Smem, Tx

Description This instruction shifts the accumulator, ACx, by the content of Tx and stores high part of the accumulator, ACx(31−16), to the memory (Smem) location. If the 16-bit value in Tx is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value. The input operand is shifted in the D-unit shifter according to SXMD. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, the 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the 16-bit value in Tx is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(uns(rnd(ACx << Tx))))

❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx << Tx)))

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(rnd(AC0 << T0))	The content of AC0 is shifted by the content of T0, is rounded, and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = **LO(**ACx **<<** #SHIFTW**)**	No	3	1	X

Opcode `|1110 1001|AAAA AAAI|SSSH IFTW`

Operands ACx, SHIFTW, Smem

Description This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the low part of the accumulator, ACx(15−0), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
Smem = LO(saturate(uns(ACx << #SHIFTW)))

❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
Smem = LO(saturate(ACx << #SHIFTW))

❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits

Affected by	C54CM, RDM, SST, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = LO(AC0 << #31)	The content of AC0 is shifted left by 31 bits and AC0(15−0) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	Smem = **HI(**ACx **<<** #SHIFTW**)**	No	3	1	X

Opcode `1110 1010 | AAAA AAAI | SSSH IFTW`

Operands ACx, SHIFTW, Smem

Description This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
 Smem = HI(saturate(uns(ACx << #SHIFTW)))

❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
 Smem = HI(saturate(ACx << #SHIFTW))

❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits

Affected by	C54CM, RDM, SST, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(AC0 << #31)	The content of AC0 is shifted left by 31 bits and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	Smem = **HI**(rnd**(**ACx **<<** #SHIFTW**)**)	No	4	1	X

Opcode `|1111 1010|AAAA AAAI|xxSH IFTW|SSxx x0x%`

Operands ACx, SHIFTW, Smem

Description This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation:

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx << #SHIFTW)))
- ❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits

Affected by	C54CM, RDM, SST, SXMD
Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(rnd(AC0 << #31))	The content of AC0 is shifted left by 31 bits, is rounded, and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = **HI**(saturate(uns(rnd**(**ACx**)**)))	No	3	1	X

Opcode | 1110 1000 | AAAA AAAI | SSxx x1u%

Operands ACx, Smem

Description This instruction stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location.

❑ When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.

❑ Input operands are considered signed or unsigned according to uns.

- If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
- If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.

❑ If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.

❑ When a rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:

- If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
- If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the round operation:

❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.

- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx)))

- ❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

- ❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	**Description**
*AR3 = HI(saturate(uns(rnd(AC0))))	The unsigned content of AC0 is rounded, is saturated, and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	Smem = **HI**(saturate(uns(rnd**(**ACx **<<** Tx**)**)))	No	3	1	X

Opcode `| 1110 0111 | AAAA AAAI | SSss 11u% `

Operands ACx, Smem, Tx

Description This instruction shifts the accumulator, ACx, by the content of Tx and stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location. If the 16-bit value in Tx is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

- When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- Input operands are considered signed or unsigned according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- The input operand is shifted in the D-unit shifter according to SXMD.
- When shifting, the sign position of the input operand is compared to the shift quantity.
  - If the optional uns keyword is applied to the input operand, this comparison is performed against bit 32 of the shifted operand.
  - If the optional uns keyword is not applied, this comparison is performed against bit 31 of the shifted operand that is considered signed (the sign is defined by bit 39 of the input operand and SXMD).
  - An overflow is generated accordingly.
- If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- When a shift or rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.

  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1:

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx << Tx)))
- ❑ Overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.
  - If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
  - If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.
- ❑ The 6 LSBs of Tx determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the 16-bit value in Tx is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = HI(saturate(uns(rnd(AC0 << T0))))	The unsigned content of AC0 is shifted by the content of T0, is rounded, is saturated, and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	Smem = **HI**(saturate(uns(rnd**(**ACx **<<** #SHIFTW**)**)))	No	4	1	X

Opcode

```
| 1111  1010 | AAAA  AAAI | uxSH  IFTW | SSxx  x1x%
```

Operands ACx, SHIFTW, Smem

Description This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the high part of the accumulator, ACx(31−16), to the memory (Smem) location.

- When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- Input operands are considered signed or unsigned according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.
- When shifting, the sign position of the input operand is compared to the shift quantity.
  - If the optional uns keyword is applied to the input operand, this comparison is performed against bit 32 of the shifted operand.
  - If the optional uns keyword is not applied, this comparison is performed against bit 31 of the shifted operand that is considered signed (the sign is defined by bit 39 of the input operand and SXMD).
  - An overflow is generated accordingly.
- If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- When a shift or rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.

- If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate, rnd, and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate and rnd keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  Smem = HI(saturate(rnd(ACx << #SHIFTW)))
- ❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	**Description**
*AR3 = HI(saturate(uns(rnd(AC0 << #31))))	The unsigned content of AC0 is shifted left by 31 bits, is rounded, is saturated, and AC0(31−16) is stored at the location addressed by AR3.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	**dbl(**Lmem**)** = ACx	No	3	1	X

Opcode

```
|1110 1011|AAAA AAAI|xxSS 10x0
```

Operands ACx, Lmem

Description This instruction stores the content of the accumulator, ACx(31−0), to the data memory operand (Lmem). The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
dbl(*AR3) = AC0	The content of AC0 is stored at the locations addressed by AR3 and AR3 + 1.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	**dbl(**Lmem**)** = **saturate**(uns(ACx**)**)	No	3	1	X

Opcode | 1110 1011 | AAAA AAAI | xxSS 10u1

Operands ACx, Lmem

Description This instruction stores the content of the accumulator, ACx(31−0), to the data memory operand (Lmem).

- ❑ When the C54CM bit = 0 or the SST bit = 0, the saturate and uns keywords are optional and can be applied or not.
- ❑ Input operands are considered signed or unsigned according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is considered unsigned.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.
- ❑ The 40-bit output of the operation is saturated:
  - ■ If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - ■ If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- ❑ The store operation to the memory location uses the D-unit shifter.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with C54CM = 1, overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user.
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user.

- ❑ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.
- ❑ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

Status Bits Affected by C54CM, RDM, SST, SXMD

Affects none

Repeat This instruction can be repeated.

Example

Syntax	**Description**
dbl(*AR3) = saturate(uns(AC0))	The unsigned content of AC0 is saturated and stored at the locations addressed by AR3 and AR3 + 1.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	**HI(**Lmem**)** = **HI(**ACx**) >> #1,** **LO(**Lmem**)** = **LO(**ACx**) >> #1**	No	3	1	X

Opcode `| 1110 1011 | AAAA AAAI | xxSS 1101`

Operands ACx, Lmem

Description This instruction performs two store operations in parallel and is executed in the D-unit shifter:

- ❑ The 16 highest bits of the accumulator, ACx(31−16), shifted right by 1 bit (bit 31 is sign extended according to SXMD), are stored to the 16 highest bits of the data memory operand (Lmem).
- ❑ The 16 lowest bits, ACx(15−0), shifted right by 1 bit (bit 15 is sign extended according to SXMD), are stored to the 16 lowest bits of the data memory operand (Lmem).

Status Bits

Status Bits	Affected by	SXMD
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
HI(*AR1) = HI(AC0) >> #1, LO(*AR1) = LO(AC0) >> #1	The content of AC0(31–16), shifted right by 1 bit, is stored at the location addressed by AR1 and the content of AC0(15−0), shifted right by 1 bit, is stored at the location addressed by AR1 + 1.

Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	Xmem = **LO(**ACx**),** Ymem = **HI(**ACx**)**	No	3	1	X

Opcode | 1000 0000 | XXXM MMYY | YMMM 10SS

Operands ACx, Xmem, Ymem

Description This instruction performs two store operations in parallel:

- ❑ The 16 lowest bits of the accumulator, ACx(15−0), are stored to data memory operand Xmem.
- ❑ The 16 highest bits, ACx(31−16), are stored to data memory operand Ymem.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR1 = LO(AC0), *AR2 = HI(AC0)	The content of AC0(15–0) is stored at the location addressed by AR1 and the content of AC0(31–16) is stored at the location addressed by AR2.

```
Before                          After
AC0          01 4500 0030       AC0          01 4500 0030
AR1                  0200       AR1                  0200
AR2                  0201       AR2                  0201
200                  3400       200                  0030
201                  0FD3       201                  4500
```

MOV *Store Accumulator Pair Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Lmem = **pair(HI(**ACx**))**	No	3	1	X
[2]	Lmem = **pair(LO(**ACx**))**	No	3	1	X

Description This instruction stores the content of the selected accumulator pair, ACx and AC(x + 1), to a data memory operand (Lmem).

Status Bits

Affected by none

Affects none

See Also See the following other related instructions:

❑ Addition with Parallel Store Accumulator Content to Memory

❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory

❑ Load Accumulator, Auxiliary, or Temporary Register from Memory

❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory

❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory

❑ Multiply with Parallel Store Accumulator Content to Memory

❑ Store Accumulator Content to Memory

❑ Store Accumulator, Auxiliary, or Temporary Register Content to Memory

❑ Store Auxiliary or Temporary Register Pair Content to Memory

❑ Subtraction with Parallel Store Accumulator Content to Memory

Store Accumulator Pair Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Lmem = **pair(HI(**ACx**))**	No	3	1	X

Opcode

```
|1110 1011|AAAA AAAI|xxSS 1110
```

Operands ACx, Lmem

Description This instruction stores the 16 highest bits of the accumulator, ACx(31−16), to the 16 highest bits of the data memory operand (Lmem) and stores the 16 highest bits of AC(x + 1) to the16 lowest bits of data memory operand (Lmem):

- ❑ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ Valid accumulators are AC0/AC1 and AC2/AC3.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR1+ = pair(HI(AC0))	The content of AC0(31–16) is stored at the location addressed by AR1 and the content of AC1(31–16) is stored at the location addressed by AR1 + 1. AR1 is incremented by 2.

```
Before                          After
AC0         01 4500 0030        AC0         01 4500 0030
AC1         03 5644 F800        AC1         03 5644 F800
AR1                 0200        AR1                 0202
200                 3400        200                 4500
201                 0FD3        201                 5644
```

Store Accumulator Pair Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	Lmem = **pair(LO(**ACx**))**	No	3	1	X

Opcode `| 1110 1011 | AAAA AAAI | xxSS 1111`

Operands ACx, Lmem

Description This instruction stores the 16 lowest bits of the accumulator, ACx(15−0), to the 16 highest bits of the data memory operand (Lmem) and stores the 16 lowest bits of AC(x + 1) to the16 lowest bits of data memory operand (Lmem):

- ❑ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ Valid accumulators are AC0/AC1 and AC2/AC3.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*AR3 = pair(LO(AC0))	The content of AC0(15–0) is stored at the location addressed by AR3 and the content of AC1(15–0) is stored at the location addressed by AR3 + 1.

MOV *Store Accumulator, Auxiliary, or Temporary Register Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = src	No	2	1	X
[2]	**high_byte(**Smem**)** = src	No	3	1	X
[3]	**low_byte(**Smem**)** = src	No	3	1	X

Description This instruction stores the content of the selected source (src) register to a memory (Smem) location.

Status Bits

Affected by	none
Affects	none

See Also See the following other related instructions:

- ❑ Addition with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- ❑ Load Accumulator, Auxiliary, or Temporary Register from Memory
- ❑ Multiply and Accumulate with Parallel Store Accumulator Content to Memory
- ❑ Multiply and Subtract with Parallel Store Accumulator Content to Memory
- ❑ Multiply with Parallel Store Accumulator Content to Memory
- ❑ Store Accumulator Content to Memory
- ❑ Store Accumulator Pair Content to Memory
- ❑ Store Auxiliary or Temporary Register Pair Content to Memory
- ❑ Subtraction with Parallel Store Accumulator Content to Memory

Store Accumulator, Auxiliary, or Temporary Register Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = src	No	2	1	X

Opcode `|1100 FSSS|AAAA AAAI`

Operands Smem, src

Description This instruction stores the content of the source (src) register to a memory (Smem) location.

- ❑ When the source register is an accumulator:
  - ■ The low part of the accumulator, ACx(15−0), is stored to the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the source register is an auxiliary or temporary register:
  - ■ The content of the auxiliary or temporary register is stored to the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the A-unit ALU.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
*(#0E10h) = AC0	The content of AC0(15–0) is stored at location E10h.

```
Before                         After
AC0        23 0400 6500        AC0        23 0400 6500
0E10               0000        0E10               6500
```

Store Accumulator, Auxiliary, or Temporary Register Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	**high_byte(**Smem**)** = src	No	3	1	X

Opcode `1110 0101 | AAAA AAAI | FSSS 01x0`

Operands Smem, src

Description This instruction stores the low byte (bits 7−0) of the source (src) register to the high byte (bits 15−8) of the memory (Smem) location. The low byte (bits 7−0) of Smem is unchanged.

- ❑ When the source register is an accumulator:
  - ■ The low part of the accumulator, ACx(7−0), is stored to the high byte of the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the source register is an auxiliary or temporary register:
  - ■ The low part (bits 7−0) content of the auxiliary or temporary register is stored to the high byte of the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the A-unit ALU.
- ❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

Example

Syntax	Description
high_byte(*AR1) = AC1	The content of AC1(7–0) is stored in the high byte (bits 15−8) at the location addressed by AR1.

```
Before                          After
AC1         20 FC00 6788        AC1         20 FC00 6788
AR1                 0200        AR1                 0200
200                 6903        200                 8803
```

Store Accumulator, Auxiliary, or Temporary Register Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	**low_byte(**Smem**)** = src	No	3	1	X

Opcode

```
|1110  0101|AAAA  AAAI|FSSS  01x1
```

Operands Smem, src

Description This instruction stores the low byte (bits 7−0) of the source (src) register to the low byte (bits 7−0) of the memory (Smem) location. The high byte (bits 15−8) of Smem is unchanged.

- ❑ When the source register is an accumulator:
  - ■ The low part of the accumulator, ACx(7−0), is stored to the low byte of the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
- ❑ When the source register is an auxiliary or temporary register:
  - ■ The low part (bits 7−0) content of the auxiliary or temporary register is stored to the low byte of the memory location.
  - ■ The store operation to the memory location uses a dedicated path independent of the A-unit ALU.
- ❑ In this instruction, Smem **cannot** reference to a memory-mapped register (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

Status Bits

	Affected by	none
	Affects	none

Repeat This instruction can be repeated.

Example

Syntax	Description
low_byte(*AR3) = AC0	The content of AC0(7–0) is stored in the low byte (bits 7−0) at the location addressed by AR3.

MOV *Store Auxiliary or Temporary Register Pair Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Lmem = **pair(**TAx**)**	No	3	1	X

Opcode

```
|1110 1011|AAAA AAAI|FSSS 1100
```

Operands TAx, Lmem

Description This instruction stores the content of the temporary or auxiliary register (TAx) to the 16 highest bits of the data memory operand (Lmem) and stores the content of TA(x + 1) to the 16 lowest bits of data memory operand (Lmem):

- The store operation to the memory location uses a dedicated path independent of the A-unit ALU.
- Valid auxiliary registers are AR0, AR2, AR4, and AR6.
- Valid temporary registers are T0 and T2.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Load Accumulator, Auxiliary, or Temporary Register from Memory
- Store Accumulator, Auxiliary, or Temporary Register Content to Memory

Example

Syntax	Description
*AR2 = pair(T0)	The content of T0 is stored at the location addressed by AR2 and the content of T1 is stored at the location addressed by AR2 + 1.

MOV *Store CPU Register Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = **BK03**	No	3	1	X
[2]	Smem = **BK47**	No	3	1	X
[3]	Smem = **BKC**	No	3	1	X
[4]	Smem = **BSA01**	No	3	1	X
[5]	Smem = **BSA23**	No	3	1	X
[6]	Smem = **BSA45**	No	3	1	X
[7]	Smem = **BSA67**	No	3	1	X
[8]	Smem = **BSAC**	No	3	1	X
[9]	Smem = **BRC0**	No	3	1	X
[10]	Smem = **BRC1**	No	3	1	X
[11]	Smem = **CDP**	No	3	1	X
[12]	Smem = **CSR**	No	3	1	X
[13]	Smem = **DP**	No	3	1	X
[14]	Smem = **DPH**	No	3	1	X
[15]	Smem = **PDP**	No	3	1	X
[16]	Smem = **SP**	No	3	1	X
[17]	Smem = **SSP**	No	3	1	X
[18]	Smem = **TRN0**	No	3	1	X
[19]	Smem = **TRN1**	No	3	1	X
[20]	**dbl(**Lmem**)** = **RETA**	No	3	5	X

Opcode See Table 5−6 (page 5-599).

Operands Lmem, Smem

Description These instructions store the content of the selected source CPU register to a memory (Smem) location or a data memory operand (Lmem).

For instructions [9] and [10], the block repeat register (BRCx) is decremented in the address phase of the last instruction of the loop. These instructions have a 3-cycle latency requirement versus the last instruction of the loop.

For instruction [20], the content of the 24-bit RETA register (the return address of the calling subroutine) and the 8-bit CFCT register (active control flow execution context flags of the calling subroutine) are stored to the data memory operand (Lmem):

❑ The content of the CFCT register and the 8 highest bits of the RETA register are stored in the 16 highest bits of Lmem.

❑ The 16 lowest bits of the RETA register are stored in the 16 lowest bits of Lmem.

When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.

Status Bits

Affected by	none
Affects	none

Repeat Instruction [20] cannot be repeated; all other instructions can be repeated.

See Also See the following other related instructions:

❑ Load CPU Register from Memory

❑ Load CPU Register with Immediate Value

❑ Move CPU Register Content to Auxiliary or Temporary Register

❑ Store Accumulator Content to Memory

❑ Store Accumulator Pair Content to Memory

❑ Store Accumulator, Auxiliary, or Temporary Register Content to Memory

❑ Store Auxiliary or Temporary Register Pair Content to Memory

Example 1

Syntax	Description
*AR1+ = SP	The content of the data stack pointer (SP) is stored in the location addressed by AR1. AR1 is incremented by 1.

```
Before                    After
AR1           0200        AR1           0201
SP            0200        SP            0200
200           0000        200           0200
```

Example 2

Syntax	Description
*AR1+ = SSP	The content of the system stack pointer (SSP) is stored in the location addressed by AR1. AR1 is incremented by 1.

```
Before                  After
AR1         0201        AR1         0202
SSP         0000        SSP         0000
201         00FF        201         0000
```

Example 3

Syntax	Description
*AR1+ = TRN0	The content of the transition register (TRN0) is stored in the location addressed by AR1. AR1 is incremented by 1.

```
Before                  After
AR1         0202        AR1         0203
TRN0        3490        TRN0        3490
202         0000        202         3490
```

Example 4

Syntax	Description
*AR1+ = TRN1	The content of the transition register (TRN1) is stored in the location addressed by AR1. AR1 is incremented by 1.

```
Before                  After
AR1         0203        AR1         0204
TRN1        0020        TRN1        0020
203         0000        203         0020
```

Example 5

Syntax	Description
dbl(*AR3) = RETA	The contents of the RETA and CFCT are stored in the location addressed by AR3 and AR3 + 1.

Table 5−6. Opcodes for Store CPU Register Content to Memory Instruction

No.	Syntax	Opcode
[1]	Smem = **BK03**	`1110 0101 AAAA AAAI 1001 10xx`
[2]	Smem = **BK47**	`1110 0101 AAAA AAAI 1010 10xx`
[3]	Smem = **BKC**	`1110 0101 AAAA AAAI 1011 10xx`
[4]	Smem = **BSA01**	`1110 0101 AAAA AAAI 0010 10xx`
[5]	Smem = **BSA23**	`1110 0101 AAAA AAAI 0011 10xx`
[6]	Smem = **BSA45**	`1110 0101 AAAA AAAI 0100 10xx`
[7]	Smem = **BSA67**	`1110 0101 AAAA AAAI 0101 10xx`
[8]	Smem = **BSAC**	`1110 0101 AAAA AAAI 0110 10xx`
[9]	Smem = **BRC0**	`1110 0101 AAAA AAAI x001 11xx`
[10]	Smem = **BRC1**	`1110 0101 AAAA AAAI x010 11xx`
[11]	Smem = **CDP**	`1110 0101 AAAA AAAI 0001 10xx`
[12]	Smem = **CSR**	`1110 0101 AAAA AAAI x000 11xx`
[13]	Smem = **DP**	`1110 0101 AAAA AAAI 0000 10xx`
[14]	Smem = **DPH**	`1110 0101 AAAA AAAI 1100 10xx`
[15]	Smem = **PDP**	`1110 0101 AAAA AAAI 1111 10xx`
[16]	Smem = **SP**	`1110 0101 AAAA AAAI 0111 10xx`
[17]	Smem = **SSP**	`1110 0101 AAAA AAAI 1000 10xx`
[18]	Smem = **TRN0**	`1110 0101 AAAA AAAI x011 11xx`
[19]	Smem = **TRN1**	`1110 0101 AAAA AAAI x100 11xx`
[20]	**dbl(**Lmem**)** = **RETA**	`1110 1011 AAAA AAAI xxxx 01xx`

MOV *Store Extended Auxiliary Register Content to Memory*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**dbl(**Lmem**)** = XAsrc	No	3	1	X

Opcode | 1110 1101 | AAAA AAAI | XSSS 0101

Operands Lmem, XAsrc

Description This instruction moves the content of the 23-bit source register (XARx, XSP, XSSP, XDP, or XCDP) to the 32-bit data memory location addressed by data memory operand (Lmem). The upper 9 bits of the data memory are filled with 0:

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Load Extended Auxiliary Register from Memory
- ❑ Load Extended Auxiliary Register with Immediate Value
- ❑ Modify Extended Auxiliary Register Content
- ❑ Move Extended Auxiliary Register Content

Example

Syntax	Description
dbl(*AR3) = XAR1	The 7 highest bits of XAR1 are moved to the 7 lowest bits of the location addressed by AR3, the 9 highest bits are filled with 0, and the 16 lowest bits of XAR1 are moved to the location addressed by AR3 + 1.

```
Before                         After
XAR1            7F 3492        XAR1            7F 3492
AR3                0200        AR3                0200
200                3765        200                007F
201                0FD3        201                3492
```

SUBC *Subtract Conditionally*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**subc(**Smem, ACx, ACy**)**	No	3	1	X

Opcode `| 1101 1110 | AAAA AAAI | SSDD 0011`

Operands ACx, ACy, Smem

Description This instruction performs a conditional subtraction in the D-unit ALU. The D-unit shifter is not used to perform the memory operand shift.

- The 16-bit data memory operand Smem is sign extended to 40 bits according to SXMD, shifted left by 15 bits, and subtracted from the content of the source accumulator ACx.
  - The shift operation is equivalent to the signed shift instruction.
  - Overflow and carry bit is always detected at bit position 31. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - If an overflow is detected and reported in accumulator overflow bit ACOVy, no saturation is performed on the result of the operation.
- If the result of the subtraction is greater than 0 (bit 39 = 0), the result is shifted left by 1 bit, added to 1, and stored in the destination accumulator ACy.
- If the result of the subtraction is less than 0 (bit 39 = 1), the source accumulator ACx is shifted left by 1 bit and stored in the destination accumulator ACy.

```
if ((ACx – (Smem << #15)) >= 0)
   ACy = (ACx – (Smem << #15)) << #1 + 1
else
   ACy = ACx << #1
```

This instruction is used to make a 16 step 16-bit by 16-bit division. The divisor and the dividend are both assumed to be positive in this instruction. SXMD affects this operation:

- If SXMD = 1, the divisor must have a 0 value in the most significant bit
- If SXMD = 0, any 16-bit divisor value produces the expected result

The dividend, which is in the source accumulator ACx, must be positive (bit 31 = 0) during the computation.

Status Bits	Affected by	SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Addition or Subtraction Conditionally
- Addition or Subtraction Conditionally with Shift
- Addition, Subtraction, or Move Accumulator Content Conditionally
- Dual 16-Bit Subtraction and Addition
- Subtraction
- Subtraction with Parallel Store Accumulator Content to Memory

Example 1

Syntax	Description
subc(*AR1, AC0, AC1)	The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC0. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result generated an overflow and a carry.

```
Before                          After
AC0        23 4300 0000         AC0        23 4300 0000
AC1        00 0000 0000         AC1        46 8400 0001
AR1                 300         AR1                 300
300                 200         300                 200
SXMD                  0         SXMD                  0
ACOV1                 0         ACOV1                 1
CARRY                 0         CARRY                 1
```

Example 2

Syntax	Description

repeat (CSR)
subc(*AR1, AC1, AC1)
	The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result generated a carry.

```
Before                          After
AC1        00 0746 0000         AC1        00 1A18 0007
AR1                 200         AR1                 200
200                0100         200                0100
CSR                   1         CSR                   0
ACOV1                 0         ACOV1                 0
CARRY                 0         CARRY                 1
```

SUB *Subtraction*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst – src	Yes	2	1	X
[2]	dst = dst – k4	Yes	2	1	X
[3]	dst = src – K16	No	4	1	X
[4]	dst = src – Smem	No	3	1	X
[5]	dst = Smem – src	No	3	1	X
[6]	ACy = ACy – **(**ACx **<<** Tx**)**	Yes	2	1	X
[7]	ACy = ACy – **(**ACx **<<** #SHIFTW**)**	Yes	3	1	X
[8]	ACy = ACx – **(**K16 **<< #16)**	No	4	1	X
[9]	ACy = ACx – **(**K16 **<<** #SHFT**)**	No	4	1	X
[10]	ACy = ACx – **(**Smem **<<** Tx**)**	No	3	1	X
[11]	ACy = ACx – **(**Smem **<< #16)**	No	3	1	X
[12]	ACy = **(**Smem **<< #16)** – ACx	No	3	1	X
[13]	ACy = ACx – uns(Smem) – **BORROW**	No	3	1	X
[14]	ACy = ACx – uns(Smem)	No	3	1	X
[15]	ACy = ACx – **(**uns(Smem) **<<** #SHIFTW**)**	No	4	1	X
[16]	ACy = ACx – **dbl(**Lmem**)**	No	3	1	X
[17]	ACy = **dbl(**Lmem**)** – ACx	No	3	1	X
[18]	ACx = **(**Xmem **<< #16)** – **(**Ymem **<< #16)**	No	3	1	X

Description These instructions perform a subtraction operation.

Status Bits

Affected by	CARRY, C54CM, M40, SATA, SATD, SXMD
Affects	ACOVx, ACOVy, CARRY

See Also

See the following other related instructions:

- ❑ Addition
- ❑ Addition or Subtraction Conditionally
- ❑ Addition or Subtraction Conditionally with Shift
- ❑ Addition, Subtraction, or Move Accumulator Content Conditionally
- ❑ Dual 16-Bit Addition and Subtraction
- ❑ Dual 16-Bit Subtractions
- ❑ Dual 16-Bit Subtraction and Addition
- ❑ Subtract Conditionally
- ❑ Subtraction with Parallel Store Accumulator Content to Memory

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst – src	Yes	2	1	X

Opcode `| 0010 011E | FSSS FDDD`

Operands dst, src

Description This instruction performs a subtraction operation between two registers.

❑ When the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

❑ When the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
- Overflow detection is done at bit position 15.
- When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – AC1	The content of AC1 is subtracted from the content of AC0 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst – k4	Yes	2	1	X

Opcode

```
|0100  011E|kkkk  FDDD
```

Operands dst, k4

Description This instruction subtracts a 4-bit unsigned constant, k4, from a register.

❑ When the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit ALU.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

❑ When the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- Overflow detection is done at bit position 15.
- When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – #15	An unsigned 4-bit value (15) is subtracted from the content of AC0 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst = src – K16	No	4	1	X

Opcode

```
|0111  1100|KKKK  KKKK|KKKK  KKKK|FDDD  FSSS
```

Operands dst, K16, src

Description This instruction subtracts a 16-bit signed constant, K16, from a register.

❑ When the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit ALU.
- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
- The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

❑ When the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
- Overflow detection is done at bit position 15.
- When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – FFFFh	A signed 16-bit value (FFFFh) is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src – Smem	No	3	1	X

Opcode

```
|1101 0111|AAAA AAAI|FDDD FSSS
```

Operands dst, Smem, src

Description This instruction subtracts the content of a memory (Smem) location from a register content.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- When the destination operand (dst) is an auxiliary or temporary register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - Overflow detection is done at bit position 15.
  - When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – *AR3	The content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	dst = Smem – src	No	3	1	X

Opcode `1101 1000 | AAAA AAAI | FDDD FSSS`

Operands dst, Smem, src

Description This instruction subtracts a register content from the content of a memory (Smem) location.

- ❑ When the destination operand (dst) is an accumulator:
  - ■ The operation is performed on 40 bits in the D-unit ALU.
  - ■ If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended according to SXMD.
  - ■ The content of the memory location is sign extended to 40 bits according to SXMD.
  - ■ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - ■ When an overflow is detected, the accumulator is saturated according to SATD.
- ❑ When the destination operand (dst) is an auxiliary or temporary register:
  - ■ The operation is performed on 16 bits in the A-unit ALU.
  - ■ If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - ■ Overflow detection is done at bit position 15.
  - ■ When an overflow is detected, the destination register is saturated according to SATA.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	M40, SATA, SATD, SXMD
	Affects	ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	**Description**
AC0 = *AR3 – AC1	The content of AC1 is subtracted from the content addressed by AR3 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACy – **(**ACx **<<** Tx**)**	Yes	2	1	X

Opcode `0101 101E DDSS ss01`

Operands ACx, ACy, Tx

Description This instruction subtracts an accumulator content ACx shifted by the content of Tx from an accumulator content ACy.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- ❑ An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
- ❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits

Affected by	C54CM, M40, SATD, SXMD
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (AC1 << T0)	The content of AC1 shifted by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACy – **(**ACx **<<** #SHIFTW**)**	Yes	3	1	X

Opcode `|0001 000E|DDSS 0100|xxSH IFTW`

Operands ACx, ACy, SHIFTW

Description This instruction subtracts an accumulator content ACx shifted by the 6-bit value, SHIFTW, from an accumulator content ACy.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC0 – (AC1 << #31)	The content of AC1 shifted left by 31 bits is subtracted from the content of AC0 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACy = ACx – **(**K16 **<< #16)**	No	4	1	X

Opcode | 0111 1010 | KKKK KKKK | KKKK KKKK | SSDD 001x

Operands ACx, ACy, K16

Description This instruction subtracts the 16-bit signed constant, K16, shifted left by 16 bits from an accumulator content ACx.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (FFFFh << #16)	A signed 16-bit value (FFFFh) shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ACy = ACx – **(**K16 **<<** #SHFT**)**	No	4	1	X

Opcode

```
|0111 0001|KKKK KKKK|KKKK KKKK|SSDD SHFT
```

Operands ACx, ACy, K16, SHFT

Description This instruction subtracts the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, from an accumulator content ACx.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits		
	Affected by	M40, SATD, SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 – (#9800h << #5)	A signed 16-bit value (9800h) shifted left by 5 bits is subtracted from the content of AC0 and the result is stored in AC1.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ACy = ACx – **(**Smem **<<** Tx**)**	No	3	1	X

Opcode `1101 1101 | AAAA AAAI | SSDD ss01`

Operands ACx, ACy, Smem, Tx

Description This instruction subtracts the content of a memory (Smem) location shifted by the content of Tx from an accumulator content ACx.

- ❑ The operation is performed on 40 bits in the D-unit shifter.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- ❑ An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
- ❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within –32 to +31. When the value is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (*AR3 << T0)	The content addressed by AR3 shifted by the content of T0 is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx – **(**Smem **<< #16)**	No	3	1	X

Opcode `| 1101 1110 | AAAA AAAI | SSDD 0101`

Operands ACx, ACy, Smem

Description This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator content ACx.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are sign extended to 40 bits according to SXMD.
- ❑ The shift operation is equivalent to the signed shift instruction.
- ❑ Overflow detection and CARRY status bit depends on M40. If the result of the subtraction generates a borrow, the CARRY status bit is cleared; otherwise, the CARRY status bit is not affected.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits		
	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (*AR3 << #16)	The content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ACy = **(**Smem **<< #16)** – ACx	No	3	1	X

Opcode `| 1101 1110 | AAAA AAAI | SSDD 0110`

Operands ACx, ACy, Smem

Description This instruction subtracts an accumulator content ACx from the content of a memory (Smem) location shifted left by 16 bits.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

❑ The shift operation is equivalent to the signed shift instruction.

❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = (*AR3 << #16) – AC1	The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ACy = ACx – uns(Smem) – **BORROW**	No	3	1	X

Opcode `| 1101 1111 | AAAA AAAI | SSDD 101u`

Operands ACx, ACy, Smem

Description This instruction subtracts the logical complement of the CARRY status bit (borrow) and the content of a memory (Smem) location from an accumulator content ACx.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are extended to 40 bits according to uns.

- ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.

- ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.

❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	CARRY, M40, SATD, SXMD
	Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC1 = AC0 – uns(*AR1) – BORROW	The complement of the CARRY bit (1) and the unsigned content addressed by AR1 (F000h) are subtracted from the content of AC0 and the result is stored in AC1.

```
Before                              After
AC0          00 EC00 0000           AC0          00 EC00 0000
AC1          00 0000 0000           AC1          00 EBFF 0FFF
AR1                  0302           AR1                  0302
302                  F000           302                  F000
CARRY                   0           CARRY                   1
```

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ACy = ACx – uns(Smem)	No	3	1	X

Opcode

```
|1101 1111|AAAA AAAI|SSDD 111u
```

Operands ACx, ACy, Smem

Description This instruction subtracts the content of a memory (Smem) location from an accumulator content ACx.

- ❑ The operation is performed on 40 bits in the D-unit ALU.
- ❑ Input operands are extended to 40 bits according to uns.
  - ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - ■ If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- ❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- ❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SATD, SXMD
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – uns(*AR3)	The unsigned content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ACy = ACx – **(**uns(Smem) **<< #SHIFTW)**	No	4	1	X

Opcode | 1111 1001 | AAAA AAAI | uxSH IFTW | SSDD 01xx

Operands ACx, ACy, SHIFTW, Smem

Description This instruction subtracts the content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, from an accumulator content ACx.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are extended to 40 bits according to uns.
  - If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.
  - If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – (uns(*AR3) << #31)	The unsigned content addressed by AR3 shifted left by 31 bits is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	ACy = ACx – **dbl(**Lmem**)**	No	3	1	X

Opcode `1110 1101 | AAAA AAAI | SSDD 001n`

Operands ACx, ACy, Lmem

Description This instruction subtracts the content of data memory operand dbl(Lmem) from an accumulator content ACx.

- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1
- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by	M40, SATD, SXMD
Affects	ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = AC1 – dbl(*AR3+)	The content (long word) addressed by AR3 and AR3 + 1 is subtracted from the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	ACy = **dbl(**Lmem**)** – ACx	No	3	1	X

Opcode `| 1110 1101 | AAAA AAAI | SSDD 010x`

Operands ACx, ACy, Lmem

Description This instruction subtracts an accumulator content ACx from the content of data memory operand dbl(Lmem).

❑ The data memory operand dbl(Lmem) addresses are aligned:

- ■ if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
- ■ if Lmem address is odd: most significant word = Lmem, least significant word = Lmem − 1

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = dbl(*AR3) – AC1	The content of AC1 is subtracted from the content (long word) addressed by AR3 and AR3 + 1 and the result is stored in AC0.

Subtraction

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	ACx = **(**Xmem **<< #16)** – **(**Ymem **<< #16)**	No	3	1	X

Opcode `1000 0001 | XXXM MMYY | YMMM 01DD`

Operands ACx, Xmem, Ymem

Description This instruction subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted left 16 bits.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, M40, SATD, SXMD

Affects ACOVx, CARRY

Repeat This instruction can be repeated.

Example

Syntax	Description
AC0 = (*AR3 << #16) – (*AR4 << #16)	The content addressed by AR4 shifted left by 16 bits is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0.

SUB::MOV Subtraction with Parallel Store Accumulator Content to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = **(**Xmem **<< #16)** – ACx, Ymem = **HI(**ACy **<< T2)**	No	4	1	X

Opcode

```
|1000 0111|XXXM MMYY|YMMM SSDD|101x xxxx
```

Operands ACx, ACy, T2, Xmem, Ymem

Description This instruction performs two operations in parallel: subtraction and store.

The first operation subtracts an accumulator content from the content of data memory operand Xmem shifted left by 16 bits.

❑ The operation is performed on 40 bits in the D-unit ALU.

❑ Input operands are sign extended to 40 bits according to SXMD.

❑ The shift operation is equivalent to the signed shift instruction.

❑ Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

❑ When an overflow is detected, the accumulator is saturated according to SATD.

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(31−16) to data memory operand Ymem. If the 16-bit value in T2 is not within –32 to +31, the shift is saturated to –32 or +31 and the shift is performed with this value.

❑ The input operand is shifted in the D-unit shifter according to SXMD.

❑ After the shift, the high part of the accumulator, ACy(31−16), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within –32 to +31. When the 16-bit value in T2 is between –32 to –17, a modulo 16 operation transforms the shift quantity to within –16 to –1.

- ❑ If the SST bit = 1 and the SXMD bit = 0, then the saturate and uns keywords are applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = (Xmem << #16) – ACx,
  Ymem = HI(saturate(uns(ACy << T2)))
- ❑ If the SST bit = 1 and the SXMD bit = 1, then only the saturate keyword is applied to the instruction regardless of the optional keywords selected by the user, with the following syntax:
  ACy = (Xmem << #16) – ACx,
  Ymem = HI(saturate(ACy << T2))

Status Bits Affected by C54CM, M40, RDM, SATD, SST, SXMD

Affects ACOVy, CARRY

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Addition or Subtraction Conditionally
- ❑ Addition or Subtraction Conditionally with Shift
- ❑ Addition, Subtraction, or Move Accumulator Content Conditionally
- ❑ Dual 16-Bit Addition and Subtraction
- ❑ Dual 16-Bit Subtractions
- ❑ Dual 16-Bit Subtraction and Addition
- ❑ Subtraction
- ❑ Subtract Conditionally

Example

Syntax	Description
AC0 = (*AR3 << #16) – AC1, *AR4 = HI(AC0 << T2)	Both instructions are performed in parallel. The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31−16) is stored at the address of AR4.

SWAP *Swap Accumulator Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	swap(ACx, ACy**)**				
[1]	**swap(AC0, AC2)**	Yes	2	1	X
[2]	**swap(AC1, AC3)**	Yes	2	1	X

Opcode

```
swap(AC0, AC2)          | 0101  111E | 0000  0000
swap(AC1, AC3)          | 0101  111E | 0000  0001
```

Operands ACx, ACy

Description This instruction performs parallel moves between two accumulators. These operations are performed in a dedicated datapath independent of the D-unit operators.

This instruction moves the content of the first accumulator (ACx) to the second accumulator (ACy), and reciprocally moves the content of the second accumulator to the first accumulator.

Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Swap Accumulator Pair Content

❑ Swap Auxiliary Register Content

❑ Swap Auxiliary and Temporary Register Content

❑ Swap Temporary Register Content

Example

Syntax	Description
swap(AC0, AC2)	The content of AC0 is moved to AC2 and the content of AC2 is moved to AC0.

```
Before                          After
AC0        01 E500 0030         AC0        00 2800 0200
AC2        00 2800 0200         AC2        01 E500 0030
```

SWAP *Swap Accumulator Pair Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**swap(pair(AC0), pair(AC2))**	Yes	2	1	X

Opcode `|0101 111E|0001 0000`

Operands AC0, AC2

Description This instruction performs two parallel moves between four accumulators (AC0 and AC2, AC1 and AC3) in one cycle. These operations are performed in a dedicated datapath independent of the D-unit operators. Accumulator swapping is performed in the execute phase of the pipeline.

This instruction performs two parallel moves:

❑ the content of AC0 to AC2, and reciprocally the content of AC2 to AC0

❑ the content of AC1 to AC3, and reciprocally the content of AC3 to AC1

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Swap Accumulator Content

❑ Swap Auxiliary Register Pair Content

❑ Swap Auxiliary and Temporary Register Pair Content

❑ Swap Temporary Register Pair Content

Example

Syntax	Description
swap(pair(AC0), pair(AC2))	The following two swap instructions are performed in parallel: the content of AC0 is moved to AC2 and the content of AC2 is moved to AC0, and the content of AC1 is moved to AC3 and the content of AC3 is moved to AC1.

```
Before                          After
AC0        01 E500 0030         AC0        00 2800 0200
AC1        00 FFFF 0000         AC1        00 8800 0800
AC2        00 2800 0200         AC2        01 E500 0030
AC3        00 8800 0800         AC3        00 FFFF 0000
```

SWAP *Swap Auxiliary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	swap(ARx, ARy**)**				
[1]	**swap(AR0, AR1)**	Yes	2	1	AD
[2]	**swap(AR0, AR2)**	Yes	2	1	AD
[3]	**swap(AR1, AR3)**	Yes	2	1	AD

Opcode

```
swap(AR0, AR1)          |0101  111E|0011  1000
swap(AR0, AR2)          |0101  111E|0000  1000
swap(AR1, AR3)          |0101  111E|0000  1001
```

Operands ARx, ARy

Description This instruction performs parallel moves between two auxiliary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the first auxiliary register (ARx) to the second auxiliary register (ARy), and reciprocally moves the content of the second auxiliary register to the first auxiliary register.

Auxiliary register swapping is performed in the address phase of the pipeline.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Swap Accumulator Content
- ❑ Swap Auxiliary and Temporary Register Content
- ❑ Swap Auxiliary Register Pair Content
- ❑ Swap Temporary Register Content

Example

Syntax	Description
swap(AR0, AR2)	The content of AR0 is moved to AR2 and the content of AR2 is moved to AR0.

```
Before                  After
AR0          6500       AR0          0300
AR2          0300       AR2          6500
```

SWAPP *Swap Auxiliary Register Pair Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**swap(pair(AR0), pair(AR2))**	Yes	2	1	AD

Opcode `| 0101 111E | 0001 1000`

Operands AR0, AR2

Description This instruction performs two parallel moves between four auxiliary registers (AR0 and AR2, AR1 and AR3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary register swapping is performed in the address phase of the pipeline.

This instruction performs two parallel moves:

- the content of AR0 to AR2, and reciprocally the content of AR2 to AR0
- the content of AR1 to AR3, and reciprocally the content of AR3 to AR1

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Swap Accumulator Pair Content
- Swap Auxiliary Register Content
- Swap Auxiliary and Temporary Register Pair Content
- Swap Temporary Register Pair Content

Example

Syntax	Description
swap(pair(AR0), pair(AR2))	The following two swap instructions are performed in parallel: the content of AR0 is moved to AR2 and the content of AR2 is moved to AR0, and the content of AR1 is moved to AR3 and the content of AR3 is moved to AR1.

Before		**After**	
AR0	0200	AR0	6788
AR1	0300	AR1	0200
AR2	6788	AR2	0200
AR3	0200	AR3	0300

SWAP *Swap Auxiliary and Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	swap(ARx, Tx**)**				
[1]	**swap(AR4, T0)**	Yes	2	1	AD
[2]	**swap(AR5, T1)**	Yes	2	1	AD
[3]	**swap(AR6, T2)**	Yes	2	1	AD
[4]	**swap(AR7, T3)**	Yes	2	1	AD

Opcode

```
swap(AR4, T0)                 |0101  111E|0000  1100
swap(AR5, T1)                 |0101  111E|0000  1101
swap(AR6, T2)                 |0101  111E|0000  1110
swap(AR7, T3)                 |0101  111E|0000  1111
```

Operands ARx, Tx

Description This instruction performs parallel moves between auxiliary registers and temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the auxiliary register (ARx) to the temporary register (Tx), and reciprocally moves the content of the temporary register to the auxiliary register.

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Swap Accumulator Content
- ❑ Swap Auxiliary Register Content
- ❑ Swap Auxiliary and Temporary Register Pair Content
- ❑ Swap Auxiliary and Temporary Register Pairs Content
- ❑ Swap Temporary Register Content

Example

Syntax	Description
swap(AR4, T0)	The content of AR4 is moved to T0 and the content of T0 is moved to AR4.

```
Before                  After
T0              6500    T0              0300
AR4             0300    AR4             6500
```

SWAPP *Swap Auxiliary and Temporary Register Pair Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	swap(pair(ARx)**, pair**(Tx)**)**				
[1]	**swap(pair(AR4), pair(T0))**	Yes	2	1	AD
[2]	**swap(pair(AR6), pair(T2))**	Yes	2	1	AD

Opcode

```
swap(pair(AR4), pair(T0))        |0101  111E|0001  1100
swap(pair(AR6), pair(T2))        |0101  111E|0001  1110
```

Operands ARx, Tx

Description This instruction performs two parallel moves between two auxiliary registers and two temporary registers in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Instruction [1] performs two parallel moves:

❑ the content of AR4 to T0, and reciprocally the content of T0 to AR4

❑ the content of AR5 to T1, and reciprocally the content of T1 to AR5

Instruction [2] performs two parallel moves:

❑ the content of AR6 to T2, and reciprocally the content of T2 to AR6

❑ the content of AR7 to T3, and reciprocally the content of T3 to AR7

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Swap Accumulator Pair Content

❑ Swap Auxiliary Register Pair Content

❑ Swap Auxiliary and Temporary Register Content

❑ Swap Auxiliary and Temporary Register Pairs Content

❑ Swap Temporary Register Pair Content

Example

Syntax	Description
swap(pair(AR4), pair(T0))	The following two swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, and the content of AR5 is moved to T1 and the content of T1 is moved to AR5.

```
Before                   After
AR4          0200        AR4          6788
AR5          0300        AR5          0200
T0           6788        T0           0200
T1           0200        T1           0300
```

SWAP4 *Swap Auxiliary and Temporary Register Pairs Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**swap(block(AR4), block(T0))**	Yes	2	1	AD

Opcode

```
| 0101  111E | 0010  1100
```

Operands AR4, T0

Description This instruction performs four parallel moves between four auxiliary registers (AR4, AR5, AR6, and AR7) and four temporary registers (T0, T1, T2, and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

This instruction performs four parallel moves:

- the content of AR4 to T0, and reciprocally the content of T0 to AR4
- the content of AR5 to T1, and reciprocally the content of T1 to AR5
- the content of AR6 to T2, and reciprocally the content of T2 to AR6
- the content of AR7 to T3, and reciprocally the content of T3 to AR7

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Swap Auxiliary and Temporary Register Content
- Swap Auxiliary and Temporary Register Pair Content

Example

Syntax	Description
swap (block(AR4), block(T0))	The following four swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, the content of AR5 is moved to T1 and the content of T1 is moved to AR5, the content of AR6 is moved to T2 and the content of T2 is moved to AR6, and the content of AR7 is moved to T3 and the content of T3 is moved to AR7.

Before		**After**	
AR4	0200	AR4	0030
AR5	0300	AR5	0200
AR6	0240	AR6	3400
AR7	0400	AR7	0FD3
T0	0030	T0	0200
T1	0200	T1	0300
T2	3400	T2	0240
T3	0FD3	T3	0400

SWAP *Swap Temporary Register Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	swap(Tx, Ty**)**				
[1]	**swap(T0, T2)**	Yes	2	1	AD
[2]	**swap(T1, T3)**	Yes	2	1	AD

Opcode

```
swap(T0, T2)          |0101  111E|0000  0100
swap(T1, T3)          |0101  111E|0000  0101
```

Operands Tx, Ty

Description This instruction performs parallel moves between two temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the first temporary register (Tx) to the second temporary register (Ty), and reciprocally moves the content of the second temporary register to the first temporary register.

Temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by	none
Affects	none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Swap Accumulator Content

❑ Swap Auxiliary Register Content

❑ Swap Auxiliary and Temporary Register Content

❑ Swap Temporary Register Pair Content

Example

Syntax	Description
swap(T0, T2)	The content of T0 is moved to T2 and the content of T2 is moved to T0.

```
Before                 After
T0          6500       T0          0300
T2          0300       T2          6500
```

SWAPP *Swap Temporary Register Pair Content*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**swap(pair(T0), pair(T2))**	Yes	2	1	AD

Opcode | `0101 111E | 0001 0100`

Operands T0, T2

Description This instruction performs two parallel moves between four temporary registers (T0 and T2, T1 and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Temporary register swapping is performed in the address phase of the pipeline.

This instruction performs two parallel moves:

- ❑ the content of T0 to T2, and reciprocally the content of T2 to T0
- ❑ the content of T1 to T3, and reciprocally the content of T3 to T1

Status Bits Affected by none

Affects none

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Swap Accumulator Pair Content
- ❑ Swap Auxiliary Register Pair Content
- ❑ Swap Auxiliary and Temporary Register Pair Content
- ❑ Swap Temporary Register Content

Example

Syntax	Description
swap(pair(T0), pair(T2))	The following two swap instructions are performed in parallel: the content of T0 is moved to T2 and the content of T2 is moved to T0, and the content of T1 is moved to T3 and the content of T3 is moved to T1.

```
Before                     After
T0            0200         T0            6788
T1            0300         T1            0200
T2            6788         T2            0200
T3            0200         T3            0300
```

BTST *Test Accumulator, Auxiliary, or Temporary Register Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = **bit(**src, Baddr**)**	No	3	1	X
[2]	**TC2** = **bit(**src, Baddr**)**	No	3	1	X

Opcode

`TC1`	`1110 1100`	`AAAA AAAI`	`FSSS 1000`
`TC2`	`1110 1100`	`AAAA AAAI`	`FSSS 1001`

Operands Baddr, src, TCx

Description This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests a single bit of the source register location as defined by the bit addressing mode, Baddr. The tested bit is copied into the selected TCx status bit. The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, 0 is stored into the selected TCx status bit.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- Clear Accumulator, Auxiliary, or Temporary Register Bit
- Complement Accumulator, Auxiliary, or Temporary Register Bit
- Set Accumulator, Auxiliary, or Temporary Register Bit
- Test Accumulator, Auxiliary, or Temporary Register Bit Pair
- Test Memory Bit

Example

Syntax	Description
TC1 = bit(T0, @#12)	The bit at the position defined by the register bit address (12) in T0 is tested and the tested bit is copied into TC1.

```
Before                     After
T0              FE00       T0              FE00
TC1                0       TC1                1
```

BTSTP *Test Accumulator, Auxiliary, or Temporary Register Bit Pair*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**bit(**src**, pair(**Baddr**))**	No	3	1	X

Opcode `|1110 1100|AAAA AAAI|FSSS 010x`

Operands Baddr, src

Description This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests two consecutive bits of the source register location as defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits are copied into status bits TC1 and TC2:

- TC1 tests the bit that is defined by Baddr
- TC2 tests the bit defined by Baddr + 1

The generated bit address must be within:

- 0–38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–38:
  - If the generated bit address is 39, bit 39 of the register is stored into TC1 and 0 is stored into TC2.
  - In all other cases, 0 is stored into TC1 and TC2.
- 0–14 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position). If the generated bit address is not within 0–14:
  - If the generated bit address is 15, bit 15 of the register is stored into TC1 and 0 is stored into TC2.
  - In all other cases, 0 is stored into TC1 and TC2.

Status Bits

Affected by	none
Affects	TC1, TC2

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Clear Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Complement Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Set Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Test Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Test Memory Bit

Example

Syntax	Description
bit(AC0, pair(AR1(T0)))	The bit at the position defined by the content of AR1(T0) in AC0 is tested and the tested bit is copied into TC1. The bit at the position defined by the content of AR1(T0) + 1 in AC0 is tested and the tested bit is copied into TC2.

```
Before                          After
AC0          E0 1234 0000       AC0          E0 1234 0000
AR1                  0026       AR1                  0026
T0                   0001       T0                   0001
TC1                     0       TC1                     1
TC2                     0       TC2                     0
```

BTST Test Memory Bit

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = **bit(**Smem, src**)**	No	3	1	X
[2]	TCx = **bit(**Smem, k4**)**	No	3	1	X

Description These instructions perform a bit manipulation in the A-unit ALU. These instructions test a single bit of a memory (Smem) location. The bit tested is defined by either the content of the source (src) operand or a 4-bit immediate value, k4. The tested bit is copied into the selected TCx status bit.

For instruction [1], the generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by	none
Affects	TCx

See Also See the following other related instructions:

- ❑ Clear Memory Bit
- ❑ Complement Memory Bit
- ❑ Set Memory Bit
- ❑ Test Accumulator, Auxiliary, or Temporary Register Bit
- ❑ Test Accumulator, Auxiliary, or Temporary Register Bit Pair
- ❑ Test and Clear Memory Bit
- ❑ Test and Complement Memory Bit
- ❑ Test and Set Memory Bit

Test Memory Bit

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1a]	**TC1** = **bit(**Smem, src**)**	No	3	1	X
[1b]	**TC2** = **bit(**Smem, src**)**	No	3	1	X

Opcode

```
TC1        |1110  0000|AAAA  AAAI|FSSS  xxx0
TC2        |1110  0000|AAAA  AAAI|FSSS  xxx1
```

Operands Smem, src, TCx

Description This instruction performs a bit manipulation in the A-unit ALU. This instruction tests a single bit of a memory (Smem) location. The bit tested is defined by the content of the source (src) operand. The tested bit is copied into the selected TCx status bit.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC1 = bit(*AR0, AC0)	The bit at the position defined by AC0(3–0) in the content addressed by AR0 is tested and the tested bit is copied into TC1.

```
Before                         After
AC0          00 0000 0008      AC0          00 0000 0008
*AR0                 00C0      *AR0                 00C0
TC1                     0      TC1                     0
```

Test Memory Bit

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2a]	**TC1** = **bit(**Smem, k4**)**	No	3	1	X
[2b]	**TC2** = **bit(**Smem, k4**)**	No	3	1	X

Opcode

```
TC1        |1101  1100|AAAA  AAAI|kkkk  xx00
TC2        |1101  1100|AAAA  AAAI|kkkk  xx01
```

Operands k4, Smem, TCx

Description This instruction performs a bit manipulation in the A-unit ALU. This instruction tests a single bit of a memory (Smem) location. The bit tested is defined by a 4-bit immediate value, k4. The tested bit is copied into the selected TCx status bit.

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

Example

Syntax	Description
TC1 = bit(*AR3, #12)	The bit at the position defined by an unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1.

BTSTCLR *Test and Clear Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = **bit(**Smem, k4**), bit(**Smem, k4**)** = **#0**	No	3	1	X
[2]	**TC2** = **bit(**Smem, k4**), bit(**Smem, k4**)** = **#0**	No	3	1	X

Opcode

```
TC1                |1110 0011|AAAA AAAI|kkkk 010x
TC2                |1110 0011|AAAA AAAI|kkkk 011x
```

Operands k4, Smem, TCx

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is cleared to 0 in Smem.

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Clear Memory Bit

❑ Complement Memory Bit

❑ Set Memory Bit

❑ Test and Complement Memory Bit

❑ Test and Set Memory Bit

❑ Test Memory Bit

Example

Syntax	Description
TC1 = bit(*AR3, #12), bit(*AR3, #12) = #0	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is cleared to 0.

BTSTNOT *Test and Complement Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = **bit(**Smem, k4**), cbit(**Smem, k4**)**	No	3	1	X
[2]	**TC2** = **bit(**Smem, k4**), cbit(**Smem, k4**)**	No	3	1	X

Opcode

```
TC1          |1110  0011|AAAA  AAAI|kkkk  100x
TC2          |1110  0011|AAAA  AAAI|kkkk  101x
```

Operands k4, Smem, TCx

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location and the tested bit is copied into status bit TCx and is complemented in Smem.

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

- ❑ Clear Memory Bit
- ❑ Complement Memory Bit
- ❑ Set Memory Bit
- ❑ Test and Clear Memory Bit
- ❑ Test and Set Memory Bit
- ❑ Test Memory Bit

Example

Syntax	Description
TC1 = bit(*AR0, #12), cbit(*AR0, #12)	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR0 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR0 is complemented.

```
Before                  After
*AR0          0040      *AR0          1040
TC1              0      TC1              0
```

BTSTSET *Test and Set Memory Bit*

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	**TC1** = **bit(**Smem, k4**), bit(**Smem, k4**)** = **#1**	No	3	1	X
[2]	**TC2** = **bit(**Smem, k4**), bit(**Smem, k4**)** = **#1**	No	3	1	X

Opcode

`TC1`	`1110 0011`	`AAAA AAAI`	`kkkk 000x`
`TC2`	`1110 0011`	`AAAA AAAI`	`kkkk 001x`

Operands k4, Smem, TCx

Description This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is set to 1 in Smem.

Status Bits

Affected by	none
Affects	TCx

Repeat This instruction can be repeated.

See Also See the following other related instructions:

❑ Clear Memory Bit

❑ Complement Memory Bit

❑ Set Memory Bit

❑ Test and Clear Memory Bit

❑ Test and Complement Memory Bit

❑ Test Memory Bit

Example

Syntax	Description
TC1 = bit(*AR3, #12), bit(*AR3, #12) = #1	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is set to 1.

Chapter 6

Instruction Opcodes in Sequential Order

This chapter provides the opcode in sequential order for each TMS320C55x™ DSP instruction syntax.

Topic **Page**

6.1 Instruction Set Opcodes

Table 6−1 lists the opcodes of the instruction set. See Table 6−2 (page 6-19) for a list of the symbols and abbreviations used in the instruction set opcode. See Table 1−1 (page 1-2) and Table 1−2 (page 1-6) for a list of the terms, symbols, and abbreviations used in the algebraic syntax.

Table 6−1. Instruction Set Opcodes

Opcode	Algebraic syntax
`0000000E xCCCCCCC kkkkkkkk`	while (cond && (RPTC < k8)) repeat
`0000001E xCCCCCCC xxxxxxxx`	if (cond) return
`0000010E xCCCCCCC LLLLLLLL`	if (cond) goto L8
`0000011E LLLLLLLL LLLLLLLL`	goto L16
`0000100E LLLLLLLL LLLLLLLL`	call L16
`0000110E kkkkkkkk kkkkkkkk`	repeat(k16)
`0000111E llllllll llllllll`	blockrepeat{}
`0001000E DDSS0000 xxSHIFTW`	ACy = ACy & (ACx <<< #SHIFTW)
`0001000E DDSS0001 xxSHIFTW`	ACy = ACy | (ACx <<< #SHIFTW)
`0001000E DDSS0010 xxSHIFTW`	ACy = ACy ^ (ACx <<< #SHIFTW)
`0001000E DDSS0011 xxSHIFTW`	ACy = ACy + (ACx << #SHIFTW)
`0001000E DDSS0100 xxSHIFTW`	ACy = ACy − (ACx << #SHIFTW)
`0001000E DDSS0101 xxSHIFTW`	ACy = ACx << #SHIFTW
`0001000E DDSS0110 xxSHIFTW`	ACy = ACx <<C #SHIFTW
`0001000E DDSS0111 xxSHIFTW`	ACy = ACx <<< #SHIFTW
`0001000E xxSS1000 xxddxxxx`	Tx = exp(ACx)
`0001000E DDSS1001 xxddxxxx`	ACy = mant(ACx), Tx = −exp(ACx)
`0001000E xxSS1010 SSddxxxt`	Tx = count(ACx,ACy,TCx)
`0001000E DDSS1100 SSDDnnnn`	max_diff(ACx,ACy,ACz,ACw)
`0001000E DDSS1101 SSDDxxxr`	max_diff_dbl(ACx,ACy,ACz,ACw,TRNx)
`0001000E DDSS1110 SSDDxxxx`	min_diff(ACx,ACy,ACz,ACw)
`0001000E DDSS1111 SSDDxxxr`	min_diff_dbl(ACx,ACy,ACz,ACw,TRNx)
`0001001E FSSScc00 FDDDxuxt`	TCx = uns(src RELOP dst)
`0001001E FSSScc01 FDDD0utt`	TCx = TCy & uns(src RELOP dst)
`0001001E FSSScc01 FDDD1utt`	TCx = !TCy & uns(src RELOP dst)
`0001001E FSSScc10 FDDD0utt`	TCx = TCy | uns(src RELOP dst)
`0001001E FSSScc10 FDDD1utt`	TCx = !TCy | uns(src RELOP dst)
`0001001E FSSSxx11 FDDD0xvv`	dst = BitOut \\ src \\ BitIn

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
0001001E FSSSxx11 FDDD1xvv	dst = BitIn // src // BitOut
0001010E FSSSxxxx FDDD0000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD0001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD0010	mar(TAy – TAx)
0001010E PPPPPPPP FDDD0100	mar(TAx + P8)
0001010E PPPPPPPP FDDD0101	mar(TAx = P8)
0001010E PPPPPPPP FDDD0110	mar(TAx – P8)
0001010E FSSSxxxx FDDD1000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD1001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD1010	mar(TAy – TAx)
0001010E PPPPPPPP FDDD1100	mar(TAx + P8)
0001010E PPPPPPPP FDDD1101	mar(TAx = P8)
0001010E PPPPPPPP FDDD1110	mar(TAx – P8)
0001010E XACS0001 XACD0000 (Note: for DAG_X)	mar(XACdst + XACsrc)
0001010E XACS0001 XACD0001 (Note: for DAG_X)	mar(XACdst = XACsrc)
0001010E XACS0001 XACD0010 (Note: for DAG_X)	mar(XACdst − XACsrc)
0001010E XACS0001 XACD1000 (Note: for DAG_Y)	mar(XACdst + XACsrc)
0001010E XACS0001 XACD1001 (Note: for DAG_Y)	mar(XACdst = XACsrc)
0001010E XACS0001 XACD1010 (Note: for DAG_Y)	mar(XACdst − XACsrc)
0001011E xxxxxkkk kkkk0000	DPH = k7
0001011E xxxkkkkk kkkk0011	PDP = k9
0001011E kkkkkkkk kkkk0100	BK03 = k12
0001011E kkkkkkkk kkkk0101	BK47 = k12
0001011E kkkkkkkk kkkk0110	BKC = k12
0001011E kkkkkkkk kkkk1000	CSR = k12
0001011E kkkkkkkk kkkk1001	BRC0 = k12
0001011E kkkkkkkk kkkk1010	BRC1 = k12
0001100E kkkkkkkk FDDDFSSS	dst = src & k8
0001101E kkkkkkkk FDDDFSSS	dst = src | k8

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
0001110E kkkkkkkk FDDDFSSS	dst = src ^ k8
0001111E KKKKKKKK SSDDxx0%	ACy = rnd(ACx * K8)
0001111E KKKKKKKK SSDDss1%	ACy = rnd(ACx + (Tx * K8))
0010000E	nop
0010001E FSSSFDDD	dst = src
0010010E FSSSFDDD	dst = dst + src
0010011E FSSSFDDD	dst = dst − src
0010100E FSSSFDDD	dst = dst & src
0010101E FSSSFDDD	dst = dst | src
0010110E FSSSFDDD	dst = dst ^ src
0010111E FSSSFDDD	dst = max(src, dst)
0011000E FSSSFDDD	dst = min(src, dst)
0011001E FSSSFDDD	dst = |src|
0011010E FSSSFDDD	dst = −src
0011011E FSSSFDDD	dst = ~src
0011100E FSSSFDDD (Note: FSSS = src1, FDDD = src2)	push(src1, src2)
0011101E FSSSFDDD (Note: FSSS = dst1, FDDD = dst2)	dst1, dst2 = pop()
0011110E kkkkFDDD	dst = k4
0011111E kkkkFDDD	dst = −k4
0100000E kkkkFDDD	dst = dst + k4
0100001E kkkkFDDD	dst = dst − k4
01000101 11110010	lock()
0100010E 00SSFDDD	TAx = HI(ACx)
0100010E 01x0FDDD	dst = dst >> #1
0100010E 01x1FDDD	dst = dst << #1
0100010E 1000FDDD	TAx = SP
0100010E 1001FDDD	TAx = SSP
0100010E 1010FDDD	TAx = CDP
0100010E 1100FDDD	TAx = BRC0
0100010E 1101FDDD	TAx = BRC1
0100010E 1110FDDD	TAx = RPTC

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`0100011E kkkk0000`	bit(ST0, k4) = #0
`0100011E kkkk0001`	bit(ST0, k4) = #1
`0100011E kkkk0010`	bit(ST1, k4) = #0
`0100011E kkkk0011`	bit(ST1, k4) = #1
`0100011E kkkk0100`	bit(ST2, k4) = #0
`0100011E kkkk0101`	bit(ST2, k4) = #1
`0100011E kkkk0110`	bit(ST3, k4) = #0
`0100011E kkkk0111`	bit(ST3, k4) = #1
`0100100E xxxxx000`	repeat(CSR)
`0100100E FSSSx001`	repeat(CSR), CSR += TAx
`0100100E kkkkx010`	repeat(CSR), CSR += k4
`0100100E kkkkx011`	repeat(CSR), CSR −= k4
`0100100E xxxxx100`	return
`01001000 xxxxx100`	return_int
`0100101E 0LLLLLLL`	goto L7
`0100101E 1lllllll`	localrepeat{}
`0100110E kkkkkkkk`	repeat(k8)
`0100111E KKKKKKKK`	SP = SP + K8
`0101000E FDDDx000`	dst = dst <<< #1
`0101000E FDDDx001`	dst = dst >>> #1
`0101000E FDDDx010`	dst = pop()
`0101000E xxDDx011`	ACx = dbl(pop())
`0101000E FSSSx110`	push(src)
`0101000E xxSSx111`	dbl(push(ACx))
`0101000E XDDD0100`	xdst = popboth()
`0101000E XSSS0101`	pshboth(xsrc)
`0101001E FSSS00DD`	HI(ACx) = TAx
`0101001E FSSS1000`	SP = TAx
`0101001E FSSS1001`	SSP = TAx
`0101001E FSSS1010`	CDP = TAx
`0101001E FSSS1100`	CSR = TAx
`0101001E FSSS1101`	BRC1 = TAx
`0101001E FSSS1110`	BRC0 = TAx

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`0101010E DDSS000%`	ACy = rnd(ACy + |ACx|)
`0101010E DDSS001%`	ACy = rnd(ACy + (ACx * ACx))
`0101010E DDSS010%`	ACy = rnd(ACy − (ACx * ACx))
`0101010E DDSS011%`	ACy = rnd(ACy * ACx)
`0101010E DDSS100%`	ACy = rnd(ACx * ACx)
`0101010E DDSS101%`	ACy = rnd(ACx)
`0101010E DDSS110%`	ACy = saturate(rnd(ACx))
`0101011E DDSSss0%`	ACy = rnd(ACy + (ACx * Tx))
`0101011E DDSSss1%`	ACy = rnd(ACy − (ACx * Tx))
`0101100E DDSSss0%`	ACy = rnd(ACx * Tx)
`0101100E DDSSss1%`	ACy = rnd((ACy * Tx) + ACx)
`0101101E DDSSss00`	ACy = ACy + (ACx << Tx)
`0101101E DDSSss01`	ACy = ACy − (ACx << Tx)
`0101101E DDxxxx1t`	ACx = sftc(ACx,TCx)
`0101110E DDSSss00`	ACy = ACx <<< Tx
`0101110E DDSSss01`	ACy = ACx << Tx
`0101110E DDSSss10`	ACy = ACx <<C Tx
`0101111E 00kkkkkk`	swap( )
`01100lll lCCCCCCC`	if (cond) goto l4
`01101000 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP`	if (cond) goto P24
`01101001 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP`	if (cond) call P24
`01101010 PPPPPPPP PPPPPPPP PPPPPPPP`	goto P24
`01101100 PPPPPPPP PPPPPPPP PPPPPPPP`	call P24
`01101101 xCCCCCCC LLLLLLLL LLLLLLLL`	if (cond) goto L16
`01101110 xCCCCCCC LLLLLLLL LLLLLLLL`	if (cond) call L16
`01101111 FSSSccxu KKKKKKKK LLLLLLLL`	compare (uns(src RELOP K8)) goto L8
`01110000 KKKKKKKK KKKKKKKK SSDDSHFT`	ACy = ACx + (K16 << #SHFT)
`01110001 KKKKKKKK KKKKKKKK SSDDSHFT`	ACy = ACx − (K16 << #SHFT)
`01110010 kkkkkkkk kkkkkkkk SSDDSHFT`	ACy = ACx & (k16 <<< #SHFT)
`01110011 kkkkkkkk kkkkkkkk SSDDSHFT`	ACy = ACx | (k16 <<< #SHFT)
`01110100 kkkkkkkk kkkkkkkk SSDDSHFT`	ACy = ACx ^ (k16 <<< #SHFT)

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
01110101 KKKKKKKK KKKKKKKK xxDDSHFT	ACx = K16 << #SHFT
01110110 kkkkkkkk kkkkkkkk FDDD00SS	dst = field_extract(ACx,k16)
01110110 kkkkkkkk kkkkkkkk FDDD01SS	dst = field_expand(ACx,k16)
01110110 KKKKKKKK KKKKKKKK FDDD10xx	dst = K16
01110111 DDDDDDDD DDDDDDDD FDDDxxxx	mar(TAx = D16)
01111000 kkkkkkkk kkkkkkkk xxx0000x	DP = k16
01111000 kkkkkkkk kkkkkkkk xxx0001x	SSP = k16
01111000 kkkkkkkk kkkkkkkk xxx0010x	CDP = k16
01111000 kkkkkkkk kkkkkkkk xxx0011x	BSA01 = k16
01111000 kkkkkkkk kkkkkkkk xxx0100x	BSA23 = k16
01111000 kkkkkkkk kkkkkkkk xxx0101x	BSA45 = k16
01111000 kkkkkkkk kkkkkkkk xxx0110x	BSA67 = k16
01111000 kkkkkkkk kkkkkkkk xxx0111x	BSAC = k16
01111000 kkkkkkkk kkkkkkkk xxx1000x	SP = k16
01111001 KKKKKKKK KKKKKKKK SSDDxx0%	ACy = rnd(ACx * K16)
01111001 KKKKKKKK KKKKKKKK SSDDss1%	ACy = rnd(ACx + (Tx * K16))
01111010 KKKKKKKK KKKKKKKK SSDD000x	ACy = ACx + (K16 << #16)
01111010 KKKKKKKK KKKKKKKK SSDD001x	ACy = ACx − (K16 << #16)
01111010 kkkkkkkk kkkkkkkk SSDD010x	ACy = ACx & (k16 <<< #16)
01111010 kkkkkkkk kkkkkkkk SSDD011x	ACy = ACx | (k16 <<< #16)
01111010 kkkkkkkk kkkkkkkk SSDD100x	ACy = ACx ^ (k16 <<< #16)
01111010 KKKKKKKK KKKKKKKK xxDD101x	ACx = K16 << #16
01111010 xxxxxxxx xxxxxxxx xxxx110x	idle
01111011 KKKKKKKK KKKKKKKK FDDDFSSS	dst = src + K16
01111100 KKKKKKKK KKKKKKKK FDDDFSSS	dst = src − K16
01111101 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src & k16
01111110 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src | k16
01111111 kkkkkkkk kkkkkkkk FDDDFSSS	dst = src ^ k16
10000000 XXXMMMYY YMMM00xx	dbl(Ymem) = dbl(Xmem)
10000000 XXXMMMYY YMMM01xx	Ymem = Xmem

10000000 XXXMMMYY YMMM10SS	Xmem = LO(ACx),
	Ymem = HI(ACx)

10000001 XXXMMMYY YMMM00DD	ACx = (Xmem << #16) + (Ymem << #16)

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`10000001 XXXMMMYY YMMM01DD`	ACx = (Xmem << #16) − (Ymem << #16)
`10000001 XXXMMMYY YMMM10DD`	LO(ACx) = Xmem, HI(ACx) = Ymem

`10000010 XXXMMMYY YMMM00mm uuDDDDg%`	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),
	ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))

`10000010 XXXMMMYY YMMM01mm uuDDDDg%`	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))

`10000010 XXXMMMYY YMMM10mm uuDDDDg%`	ACx = M40(rnd(ACx − (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))

`10000010 XXXMMMYY YMMM11mm uuxxDDg%`	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))

`10000011 XXXMMMYY YMMM00mm uuDDDDg%`	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))

`10000011 XXXMMMYY YMMM01mm uuDDDDg%`	ACx = M40(rnd(ACx − (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))

`10000011 XXXMMMYY YMMM10mm uuDDDDg%`	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))

`10000011 XXXMMMYY YMMM11mm uuxxDDg%`	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))

`10000100 XXXMMMYY YMMM00mm uuDDDDg%`	ACx = M40(rnd(ACx − (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))

`10000100 XXXMMMYY YMMM01mm uuxxDDg%`	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))

`10000100 XXXMMMYY YMMM10mm uuDDDDg%`	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),
	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))

`10000100 XXXMMMYY YMMM11mm uuDDDDg%`	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))

`10000101 XXXMMMYY YMMM00mm uuxxDDg%`	mar(Xmem), ACx = M40(rnd(ACx − (uns(Ymem) * uns(coef(Cmem)))))

`10000101 XXXMMMYY YMMM01mm uuDDDDg%`	ACx = M40(rnd(ACx − (uns(Xmem) * uns(coef(Cmem))))),
	ACy = M40(rnd(ACy − (uns(Ymem) * uns(coef(Cmem)))))

`10000101 XXXMMMYY YMMM10mm xxxxxxxx`	mar(Xmem) ,mar(Ymem) ,mar(coef(Cmem))

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`10000101 XXXMMMYY YMMM11mm DDx0DDU%`	firs(Xmem, Ymem, coef(Cmem), ACx, ACy)
`10000101 XXXMMMYY YMMM11mm DDx1DDU%`	firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)
`10000110 XXXMMMYY YMMMxxDD 000guuU%`	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]
`10000110 XXXMMMYY YMMMSSDD 001guuU%`	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]
`10000110 XXXMMMYY YMMMSSDD 010guuU%`	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]
`10000110 XXXMMMYY YMMMSSDD 011guuU%`	ACy = M40(rnd(ACx − (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]

`10000110 XXXMMMYY YMMMDDDD 100xssU%`	ACx = rnd(ACx − (Tx * Xmem)),
	ACy = Ymem << #16 [,T3 = Xmem]

`10000110 XXXMMMYY YMMMDDDD 101xssU%`	ACx = rnd(ACx + (Tx * Xmem)),
	ACy = Ymem << #16 [,T3 = Xmem]

`10000110 XXXMMMYY YMMMDDDD 110xxxx%`	lms(Xmem, Ymem, ACx, ACy)
`10000110 XXXMMMYY YMMMDDDD 1110xxn%`	sqdst(Xmem, Ymem, ACx, ACy)
`10000110 XXXMMMYY YMMMDDDD 1111xxn%`	abdst(Xmem, Ymem, ACx, ACy)

`10000111 XXXMMMYY YMMMSSDD 000xssU%`	ACy = rnd(Tx * Xmem),
	Ymem = HI(ACx << T2) [,T3 = Xmem]

`10000111 XXXMMMYY YMMMSSDD 001xssU%`	ACy = rnd(ACy + (Tx * Xmem)),
	Ymem = HI(ACx << T2) [,T3 = Xmem]

`10000111 XXXMMMYY YMMMSSDD 010xssU%`	ACy = rnd(ACy − (Tx * Xmem)),
	Ymem = HI(ACx << T2) [,T3 = Xmem]

`10000111 XXXMMMYY YMMMSSDD 01100001`	lmsf(Xmem, Ymem, ACx, ACy)

`10000111 XXXMMMYY YMMMSSDD 100xxxxx`	ACy = ACx + (Xmem << #16),
	Ymem = HI(ACy << T2)

`10000111 XXXMMMYY YMMMSSDD 101xxxxx`	ACy = (Xmem << #16) − ACx,
	Ymem = HI(ACy << T2)

`10000111 XXXMMMYY YMMMSSDD 110xxxxx`	ACy = Xmem << #16,
	Ymem = HI(ACx << T2)

`10010000 XSSSXDDD`	xdst = xsrc
`10010001 xxxxxxSS`	goto ACx

`10010010 XXXMMMYY YMMM00mm uuDDDDg%`	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))

`10010010 XXXMMMYY YMMM01mm uuDDDDg%`	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(ACx + uns(Xmem) *
	uns(LO(coef(Cmem)))))

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`10010010 XXXMMMYY YMMM10mm uuDDDDg%`	ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))
`10010010 xxxxxxSS`	call ACx
`10010011 XXXMMMYY YMMM00mm uuDDDDg%`	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))
`10010011 XXXMMMYY YMMM01mm uuDDDDg%`	ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))), ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))
`10010011 XXXMMMYY YMMM10mm uuDDDDg%`	ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))
`10010011 XXXMMMYY YMMM11mm uuDDDDg%`	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))
`10010100 XXXMMMYY YMMM00mm uuDDDDg%`	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))
`10010100 XXXMMMYY YMMM10mm uuDDDDg%`	ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))
`10010100 xxxxxxxx`	reset
`10010101 0xxkkkkk`	intr(k5)
`10010101 1xxkkkkk`	trap(k5)
`10010101 XXXMMMYY YMMM01mm uuDDDDg%`	ACy = M40(rnd(ACy − (uns(Ymem) * uns(HI(coef(Cmem)))))), ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))
`10010110 0CCCCCCC`	if (cond) execute(AD_unit)
`10010110 1CCCCCCC`	if (cond) execute(D_unit)
`10011000`	mmap()
`10011001`	readport()
`10011010`	writeport()

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
10011100	linear()
10011101	circular()
10011110 0CCCCCCC	if (cond) execute(AD_unit)
10011110 1CCCCCCC	if (cond) execute(D_unit)
10011111 0CCCCCCC	if (cond) execute(AD_unit)
10011111 1CCCCCCC	if (cond) execute(D_unit)
1010FDDD AAAAAAAI	dst = Smem
101100DD AAAAAAAI	ACx = Smem << #16
10110100 AAAAAAAI	mar(Smem)
10110101 AAAAAAAI	push(Smem)
10110110 AAAAAAAI	delay(Smem)
10110111 AAAAAAAI	push(dbl(Lmem))
10111000 AAAAAAAI	dbl(Lmem) = pop()
10111011 AAAAAAAI	Smem = pop()
101111SS AAAAAAAI	Smem = HI(ACx)
1100FSSS AAAAAAAI	Smem = src
11010000 AAAAAAAI 0%DD01mm	ACx = rnd(Smem * uns(coef(Cmem)))
11010000 AAAAAAAI 0%DD10mm	ACx = rnd(ACx + (Smem * uns(coef(Cmem))))
11010000 AAAAAAAI 0%DD11mm	ACx = rnd(ACx − (Smem * uns(coef(Cmem))))
11010000 AAAAAAAI U%DDxxmm	ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem], delay(Smem)
11010001 AAAAAAAI U%DD00mm	ACx = rnd(Smem * coef(Cmem)) [,T3 = Smem]
11010001 AAAAAAAI U%DD01mm	ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem]
11010001 AAAAAAAI U%DD10mm	ACx = rnd(ACx − (Smem * coef(Cmem))) [,T3 = Smem]
11010010 AAAAAAAI U%DD00SS	ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]
11010010 AAAAAAAI U%DD01SS	ACy = rnd(ACy − (Smem * ACx)) [,T3 = Smem]
11010010 AAAAAAAI U%DD10SS	ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]
11010010 AAAAAAAI U%DD11SS	ACy = rnd(ACx − (Smem * Smem)) [,T3 = Smem]
11010011 AAAAAAAI U%DD00SS	ACy = rnd(Smem * ACx) [,T3 = Smem]
11010011 AAAAAAAI U%DD10xx	ACx = rnd(Smem * Smem) [,T3 = Smem]
11010011 AAAAAAAI U%DDu1ss	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]
11010100 AAAAAAAI U%DDssSS	ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]
11010101 AAAAAAAI U%DDssSS	ACy = rnd(ACx − (Tx * Smem)) [,T3 = Smem]

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11010110 AAAAAAAI FDDDFSSS	dst = src + Smem
11010111 AAAAAAAI FDDDFSSS	dst = src − Smem
11011000 AAAAAAAI FDDDFSSS	dst = Smem − src
11011001 AAAAAAAI FDDDFSSS	dst = src & Smem
11011010 AAAAAAAI FDDDFSSS	dst = src | Smem
11011011 AAAAAAAI FDDDFSSS	dst = src ^ Smem
11011100 AAAAAAAI kkkkxx00	TC1 = bit(Smem, k4)
11011100 AAAAAAAI kkkkxx01	TC2 = bit(Smem, k4)
11011100 AAAAAAAI 0000xx10	DP = Smem
11011100 AAAAAAAI 0001xx10	CDP = Smem
11011100 AAAAAAAI 0010xx10	BSA01 = Smem
11011100 AAAAAAAI 0011xx10	BSA23 = Smem
11011100 AAAAAAAI 0100xx10	BSA45 = Smem
11011100 AAAAAAAI 0101xx10	BSA67 = Smem
11011100 AAAAAAAI 0110xx10	BSAC = Smem
11011100 AAAAAAAI 0111xx10	SP = Smem
11011100 AAAAAAAI 1000xx10	SSP = Smem
11011100 AAAAAAAI 1001xx10	BK03 = Smem
11011100 AAAAAAAI 1010xx10	BK47 = Smem
11011100 AAAAAAAI 1011xx10	BKC = Smem
11011100 AAAAAAAI 1100xx10	DPH = Smem
11011100 AAAAAAAI 1111xx10	PDP = Smem
11011100 AAAAAAAI x000xx11	CSR = Smem
11011100 AAAAAAAI x001xx11	BRC0 = Smem
11011100 AAAAAAAI x010xx11	BRC1 = Smem
11011100 AAAAAAAI x011xx11	TRN0 = Smem
11011100 AAAAAAAI x100xx11	TRN1 = Smem
11011101 AAAAAAAI SSDDss00	ACy = ACx + (Smem << Tx)
11011101 AAAAAAAI SSDDss01	ACy = ACx − (Smem << Tx)
11011101 AAAAAAAI SSDDss10	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)
11011101 AAAAAAAI x%DDss11	ACx = rnd(Smem << Tx)
11011110 AAAAAAAI SSDD0000	ACy = adsc(Smem, ACx, TC1)
11011110 AAAAAAAI SSDD0001	ACy = adsc(Smem, ACx, TC2)

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`11011110 AAAAAAAI SSDD0010`	ACy = adsc(Smem, ACx, TC1, TC2)
`11011110 AAAAAAAI SSDD0011`	subc(Smem, ACx, ACy)
`11011110 AAAAAAAI SSDD0100`	ACy = ACx + (Smem << #16)
`11011110 AAAAAAAI SSDD0101`	ACy = ACx − (Smem << #16)
`11011110 AAAAAAAI SSDD0110`	ACy = (Smem << #16) − ACx
`11011110 AAAAAAAI ssDD1000`	HI(ACx) = Smem + Tx, LO(ACx) = Smem − Tx
`11011110 AAAAAAAI ssDD1001`	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx
`11011111 AAAAAAAI FDDD000u`	dst = uns(high_byte(Smem))
`11011111 AAAAAAAI FDDD001u`	dst = uns(low_byte(Smem))
`11011111 AAAAAAAI xxDD010u`	ACx = uns(Smem)
`11011111 AAAAAAAI SSDD100u`	ACy = ACx + uns(Smem) + CARRY
`11011111 AAAAAAAI SSDD101u`	ACy = ACx − uns(Smem) − BORROW
`11011111 AAAAAAAI SSDD110u`	ACy = ACx + uns(Smem)
`11011111 AAAAAAAI SSDD111u`	ACy = ACx − uns(Smem)
`11100000 AAAAAAAI FSSSxxxt`	TCx = bit(Smem, src)
`11100001 AAAAAAAI DDSHIFTW`	ACx = low_byte(Smem) << #SHIFTW
`11100010 AAAAAAAI DDSHIFTW`	ACx = high_byte(Smem) << #SHIFTW
`11100011 AAAAAAAI kkkk000x`	TC1 = bit(Smem, k4), bit(Smem, k4) = #1
`11100011 AAAAAAAI kkkk001x`	TC2 = bit(Smem, k4), bit(Smem, k4) = #1
`11100011 AAAAAAAI kkkk010x`	TC1 = bit(Smem, k4), bit(Smem, k4) = #0
`11100011 AAAAAAAI kkkk011x`	TC2 = bit(Smem, k4), bit(Smem, k4) = #0
`11100011 AAAAAAAI kkkk100x`	TC1 = bit(Smem, k4), cbit(Smem, k4)
`11100011 AAAAAAAI kkkk101x`	TC2 = bit(Smem, k4), cbit(Smem, k4)
`11100011 AAAAAAAI FSSS1100`	bit(Smem, src) = #1
`11100011 AAAAAAAI FSSS1101`	bit(Smem, src) = #0
`11100011 AAAAAAAI FSSS111x`	cbit(Smem, src)
`11100100 AAAAAAAI FSSSx0xx`	push(src, Smem)
`11100100 AAAAAAAI FDDDx1xx`	dst, Smem = pop()
`11100101 AAAAAAAI FSSS01x0`	high_byte(Smem) = src
`11100101 AAAAAAAI FSSS01x1`	low_byte(Smem) = src
`11100101 AAAAAAAI 000010xx`	Smem = DP

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`11100101 AAAAAAAI 000110xx`	Smem = CDP
`11100101 AAAAAAAI 001010xx`	Smem = BSA01
`11100101 AAAAAAAI 001110xx`	Smem = BSA23
`11100101 AAAAAAAI 010010xx`	Smem = BSA45
`11100101 AAAAAAAI 010110xx`	Smem = BSA67
`11100101 AAAAAAAI 011010xx`	Smem = BSAC
`11100101 AAAAAAAI 011110xx`	Smem = SP
`11100101 AAAAAAAI 100010xx`	Smem = SSP
`11100101 AAAAAAAI 100110xx`	Smem = BK03
`11100101 AAAAAAAI 101010xx`	Smem = BK47
`11100101 AAAAAAAI 101110xx`	Smem = BKC
`11100101 AAAAAAAI 110010xx`	Smem = DPH
`11100101 AAAAAAAI 111110xx`	Smem = PDP
`11100101 AAAAAAAI x00011xx`	Smem = CSR
`11100101 AAAAAAAI x00111xx`	Smem = BRC0
`11100101 AAAAAAAI x01011xx`	Smem = BRC1
`11100101 AAAAAAAI x01111xx`	Smem = TRN0
`11100101 AAAAAAAI x10011xx`	Smem = TRN1
`11100110 AAAAAAAI KKKKKKKK`	Smem = K8
`11100111 AAAAAAAI SSss00xx`	Smem = LO(ACx << Tx)
`11100111 AAAAAAAI SSss10x%`	Smem = HI(rnd(ACx << Tx))
`11100111 AAAAAAAI SSss11u%`	Smem = HI(saturate(uns(rnd(ACx << Tx))))
`11101000 AAAAAAAI SSxxx0x%`	Smem = HI(rnd(ACx))
`11101000 AAAAAAAI SSxxx1u%`	Smem = HI(saturate(uns(rnd(ACx))))
`11101001 AAAAAAAI SSSHIFTW`	Smem = LO(ACx << #SHIFTW)
`11101010 AAAAAAAI SSSHIFTW`	Smem = HI(ACx << #SHIFTW)
`11101011 AAAAAAAI xxxx01xx`	dbl(Lmem) = RETA
`11101011 AAAAAAAI xxSS10x0`	dbl(Lmem) = ACx
`11101011 AAAAAAAI xxSS10u1`	dbl(Lmem) = saturate(uns(ACx))
`11101011 AAAAAAAI FSSS1100`	Lmem = pair(TAx)
`11101011 AAAAAAAI xxSS1101`	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1
`11101011 AAAAAAAI xxSS1110`	Lmem = pair(HI(ACx))

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
`11101011 AAAAAAAI xxSS1111`	Lmem = pair(LO(ACx))
`11101100 AAAAAAAI FSSS000x`	bit(src, Baddr) = #1
`11101100 AAAAAAAI FSSS001x`	bit(src, Baddr) = #0
`11101100 AAAAAAAI FSSS010x`	bit(src, pair(Baddr))
`11101100 AAAAAAAI FSSS011x`	cbit(src, Baddr)
`11101100 AAAAAAAI FSSS100t`	TCx = bit(src, Baddr)
`11101100 AAAAAAAI XDDD1110`	XAdst = mar(Smem)
`11101101 AAAAAAAI 00DD1010`	pair(HI(ACx)) = Lmem
`11101101 AAAAAAAI 00DD1100`	pair(LO(ACx)) = Lmem
`11101101 AAAAAAAI 00SS1110`	Lmem = pair(HI(ACx))
`11101101 AAAAAAAI 00SS1111`	Lmem = pair(LO(ACx))
`11101101 AAAAAAAI SSDD000n`	ACy = ACx + dbl(Lmem)
`11101101 AAAAAAAI SSDD001n`	ACy = ACx − dbl(Lmem)
`11101101 AAAAAAAI SSDD010x`	ACy = dbl(Lmem) − ACx
`11101101 AAAAAAAI xxxx011x`	RETA = dbl(Lmem)
`11101101 AAAAAAAI xxDD100g`	ACx = M40(dbl(Lmem))
`11101101 AAAAAAAI xxDD101x`	pair(HI(ACx)) = Lmem
`11101101 AAAAAAAI xxDD110x`	pair(LO(ACx)) = Lmem
`11101101 AAAAAAAI FDDD111x`	pair(TAx) = Lmem
`11101101 AAAAAAAI XDDD1111`	XAdst = dbl(Lmem)
`11101101 AAAAAAAI XSSS0101`	dbl(Lmem) = XAsrc
`11101110 AAAAAAAI SSDD000x`	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)
`11101110 AAAAAAAI SSDD001x`	HI(ACy) = HI(ACx) − HI(Lmem), LO(ACy) = LO(ACx) − LO(Lmem)
`11101110 AAAAAAAI SSDD010x`	HI(ACy) = HI(Lmem) − HI(ACx), LO(ACy) = LO(Lmem) − LO(ACx)
`11101110 AAAAAAAI ssDD011x`	HI(ACx) = Tx − HI(Lmem), LO(ACx) = Tx − LO(Lmem)
`11101110 AAAAAAAI ssDD100x`	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx
`11101110 AAAAAAAI ssDD101x`	HI(ACx) = HI(Lmem) − Tx, LO(ACx) = LO(Lmem) − Tx
`11101110 AAAAAAAI ssDD110x`	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) − Tx

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11101110 AAAAAAAI ssDD111x	HI(ACx) = HI(Lmem) − Tx, LO(ACx) = LO(Lmem) + Tx
11101111 AAAAAAAI xxxx00mm	Smem = coef(Cmem)
11101111 AAAAAAAI xxxx01mm	coef(Cmem) = Smem
11101111 AAAAAAAI xxxx10mm	Lmem = dbl(coef(Cmem))
11101111 AAAAAAAI xxxx11mm	dbl(coef(Cmem)) = Lmem
11110000 AAAAAAAI KKKKKKKK KKKKKKKK	TC1 = (Smem == K16)
11110001 AAAAAAAI KKKKKKKK KKKKKKKK	TC2 = (Smem == K16)
11110010 AAAAAAAI kkkkkkkk kkkkkkkk	TC1 = Smem & k16
11110011 AAAAAAAI kkkkkkkk kkkkkkkk	TC2 = Smem & k16
11110100 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem & k16
11110101 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem | k16
11110110 AAAAAAAI kkkkkkkk kkkkkkkk	Smem = Smem ^ k16
11110111 AAAAAAAI KKKKKKKK KKKKKKKK	Smem = Smem + K16
11111000 AAAAAAAI KKKKKKKK xxDDx0U%	ACx = rnd(Smem * K8) [,T3 = Smem]
11111000 AAAAAAAI KKKKKKKK SSDDx1U%	ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]
11111001 AAAAAAAI uxSHIFTW SSDD00xx	ACy = ACx + (uns(Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW SSDD01xx	ACy = ACx − (uns(Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW xxDD10xx	ACx = uns(Smem) << #SHIFTW
11111010 AAAAAAAI xxSHIFTW SSxxx0x%	Smem = HI(rnd(ACx << #SHIFTW))
11111010 AAAAAAAI uxSHIFTW SSxxx1x%	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
11111011 AAAAAAAI KKKKKKKK KKKKKKKK	Smem = K16
11111100 AAAAAAAI LLLLLLLL LLLLLLLL	if (ARn_mod != #0) goto L16

11111101 AAAAAAAI 000000mm DDDDuug%	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))

11111101 AAAAAAAI 000001mm DDDDuug%	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))

11111101 AAAAAAAI 000010mm DDDDuug%	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))

11111101 AAAAAAAI 000011mm DDDDuug%	ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax

`11111101 AAAAAAAI 000100mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))

`11111101 AAAAAAAI 000101mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 000110mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 000111mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 001000mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 001001mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 001010mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))

`11111101 AAAAAAAI 001011mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 001100mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010000mm DDDDuug%`	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))

`11111101 AAAAAAAI 010001mm DDDDuug%`	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),
	ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

Table 6−1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax

`11111101 AAAAAAAI 010010mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010011mm DDDDuug%`	ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010100mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010101mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010110mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 010111mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 011000mm DDDDuug%`	ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 011001mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 011010mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 011011mm DDDDuug%`	ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

`11111101 AAAAAAAI 011100mm DDDDuug%`	ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
	ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))

6.2 Instruction Set Opcode Symbols and Abbreviations

Table 6−2 lists the symbols and abbreviations used in the instruction set opcode.

Table 6−2. Instruction Set Opcode Symbols and Abbreviations

Bit Field Name	Bit Field Value	Bit Field Description
%	0	Rounding is disabled
	1	Rounding is enabled
AAAA AAAI		Smem addressing mode:
	AAAA AAA0	@dma, direct memory address (dma) direct access
	AAAA AAA1	Smem indirect memory access:
	0001 0001	ABS16(#k16)
	0011 0001	*(#k23)
	0101 0001	*port(#k16)
	0111 0001	*CDP
	1001 0001	*CDP+
	1011 0001	*CDP−
	1101 0001	*CDP(#K16)
	1111 0001	*+CDP(#K16)
	PPP0 0001	*ARn
	PPP0 0011	*ARn+
	PPP0 0101	*ARn−
	PPP0 0111	*(ARn + T0), when C54CM = 0 *(ARn + T0), when C54CM = 1
	PPP0 1001	*(ARn – T0), when C54CM = 0 *(ARn – T0), when C54CM = 1
	PPP0 1011	*ARn(T0), when C54CM = 0 *ARn(T0), when C54CM = 1
	PPP0 1101	*ARn(#K16)
	PPP0 1111	*+ARn(#K16)
	PPP1 0011	*(ARn + T1), when ARMS = 0 *ARn(short(#1)), when ARMS = 1
	PPP1 0101	*(ARn – T1), when ARMS = 0 *ARn(short(#2)), when ARMS = 1

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	PPP1 0111	*ARn(T1), when ARMS = 0 *ARn(short(#3)), when ARMS = 1
	PPP1 1001	*+ARn, when ARMS = 0 *ARn(short(#4)), when ARMS = 1
	PPP1 1011	*−ARn, when ARMS = 0 *ARn(short(#5)), when ARMS = 1
	PPP1 1101	*(ARn + T0B), when ARMS = 0 *ARn(short(#6)), when ARMS = 1
	PPP1 1111	*(ARn – T0B), when ARMS = 0 *ARn(short(#7)), when ARMS = 1
	PPP encodes an auxiliary register (ARn) as for XXX and YYY.	
cc		Relational operators (RELOP):
	00	== (equal to)
	01	< (less than)
	10	>= (greater than or equal to)
	11	!= (not equal to)
CCC CCCC		Conditional field (cond) on source accumulator, auxiliary, or temporary register; TCx; and CARRY:
	000 FSSS	src == 0 (source is equal to 0)
	001 FSSS	src != 0 (source is not equal to 0)
	010 FSSS	src < 0 (source is less than 0)
	011 FSSS	src <= 0 (source is less than or equal to 0)
	100 FSSS	src > 0 (source is greater than 0)
	101 FSSS	src >= 0 (source is greater than or equal to 0)
	110 00SS	overflow(ACx) (source accumulator overflow status bit (ACOVx) is tested against 1)
	110 0100	TC1 (status bit is tested against 1)
	110 0101	TC2 (status bit is tested against 1)
	110 0110	CARRY (status bit is tested against 1)
	110 0111	Reserved

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	110 1000	TC1 & TC2
	110 1001	TC1 & !TC2
	110 1010	!TC1 & TC2
	110 1011	!TC1 & !TC2
	110 11xx	Reserved
	111 00SS	!overflow(ACx) (source accumulator overflow status bit (ACOVx) is tested against 0)
	111 0100	!TC1 (status bit is tested against 0)
	111 0101	!TC2 (status bit is tested against 0)
	111 0110	!CARRY (status bit is tested against 0)
	111 0111	Reserved
	111 1000	TC1 | TC2
	111 1001	TC1 | !TC2
	111 1010	!TC1 | TC2
	111 1011	!TC1 | !TC2
	111 1100	TC1 ^ TC2
	111 1101	TC1 ^ !TC2
	111 1110	!TC1 ^ TC2
	111 1111	!TC1 ^ !TC2
`dd`		Destination temporary register (Tx, Ty):
	00	Temporary register 0 (T0)
	01	Temporary register 1 (T1)
	10	Temporary register 2 (T2)
	11	Temporary register 3 (T3)

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
DD		Destination accumulator register (ACw, ACx, ACy, ACz):
	00	Accumulator 0 (AC0)
	01	Accumulator 1 (AC1)
	10	Accumulator 2 (AC2)
	11	Accumulator 3 (AC3)
DDD . . . D		Data address label coded on n bits (absolute address)
E	0	Parallel Enable bit is cleared to 0
	1	Parallel Enable bit is set to 1
FDDD FSSS		Destination or Source accumulator, auxiliary, or temporary register (dst, src, TAx, TAy):
	0000	Accumulator 0 (AC0)
	0001	Accumulator 1 (AC1)
	0010	Accumulator 2 (AC2)
	0011	Accumulator 3 (AC3)
	0100	Temporary register 0 (T0)
	0101	Temporary register 1 (T1)
	0110	Temporary register 2 (T2)
	0111	Temporary register 3 (T3)
	1000	Auxiliary register 0 (AR0)
	1001	Auxiliary register 1 (AR1)
	1010	Auxiliary register 2 (AR2)
	1011	Auxiliary register 3 (AR3)
	1100	Auxiliary register 4 (AR4)
	1101	Auxiliary register 5 (AR5)
	1110	Auxiliary register 6 (AR6)
	1111	Auxiliary register 7 (AR7)

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
g	0	40 keyword is not applied
	1	40 keyword is applied; M40 is locally set to 1
kk kkkk		Swap code for Swap Register Content instruction:
	00 0000	swap(AC0, AC2)
	00 0001	swap(AC1, AC3)
	00 0100	swap(T0, T2)
	00 0101	swap(T1, T3)
	00 1000	swap(AR0, AR2)
	00 1001	swap(AR1, AR3)
	00 1100	swap(AR4, T0)
	00 1101	swap(AR5, T1)
	00 1110	swap(AR6, T2)
	00 1111	swap(AR7, T3)
	01 0000	swap(pair(AC0), pair(AC2))
	01 0001	Reserved
	01 0100	swap(pair(T0), pair(T2))
	01 0101	Reserved
	01 1000	swap(pair(AR0), pair(AR2))
	01 1001	Reserved
	01 1100	swap(pair(AR4), pair(T0))
	01 1101	Reserved
	01 1110	swap(pair(AR6), pair(T2))
	01 1111	Reserved
	10 1000	Reserved
	10 1100	swap(block(AR4), block(T0))
	11 1000	swap(AR0, AR1)
	11 1100	Reserved
	1x 0000	Reserved
	1x 0001	Reserved

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	1x 0100	Reserved
	1x 0101	Reserved
	1x 1001	Reserved
	1x 1101	Reserved
	1x 1110	Reserved
	1x 1111	Reserved
kkk . . . k		Unsigned constant of n bits
KKK . . . K		Signed constant of n bits
lll . . . l		Program address label coded on n bits (unsigned offset relative to program counter register)
LLL . . . L		Program address label coded on n bits (signed offset relative to program counter register)
mm		coef(Cmem)icient addressing mode (Cmem):
	00	*CDP
	01	*CDP+
	10	*CDP−
	11	*(CDP + T0)
MMM		Modifier option for Xmem or Ymem addressing mode:
	000	*ARn
	001	*ARn+
	010	*ARn−
	011	*(ARn + T0), when C54CM = 0 *(ARn + AR0), when C54CM = 1
	100	*(ARn + T1)

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	101	*(ARn – T0), when C54CM = 0 *(ARn – AR0), when C54CM = 1
	110	*(ARn – T1)
	111	*ARn(T0), when C54CM = 0 *ARn(AR0), when C54CM = 1
n		Reserved bit
PPP . . . P		Program or data address label coded on n bits (absolute address)
r	0	Select TRN0
	1	Select TRN1
SHFT		4-bit immediate shift value, 0 to 15
SHIFTW		6-bit immediate shift value, −32 to +31
ss		Source temporary register (Tx, Ty):
	00	Temporary register 0 (T0)
	01	Temporary register 1 (T1)
	10	Temporary register 2 (T2)
	11	Temporary register 3 (T3)
SS		Source accumulator register (ACw, ACx, ACy, ACz):
	00	Accumulator 0 (AC0)
	01	Accumulator 1 (AC1)
	10	Accumulator 2 (AC2)
	11	Accumulator 3 (AC3)

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
tt	00	Bit 0: destination TCy bit of Compare Register Content instruction
	01	Bit 1: source TCx bit of Compare Register Content instruction
	10	When value = 0: TC1 is selected
	11	When value = 1: TC2 is selected
u	0	uns keyword is not applied; operand is considered signed
	1	uns keyword is applied; operand is considered unsigned
U	0	No update of T3 with Smem or Xmem content
	1	T3 is updated with Smem or Xmem content
vv	00	Bit 0: shifted-out bit of Rotate instruction
	01	Bit 1: shifted-in bit of Rotate instruction
	10	When value = 0: CARRY is selected
	11	When value = 1: TC2 is selected
x		Reserved bit
XDDD XSSS		Destination or Source accumulator or extended register. All 23 bits of stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coef(Cmem)icient data pointer (XCDP), and extended auxiliary register (XARx).
	0000	Accumulator 0 (AC0)
	0001	Accumulator 1 (AC1)
	0010	Accumulator 2 (AC2)
	0011	Accumulator 3 (AC3)
	0100	Stack pointer (XSP)
	0101	System stack pointer (XSSP)
	0110	Data page pointer (XDP)
	0111	coef(Cmem)icient data pointer (XCDP)
	1000	Auxiliary register 0 (XAR0)

Table 6−2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	1001	Auxiliary register 1 (XAR1)
	1010	Auxiliary register 2 (XAR2)
	1011	Auxiliary register 3 (XAR3)
	1100	Auxiliary register 4 (XAR4)
	1101	Auxiliary register 5 (XAR5)
	1110	Auxiliary register 6 (XAR6)
	1111	Auxiliary register 7 (XAR7)
XXX YYY		Auxiliary register designation for Xmem or Ymem addressing mode:
	000	Auxiliary register 0 (AR0)
	001	Auxiliary register 1 (AR1)
	010	Auxiliary register 2 (AR2)
	011	Auxiliary register 3 (AR3)
	100	Auxiliary register 4 (AR4)
	101	Auxiliary register 5 (AR5)
	110	Auxiliary register 6 (AR6)
	111	Auxiliary register 7 (AR7)

Chapter 7

Cross-Reference of Algebraic and Mnemonic Instruction Sets

This chapter provides a cross-reference between the TMS320C55x™ DSP algebraic instruction set and the mnemonic instruction set (Table 7−1). For more information on the mnemonic instruction set, see *C55x CPU Mnemonic Instruction Set Reference Guide*, SWPU067.

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets

Algebraic Syntax	**Mnemonic Syntax**
Absolute Distance	**ABDST: Absolute Distance**
abdst(Xmem, Ymem, ACx, ACy)	ABDST Xmem, Ymem, ACx, ACy
Absolute Value	**ABS: Absolute Value**
dst = |src|	ABS [src,] dst
Addition	**ADD: Addition**
dst = dst + src	ADD [src,] dst
dst = dst + k4	ADD k4, dst
dst = src + K16	ADD K16, [src,] dst
dst = src + Smem	ADD Smem, [src,] dst
ACy = ACy + (ACx << Tx)	ADD ACx << Tx, ACy
ACy = ACy + (ACx << #SHIFTW)	ADD ACx << #SHIFTW, ACy
ACy = ACx + (K16 << #16)	ADD K16 << #16, [ACx,] ACy
ACy = ACx + (K16 << #SHFT)	ADD K16 << #SHFT, [ACx,] ACy
ACy = ACx + (Smem << Tx)	ADD Smem << Tx, [ACx,] ACy
ACy = ACx + (Smem << #16)	ADD Smem << #16, [ACx,] ACy
ACy = ACx + uns(Smem) + CARRY	ADD [uns(]Smem[)], CARRY, [ACx,] ACy
ACy = ACx + uns(Smem)	ADD [uns(]Smem[)], [ACx,] ACy
ACy = ACx + (uns(Smem) << #SHIFTW)	ADD [uns(]Smem[)] << #SHIFTW, [ACx,] ACy
ACy = ACx + dbl(Lmem)	ADD dbl(Lmem), [ACx,] ACy
ACx = (Xmem << #16) + (Ymem << #16)	ADD Xmem, Ymem, ACx
Smem = Smem + K16	ADD K16, Smem

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Addition with Absolute Value	**ADDV: Addition with Absolute Value**
ACy = rnd(ACy + |ACx|)	ADD[R]V [ACx,] ACy
Addition with Parallel Store Accumulator Content to Memory	**ADD::MOV: Addition with Parallel Store Accumulator Content to Memory**

ACy = ACx + (Xmem << #16),
Ymem = HI(ACy << T2)
	ADD Xmem << #16, ACx, ACy
	:: MOV HI(ACy << T2), Ymem

Addition or Subtraction Conditionally	**ADDSUBCC: Addition or Subtraction Conditionally**
ACy = adsc(Smem, ACx, TCx)	ADDSUBCC Smem, ACx, TCx, ACy
Addition or Subtraction Conditionally with Shift	**ADDSUB2CC: Addition or Subtraction Conditionally with Shift**
ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy
Addition, Subtraction, or Move Accumulator Content Conditionally	**ADDSUBCC: Addition, Subtraction, or Move Accumulator Content Conditionally**
ACy = adsc(Smem, ACx, TC1, TC2)	ADDSUBCC Smem, ACx, TC1, TC2, ACy
Bitwise AND	**AND: Bitwise AND**
dst = dst & src	AND src, dst
dst = src & k8	AND k8,src, dst
dst = src & k16	AND k16, src, dst
dst = src & Smem	AND Smem, src, dst
ACy = ACy & (ACx <<< #SHIFTW)	AND ACx << #SHIFTW[, ACy]
ACy = ACx & (k16 <<< #16)	AND k16 << #16, [ACx,] ACy
ACy = ACx & (k16 <<< #SHFT)	AND k16 << #SHFT, [ACx,] ACy

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Smem = Smem & k16	AND k16, Smem
Bitwise AND Memory with Immediate Value and Compare to Zero	**BAND: Bitwise AND Memory with Immediate Value and Compare to Zero**
TCx = Smem & k16	BAND Smem, k16, TCx
Bitwise OR	**OR: Bitwise OR**
dst = dst | src	OR src, dst
dst = src | k8	OR k8, src, dst
dst = src | k16	OR k16, src, dst
dst = src | Smem	OR Smem, src, dst
ACy = ACy | (ACx <<< #SHIFTW)	OR ACx << #SHIFTW[, ACy]
ACy = ACx | (k16 <<< #16)	OR k16 << #16, [ACx,] ACy
ACy = ACx | (k16 <<< #SHFT)	OR k16 << #SHFT, [ACx,] ACy
Smem = Smem | k16	OR k16, Smem
Bitwise Exclusive OR (XOR)	**XOR: Bitwise Exclusive OR (XOR)**
dst = dst ^ src	XOR src, dst
dst = src ^ k8	XOR k8, src, dst
dst = src ^ k16	XOR k16, src, dst
dst = src ^ Smem	XOR Smem, src, dst
ACy = ACy ^ (ACx <<< #SHIFTW)	XOR ACx << #SHIFTW[, ACy]
ACy = ACx ^ (k16 <<< #16)	XOR k16 << #16, [ACx,] ACy
ACy = ACx ^ (k16 <<< #SHFT)	XOR k16 << #SHFT, [ACx,] ACy

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Smem = Smem ^ k16	XOR k16, Smem
Branch Conditionally	**BCC: Branch Conditionally**
if (cond) goto l4	BCC l4, cond
if (cond) goto L8	BCC L8, cond
if (cond) goto L16	BCC L16, cond
if (cond) goto P24	BCC P24, cond
Branch Unconditionally	**B: Branch Unconditionally**
goto ACx	B ACx
goto L7	B L7
goto L16	B L16
goto P24	B P24
Branch on Auxiliary Register Not Zero	**BCC: Branch on Auxiliary Register Not Zero**
if (ARn_mod != #0) goto L16	BCC L16, ARn_mod != #0
Call Conditionally	**CALLCC: Call Conditionally**
if (cond) call L16	CALLCC L16, cond
if (cond) call P24	CALLCC P24, cond
Call Unconditionally	**CALL: Call Unconditionally**
call ACx	CALL ACx
call L16	CALL L16
call P24	CALL P24

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Circular Addressing Qualifier	**.CR: Circular Addressing Qualifier**
circular()	<instruction>.CR
Clear Accumulator, Auxiliary, or Temporary Register Bit	**BCLR: Clear Accumulator, Auxiliary, or Temporary Register Bit**
bit(src, Baddr) = #0	BCLR Baddr, src
Clear Memory Bit	**BCLR: Clear Memory Bit**
bit(Smem, src) = #0	BCLR src, Smem
Clear Status Register Bit	**BCLR: Clear Status Register Bit**
bit(STx, k4) = #0	BCLR k4, STx_55
	BCLR f−name
Compare Accumulator, Auxiliary, or Temporary Register Content	**CMP: Compare Accumulator, Auxiliary, or Temporary Register Content**
TCx = uns(src RELOP dst)	CMP[U] src RELOP dst, TCx
Compare Accumulator, Auxiliary, or Temporary Register Content with AND	**CMPAND: Compare Accumulator, Auxiliary, or Temporary Register Content with AND**
TCx = TCy & uns(src RELOP dst)	CMPAND[U] src RELOP dst, TCy, TCx
TCx = !TCy & uns(src RELOP dst)	CMPAND[U] src RELOP dst, !TCy, TCx
Compare Accumulator, Auxiliary, or Temporary Register Content with OR	**CMPOR: Compare Accumulator, Auxiliary, or Temporary Register Content with OR**
TCx = TCy | uns(src RELOP dst)	CMPOR[U] src RELOP dst, TCy, TCx
TCx = !TCy | uns(src RELOP dst)	CMPOR[U] src RELOP dst, !TCy, TCx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Compare Accumulator, Auxiliary, or Temporary Register Content Maximum	**MAX: Compare Accumulator, Auxiliary, or Temporary Register Content Maximum**
dst = max(src, dst)	MAX [src,] dst
Compare Accumulator, Auxiliary, or Temporary Register Content Minimum	**MIN: Compare Accumulator, Auxiliary, or Temporary Register Content Minimum**
dst = min(src, dst)	MIN [src,] dst
Compare and Branch	**BCC: Compare and Branch**
compare (uns(src RELOP K8)) goto L8	BCC[U] L8, src RELOP K8
Compare and Select Accumulator Content Maximum	**MAXDIFF: Compare and Select Accumulator Content Maximum**
max_diff(ACx, ACy, ACz, ACw)	MAXDIFF ACx, ACy, ACz, ACw
max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	DMAXDIFF ACx, ACy, ACz, ACw, TRNx
Compare and Select Accumulator Content Minimum	**MINDIFF: Compare and Select Accumulator Content Minimum**
min_diff(ACx, ACy, ACz, ACw)	MINDIFF ACx, ACy, ACz, ACw
min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	DMINDIFF ACx, ACy, ACz, ACw, TRNx
Compare Memory with Immediate Value	**CMP: Compare Memory with Immediate Value**
TCx = (Smem == K16)	CMP Smem == K16, TCx
Complement Accumulator, Auxiliary, or Temporary Register Bit	**BNOT: Complement Accumulator, Auxiliary, or Temporary Register Bit**
cbit(src, Baddr)	BNOT Baddr, src

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Complement Accumulator, Auxiliary, or Temporary Register Content	**NOT: Complement Accumulator, Auxiliary, or Temporary Register Content**
dst = ~src	NOT [src,] dst
Complement Memory Bit	**BNOT: Complement Memory Bit**
cbit(Smem, src)	BNOT src, Smem
Compute Exponent of Accumulator Content	**EXP: Compute Exponent of Accumulator Content**
Tx = exp(ACx)	EXP ACx, Tx
Compute Mantissa and Exponent of Accumulator Content	**MANT::NEXP: Compute Mantissa and Exponent of Accumulator Content**
ACy = mant(ACx), Tx = −exp(ACx)	MANT ACx, ACy :: NEXP ACx, Tx
Count Accumulator Bits	**BCNT: Count Accumulator Bits**
Tx = count(ACx, ACy, TCx)	BCNT ACx, ACy, TCx, Tx
Dual 16-Bit Additions	**ADD: Dual 16-Bit Additions**
HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	ADD dual(Lmem), [ACx,] ACy
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	ADD dual(Lmem), Tx, ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Dual 16-Bit Addition and Subtraction	**ADDSUB: Dual 16-Bit Addition and Subtraction**
HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	ADDSUB Tx, Smem, ACx
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	ADDSUB Tx, dual(Lmem), ACx
Dual 16-Bit Subtractions	**SUB: Dual 16-Bit Subtractions**
HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)	SUB dual(Lmem), [ACx,] ACy
HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)	SUB ACx, dual(Lmem), ACy
HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)	SUB dual(Lmem), Tx, ACx
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	SUB Tx, dual(Lmem), ACx
Dual 16-Bit Subtraction and Addition	**SUBADD: Dual 16-Bit Subtraction and Addition**
HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	SUBADD Tx, Smem, ACx
HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	SUBADD Tx, dual(Lmem), ACx
Execute Conditionally	**XCC: Execute Conditionally**
if (cond) execute(AD_Unit)	XCC [label,]cond
if (cond) execute(D_Unit)	XCCPART [label,]cond

Table 7−1. *Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)*

Algebraic Syntax	Mnemonic Syntax
Expand Accumulator Bit Field	**BFXPA: Expand Accumulator Bit Field**
dst = field_expand(ACx, k16)	BFXPA k16, ACx, dst
Extract Accumulator Bit Field	**BFXTR: Extract Accumulator Bit Field**
dst = field_extract(ACx, k16)	BFXTR k16, ACx, dst
Finite Impulse Response Filter, Antisymmetrical	**FIRSSUB: Finite Impulse Response Filter, Antisymmetrical**
firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy
Finite Impulse Response Filter, Symmetrical	**FIRSADD: Finite Impulse Response Filter, Symmetrical**
firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	FIRSADD Xmem, Ymem, Cmem, ACx, ACy
Idle	**IDLE**
idle	IDLE
Least Mean Square (LMS)	**LMS: Least Mean Square**
lms(Xmem, Ymem, ACx, ACy)	LMS Xmem, Ymem, ACx, ACy
lmsf(Xmem, Ymem, ACx, ACy)	LMSF Xmem, Ymem, ACx, ACy
Linear Addressing Qualifier	**.LR: Linear Addressing Qualifier**
linear()	<instruction>.LR

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	**Mnemonic Syntax**
Load Accumulator from Memory	**MOV: Load Accumulator from Memory**
ACx = rnd(Smem << Tx)	MOV [rnd(]Smem << Tx[)], ACx
ACx = low_byte(Smem) << #SHIFTW	MOV low_byte(Smem) << #SHIFTW, ACx
ACx = high_byte(Smem) << #SHIFTW	MOV high_byte(Smem) << #SHIFTW, ACx
ACx = Smem << #16	MOV Smem << #16, ACx
ACx = uns(Smem)	MOV [uns(]Smem[)], ACx
ACx = uns(Smem) << #SHIFTW	MOV [uns(]Smem[)] << #SHIFTW, ACx
ACx = M40(dbl(Lmem))	MOV[40] dbl(Lmem), ACx
LO(ACx) = Xmem, HI(ACx) = Ymem	MOV Xmem, Ymem, ACx
Load Accumulator from Memory with Parallel Store Accumulator Content to Memory	**MOV::MOV: Load Accumulator from Memory with Parallel Store Accumulator Content to Memory**

ACy = Xmem << #16,
Ymem = HI(ACx << T2)
	MOV Xmem << #16, ACy
	:: MOV HI(ACx << T2), Ymem

Load Accumulator Pair from Memory	**MOV: Load Accumulator Pair from Memory**
pair(HI(ACx)) = Lmem	MOV dbl(Lmem), pair(HI(ACx))
pair(LO(ACx)) = Lmem	MOV dbl(Lmem), pair(LO(ACx))
Load Accumulator with Immediate Value	**MOV: Load Accumulator with Immediate Value**
ACx = K16 << #16	MOV K16 << #16, ACx
ACx = K16 << #SHFT	MOV K16 << #SHFT, ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Load Accumulator, Auxiliary, or Temporary Register from Memory	**MOV: Load Accumulator, Auxiliary, or Temporary Register from Memory**
dst = Smem	MOV Smem, dst
dst = uns(high_byte(Smem))	MOV [uns(]high_byte(Smem)[)], dst
dst = uns(low_byte(Smem))	MOV [uns(]low_byte(Smem)[)], dst
Load Accumulator, Auxiliary, or Temporary Register with Immediate Value	**MOV: Load Accumulator, Auxiliary, or Temporary Register with Immediate Value**
dst = k4	MOV k4, dst
dst = –k4	MOV –k4, dst
dst = K16	MOV K16, dst
Load Auxiliary or Temporary Register Pair from Memory	**MOV: Load Auxiliary or Temporary Register Pair from Memory**
pair(TAx) = Lmem	MOV dbl(Lmem), pair(TAx)
Load CPU Register from Memory	**MOV: Load CPU Register from Memory**
BK03 = Smem	MOV Smem, BK03
BK47 = Smem	MOV Smem, BK47
BKC = Smem	MOV Smem, BKC
BSA01 = Smem	MOV Smem, BSA01
BSA23 = Smem	MOV Smem, BSA23
BSA45 = Smem	MOV Smem, BSA45
BSA67 = Smem	MOV Smem, BSA67
BSAC = Smem	MOV Smem, BSAC
BRC0 = Smem	MOV Smem, BRC0

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	**Mnemonic Syntax**
BRC1 = Smem	MOV Smem, BRC1
CDP = Smem	MOV Smem, CDP
CSR = Smem	MOV Smem, CSR
DP = Smem	MOV Smem, DP
DPH = Smem	MOV Smem, DPH
PDP = Smem	MOV Smem, PDP
SP = Smem	MOV Smem, SP
SSP = Smem	MOV Smem, SSP
TRN0 = Smem	MOV Smem, TRN0
TRN1 = Smem	MOV Smem, TRN1
RETA = dbl(Lmem)	MOV dbl(Lmem), RETA
Load CPU Register with Immediate Value	**MOV: Load CPU Register with Immediate Value**
BK03 = k12	MOV k12, BK03
BK47 = k12	MOV k12, BK47
BKC = k12	MOV k12, BKC
BRC0 = k12	MOV k12, BRC0
BRC1 = k12	MOV k12, BRC1
CSR = k12	MOV k12, CSR
DPH = k7	MOV k7, DPH
PDP = k9	MOV k9, PDP
BSA01 = k16	MOV k16, BSA01
BSA23 = k16	MOV k16, BSA23

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
BSA45 = k16	MOV k16, BSA45
BSA67 = k16	MOV k16, BSA67
BSAC = k16	MOV k16, BSAC
CDP = k16	MOV k16, CDP
DP = k16	MOV k16, DP
SP = k16	MOV k16, SP
SSP = k16	MOV k16, SSP
Load Extended Auxiliary Register from Memory	**MOV: Load Extended Auxiliary Register from Memory**
XAdst = dbl(Lmem)	MOV dbl(Lmem), XAdst
Load Extended Auxiliary Register with Immediate Value	**AMOV: Load Extended Auxiliary Register with Immediate Value**
XAdst = k23	AMOV k23, XAdst
Load Memory with Immediate Value	**MOV: Load Memory with Immediate Value**
Smem = K8	MOV K8, Smem
Smem = K16	MOV K16, Smem
Lock Access Qualifier	**.LK: Lock Access Qualifier**
lock()	.LK
Memory Delay	**DELAY: Memory Delay**
delay(Smem)	DELAY Smem

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Memory-Mapped Register Access Qualifier	**mmap: Memory-Mapped Register Access Qualifier**
mmap()	mmap
Modify Auxiliary Register Content	**AMAR: Modify Auxiliary Register Content**
mar(Smem)	AMAR Smem
Modify Auxiliary Register Content with Parallel Multiply	**AMAR::MPY: Modify Auxiliary Register Content with Parallel Multiply**
mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	AMAR Xmem :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx
Modify Auxiliary Register Content with Parallel Multiply and Accumulate	**AMAR::MAC: Modify Auxiliary Register Content with Parallel Multiply and Accumulate**
mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(coef(Cmem)))))	AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx
mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx >> #16
Modify Auxiliary Register Content with Parallel Multiply and Subtract	**AMAR::MAS: Modify Auxiliary Register Content with Parallel Multiply and Subtract**
mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(coef(Cmem)))))	AMAR Xmem :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx
Modify Auxiliary or Temporary Register Content	**AMOV: Modify Auxiliary or Temporary Register Content**
mar(TAy = TAx)	AMOV TAx, TAy
mar(TAx = P8)	AMOV P8, TAx
mar(TAx = D16)	AMOV D16, TAx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Modify Auxiliary or Temporary Register Content by Addition	**AADD: Modify Auxiliary or Temporary Register Content by Addition**
mar(TAy + TAx)	AADD TAx, TAy
mar(TAx + P8)	AADD P8, TAx
Modify Auxiliary or Temporary Register Content by Subtraction	**ASUB: Modify Auxiliary or Temporary Register Content by Subtraction**
mar(TAy – TAx)	ASUB TAx, TAy
mar(TAx – P8)	ASUB P8, TAx
Modify Data Stack Pointer	**AADD: Modify Data Stack Pointer (SP)**
SP = SP + K8	AADD K8, SP
Modify Extended Auxiliary Register Content	**AMAR: Modify Extended Auxiliary Register Content**
XAdst = mar(Smem)	AMAR Smem, XAdst
mar(XACdst = XACsrc) for DAG_X	AMOV XACsrc, XACdst for DAG_X
mar(XACdst = XACsrc) for DAG_Y	AMOV XACsrc, XACdst for DAG_Y
Modify Extended Auxiliary Register Content by Addition	**AADD: Modify Extended Auxiliary Register Content by Addition**
mar(XACdst + XACsrc) for DAG_X	AADD XACsrc, XACdst for DAG_X
mar(XACdst + XACsrc) for DAG_Y	AADD XACsrc, XACdst for DAG_Y

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Modify Extended Auxiliary Register Content by Subtraction	**ASUB: Modify Extended Auxiliary Register Content by Subtraction**
mar(XACdst − XACsrc) for DAG_X	ASUB XACsrc, XACdst for DAG_X
mar(XACdst − XACsrc) for DAG_Y	ASUB XACsrc, XACdst for DAG_Y
Move Accumulator Content to Auxiliary or Temporary Register	**MOV: Move Accumulator Content to Auxiliary or Temporary Register**
TAx = HI(ACx)	MOV HI(ACx), TAx
Move Accumulator, Auxiliary, or Temporary Register Content	**MOV: Move Accumulator, Auxiliary, or Temporary Register Content**
dst = src	MOV src, dst
Move Auxiliary or Temporary Register Content to Accumulator	**MOV: Move Auxiliary or Temporary Register Content to Accumulator**
HI(ACx) = TAx	MOV TAx, HI(ACx)
Move Auxiliary or Temporary Register Content to CPU Register	**MOV: Move Auxiliary or Temporary Register Content to CPU Register**
BRC0 = TAx	MOV TAx, BRC0
BRC1 = TAx	MOV TAx, BRC1
CDP = TAx	MOV TAx, CDP
CSR = TAx	MOV TAx, CSR
SP = TAx	MOV TAx, SP
SSP = TAx	MOV TAx, SSP

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Move CPU Register Content to Auxiliary or Temporary Register	**MOV: Move CPU Register Content to Auxiliary or Temporary Register**
TAx = BRC0	MOV BRC0, TAx
TAx = BRC1	MOV BRC1, TAx
TAx = CDP	MOV CDP, TAx
TAx = RPTC	MOV RPTC, TAx
TAx = SP	MOV SP, TAx
TAx = SSP	MOV SSP, TAx
Move Extended Auxiliary Register Content	**MOV: Move Extended Auxiliary Register Content**
xdst = xsrc	MOV xsrc, xdst
Move Memory to Memory	**MOV: Move Memory to Memory**
Smem = coef(Cmem)	MOV Cmem, Smem
coef(Cmem) = Smem	MOV Smem, Cmem
Lmem = dbl(coef(Cmem))	MOV Cmem, dbl(Lmem)
dbl(coef(Cmem)) = Lmem	MOV dbl(Lmem), Cmem
dbl(Ymem) = dbl(Xmem)	MOV dbl(Xmem), dbl(Ymem)
Ymem = Xmem	MOV Xmem, Ymem

Table 7−1. *Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)*

Algebraic Syntax	Mnemonic Syntax
Multiply	**MPY: Multiply**
ACy = rnd(ACy * ACx)	MPY[R] [ACx,] ACy
ACy = rnd(ACx * Tx)	MPY[R] Tx, [ACx,] ACy
ACy = rnd(ACx * K8)	MPYK[R] K8, [ACx,] ACy
ACy = rnd(ACx * K16)	MPYK[R] K16, [ACx,] ACy
ACx = rnd(Smem * uns(coef(Cmem)))	MPY[R] Smem, uns(Cmem), ACx
ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	MPYM[R] [T3 =]Smem, Cmem, ACx
ACy = rnd(Smem * ACx)[, T3 = Smem]	MPYM[R] [T3 =]Smem, [ACx,] ACy
ACx = rnd(Smem * K8)[, T3 = Smem]	MPYMK[R] [T3 =]Smem, K8, ACx
ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	MPYM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx
ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	MPYM[R][U] [T3 =]Smem, Tx, ACx
Multiply with Parallel Multiply and Accumulate	**MPY::MAC: Multiply with Parallel Multiply and Accumulate**

ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),
ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
	MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16

ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))
	MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))
	MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,
	:: MAC[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Multiply with Parallel Multiply and Subtract	**MPY::MAS: Multiply with Parallel Multiply and Subtract**

ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))
	MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))
	MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,
	:: MAS[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx

Multiply with Parallel Store Accumulator Content to Memory	**MPYM::MOV: Multiply with Parallel Store Accumulator Content to Memory**

ACy = rnd(Tx * Xmem),
Ymem = HI(ACx << T2) [,T3 = Xmem]
	MPYM[R] [T3 =]Xmem, Tx, ACy
	:: MOV HI(ACx << T2), Ymem

Multiply and Accumulate (MAC)	**MAC: Multiply and Accumulate**
ACy = rnd(ACy + (ACx * Tx))	MAC[R] ACx, Tx, ACy[, ACy]
ACy = rnd((ACy * Tx) + ACx)	MAC[R] ACy, Tx, ACx, ACy
ACx = rnd(ACx + (Smem * uns(coef(Cmem))))	MAC[R] Smem, uns(Cmem), ACx
ACy = rnd(ACx + (Tx * K8))	MACK[R] Tx, K8, [ACx,] ACy
ACy = rnd(ACx + (Tx * K16))	MACK[R] Tx, K16, [ACx,] ACy
ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	MACM[R] [T3 =]Smem, Cmem, ACx
ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	MACM[R] [T3 =]Smem, [ACx,] ACy
ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	MACM[R] [T3 =]Smem, Tx, [ACx,] ACy
ACy = rnd(ACx + (Smem * K8))[, T3 = Smem]	MACMK[R] [T3 =]Smem, K8, [ACx,] ACy
ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy
ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16 [, ACy]

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Multiply and Accumulate with Parallel Delay	**MACMZ: Multiply and Accumulate with Parallel Delay**
ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	MACM[R]Z [T3 =]Smem, Cmem, ACx
Multiply and Accumulate with Parallel Load Accumulator from Memory	**MACM::MOV: Multiply and Accumulate with Parallel Load Accumulator from Memory**

ACx = rnd(ACx + (Tx * Xmem)),
ACy = Ymem << #16 [,T3 = Xmem]
	MACM[R] [T3 =]Xmem, Tx, ACx
	:: MOV Ymem << #16, ACy

Multiply and Accumulate with Parallel Multiply	**MAC::MPY: Multiply and Accumulate with Parallel Multiply**

ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16,
	:: MPY[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Multiply and Accumulate with Parallel Multiply and Subtract	**MAC::MAS: Multiply and Accumulate with Parallel Multiply and Subtract**

ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx − uns(Xmem) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,
	:: MAS[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16,
	:: MAS[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx

Multiply and Accumulate with Parallel Store Accumulator Content to Memory	**MACM::MOV: Multiply and Accumulate with Parallel Store Accumulator Content to Memory**

ACy = rnd(ACy + (Tx * Xmem)),
Ymem = HI(ACx << T2) [,T3 = Xmem]
	MACM[R] [T3 =]Xmem, Tx, ACy
	:: MOV HI(ACx << T2), Ymem

Multiply and Subtract	**MAS: Multiply and Subtract**
ACy = rnd(ACy – (ACx * Tx))	MAS[R] Tx, [ACx,] ACy
ACx = rnd(ACx − (Smem * uns(coef(Cmem))))	MAS[R] Smem, uns(Cmem), ACx
ACx = rnd(ACx – (Smem * coef(Cmem)))[, T3 = Smem]	MASM[R] [T3 =]Smem, Cmem, ACx
ACy = rnd(ACy – (Smem * ACx))[, T3 = Smem]	MASM[R] [T3 =]Smem, [ACx,] ACy

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	**Mnemonic Syntax**
ACy = rnd(ACx – (Tx * Smem))[, T3 = Smem]	MASM[R] [T3 =]Smem, Tx, [ACx,] ACy
ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	MASM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy
Multiply and Subtract with Parallel Load Accumulator from Memory	**MASM::MOV: Multiply and Subtract with Parallel Load Accumulator from Memory**

ACx = rnd(ACx – (Tx * Xmem)),
ACy = Ymem << #16 [,T3 = Xmem]
	MASM[R] [T3 =]Xmem, Tx, ACx
	:: MOV Ymem << #16, ACy

Multiply and Subtract with Parallel Multiply	**MAS::MPY: Multiply and Subtract with Parallel Multiply**

ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))
	MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))
	MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

Multiply and Subtract with Parallel Multiply and Accumulate	**MAS::MAC: Multiply and Subtract with Parallel Multiply and Accumulate**

ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16

ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))
	MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Multiply and Subtract with Parallel Store Accumulator Content to Memory	**MASM::MOV: Multiply and Subtract with Parallel Store Accumulator Content to Memory**

ACy = rnd(ACy – (Tx * Xmem)),
Ymem = HI(ACx << T2) [,T3 = Xmem]
	MASM[R] [T3 =]Xmem, Tx, ACy
	:: MOV HI(ACx << T2), Ymem

Negate Accumulator, Auxiliary, or Temporary Register Content	**NEG: Negate Accumulator, Auxiliary, or Temporary Register Content**
dst = –src	NEG [src,] dst
No Operation	**NOP: No Operation**
nop	NOP
nop_16	NOP_16
Parallel Modify Auxiliary Register Contents	**AMAR: Parallel Modify Auxiliary Register Contents**
mar(Xmem), mar(Ymem), mar(coef(Cmem))	AMAR Xmem, Ymem, Cmem
Parallel Multiplies	**MPY::MPY: Parallel Multiplies**

ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))),
ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
	MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACy = M40(rnd(uns(Smem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(uns(Smem) * uns(LO(coef(Cmem)))))
	MPY[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(HI(Lmem)) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(uns(LO(Lmem)) * uns(LO(coef(Cmem)))))
	MPY[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MPY[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(uns(Ymem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(uns(Xmem) * uns(LO(coef(Cmem)))))
	MPY[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,
	:: MPY[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	**Mnemonic Syntax**
Parallel Multiply and Accumulates	**MAC::MAC: Parallel Multiply and Accumulates**

ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),
ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16
	:: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16

ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx + (uns(Smem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy + (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx>>#16

ACy = M40(rnd((ACy>>#16) + (uns(Smem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd((ACx>>#16) + (uns(Smem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MAC[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx>>#16

ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy + (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx>>#16

ACy = M40(rnd((ACy>>#16) + (uns(HI(Lmem)) * uns(HI(coef(Cmem))))))),
ACx = M40(rnd((ACx>>#16) + (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy>>#16
	:: MAC[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx>>#16

ACy = M40(rnd(ACy + uns(Ymem) * uns(HI(coef(Cmem))))),
ACx = M40(rnd(ACx + uns(Xmem) * uns(LO(coef(Cmem)))))
	MAC[R][40] [uns(]Ymem[)], [uns(]HI(Cmem)[)], ACy,
	:: MAC[R][40] [uns(]Xmem[)], [uns(]LO(Cmem)[)], ACx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax

ACy = M40(rnd(ACy + (uns(Ymem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy,
	:: MAC[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx >> #16

ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(LO(coef(Cmem))))))
	MAC[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy >> #16,
	:: MAC[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx >> #16

Parallel Multiply and Subtracts	**MAS::MAS: Parallel Multiply and Subtracts**

ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))),
ACy = M40(rnd(ACy – (uns(Ymem) * uns(coef(Cmem)))))
	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx
	:: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

ACy = M40(rnd(ACy − (uns(Smem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(Smem) * uns(LO(coef(Cmem))))))
	MAS[R][40] [uns(]Smem[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]Smem[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy − (uns(HI(Lmem)) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(LO(Lmem)) * uns(LO(coef(Cmem))))))
	MAS[R][40] [uns(]HI(Lmem)[)], [uns(]HI(Cmem)[)], ACy
	:: MAS[R][40] [uns(]LO(Lmem)[)], [uns(]LO(Cmem)[)], ACx

ACy = M40(rnd(ACy − (uns(Ymem) * uns(HI(coef(Cmem)))))),
ACx = M40(rnd(ACx − (uns(Xmem) * uns(LO(coef(Cmem))))))
	MAS[R][40] [uns(]HI(Ymem)[)], [uns(]HI(Cmem)[)], ACy,
	:: MAS[R][40] [uns(]LO(Xmem)[)], [uns(]LO(Cmem)[)], ACx

Peripheral Port Register Access Qualifiers	**port: Peripheral Port Register Access Qualifiers**
readport()	port(Smem)
writeport()	port(Smem)
Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers	**POPBOTH: Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers**
xdst = popboth()	POPBOTH xdst

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Pop Top of Stack	**POP: Pop Top of Stack**
dst1, dst2 = pop()	POP dst1, dst2
dst = pop()	POP dst
dst, Smem = pop()	POP dst, Smem
ACx = dbl(pop())	POP ACx
Smem = pop()	POP Smem
dbl(Lmem) = pop()	POP dbl(Lmem)
Push Accumulator or Extended Auxiliary Register Content to Stack Pointers	**PSHBOTH: Push Accumulator or Extended Auxiliary Register Content to Stack Pointers**
pshboth(xsrc)	PSHBOTH xsrc
Push to Top of Stack	**PSH: Push to Top of Stack**
push(src1, src2)	PSH src1, src2
push(src)	PSH src
push(src, Smem)	PSH src, Smem
dbl(push(ACx))	PSH ACx
push(Smem)	PSH Smem
push(dbl(Lmem))	PSH dbl(Lmem)
Repeat Block of Instructions Unconditionally	**RPTB: Repeat Block of Instructions Unconditionally**
localrepeat{ }	RPTBLOCAL pmad
blockrepeat{ }	RPTB pmad

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Repeat Single Instruction Conditionally	**RPTCC: Repeat Single Instruction Conditionally**
while (cond && (RPTC < k8)) repeat	RPTCC k8, cond
Repeat Single Instruction Unconditionally	**RPT: Repeat Single Instruction Unconditionally**
repeat(k8)	RPT k8
repeat(k16)	RPT k16
repeat(CSR)	RPT CSR
Repeat Single Instruction Unconditionally and Decrement CSR	**RPTSUB: Repeat Single Instruction Unconditionally and Decrement CSR**
repeat(CSR), CSR –= k4	RPTSUB CSR, k4
Repeat Single Instruction Unconditionally and Increment CSR	**RPTADD: Repeat Single Instruction Unconditionally and Increment CSR**
repeat(CSR), CSR += TAx	RPTADD CSR, TAx
repeat(CSR), CSR += k4	RPTADD CSR, k4
Return Conditionally	**RETCC: Return Conditionally**
if (cond) return	RETCC cond
Return Unconditionally	**RET: Return Unconditionally**
return	RET
Return from Interrupt	**RETI: Return from Interrupt**
return_int	RETI

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Rotate Left Accumulator, Auxiliary, or Temporary Register Content	**ROL: Rotate Left Accumulator, Auxiliary, or Temporary Register Content**
dst = BitOut \\ src \\ BitIn	ROL BitOut, src, BitIn, dst
Rotate Right Accumulator, Auxiliary, or Temporary Register Content	**ROR: Rotate Right Accumulator, Auxiliary, or Temporary Register Content**
dst = BitIn // src // BitOut	ROR BitIn, src, BitOut, dst
Round Accumulator Content	**ROUND: Round Accumulator Content**
ACy = rnd(ACx)	ROUND [ACx,] ACy
Saturate Accumulator Content	**SAT: Saturate Accumulator Content**
ACy = saturate(rnd(ACx))	SAT[R] [ACx,] ACy
Set Accumulator, Auxiliary, or Temporary Register Bit	**BSET: Set Accumulator, Auxiliary, or Temporary Register Bit**
bit(src, Baddr) = #1	BSET Baddr, src
Set Memory Bit	**BSET: Set Memory Bit**
bit(Smem, src) = #1	BSET src, Smem
Set Status Register Bit	**BSET: Set Status Register Bit**
bit(STx, k4) = #1	BSET k4, STx_55
	BSET f−name
Shift Accumulator Content Conditionally	**SFTCC: Shift Accumulator Content Conditionally**
ACx = sftc(ACx, TCx)	SFTCC ACx, TCx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Shift Accumulator Content Logically	**SFTL: Shift Accumulator Content Logically**
ACy = ACx <<< Tx	SFTL ACx, Tx[, ACy]
ACy = ACx <<< #SHIFTW	SFTL ACx, #SHIFTW[, ACy]
Shift Accumulator, Auxiliary, or Temporary Register Content Logically	**SFTL: Shift Accumulator, Auxiliary, or Temporary Register Content Logically**
dst = dst <<< #1	SFTL dst, #1
dst = dst >>> #1	SFTL dst, #−1
Signed Shift of Accumulator Content	**SFTS: Signed Shift of Accumulator Content**
ACy = ACx << Tx	SFTS ACx, Tx[, ACy]
ACy = ACx << #SHIFTW	SFTS ACx, #SHIFTW[, ACy]
ACy = ACx <<C Tx	SFTSC ACx, Tx[, ACy]
ACy = ACx <<C #SHIFTW	SFTSC ACx, #SHIFTW[, ACy]
Signed Shift of Accumulator, Auxiliary, or Temporary Register Content	**SFTS: Signed Shift of Accumulator, Auxiliary, or Temporary Register Content**
dst = dst >> #1	SFTS dst, #−1
dst = dst << #1	SFTS dst, #1
Software Interrupt	**INTR: Software Interrupt**
intr(k5)	INTR k5
Software Reset	**RESET: Software Reset**
reset	RESET

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Software Trap	**TRAP: Software Trap**
trap(k5)	TRAP k5
Square	**SQR: Square**
ACy = rnd(ACx * ACx)	SQR[R] [ACx,] ACy
ACx = rnd(Smem * Smem)[, T3 = Smem]	SQRM[R] [T3 =]Smem, ACx
Square and Accumulate	**SQA: Square and Accumulate**
ACy = rnd(ACy + (ACx * ACx))	SQA[R] [ACx,] ACy
ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]	SQAM[R] [T3 =]Smem, [ACx,] ACy
Square and Subtract	**SQS: Square and Subtract**
ACy = rnd(ACy – (ACx * ACx))	SQS[R] [ACx,] ACy
ACy = rnd(ACx – (Smem * Smem))[, T3 = Smem]	SQSM[R] [T3 =]Smem, [ACx,] ACy
Square Distance	**SQDST: Square Distance**
sqdst(Xmem, Ymem, ACx, ACy)	SQDST Xmem, Ymem, ACx, ACy
Store Accumulator Content to Memory	**MOV: Store Accumulator Content to Memory**
Smem = HI(ACx)	MOV HI(ACx), Smem
Smem = HI(rnd(ACx))	MOV [rnd(]HI(ACx)[)], Smem
Smem = LO(ACx << Tx)	MOV ACx << Tx, Smem
Smem = HI(rnd(ACx << Tx))	MOV [rnd(]HI(ACx << Tx)[)], Smem
Smem = LO(ACx << #SHIFTW)	MOV ACx << #SHIFTW, Smem
Smem = HI(ACx << #SHIFTW)	MOV HI(ACx << #SHIFTW), Smem

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Smem = HI(rnd(ACx << #SHIFTW))	MOV [rnd(]HI(ACx << #SHIFTW)[)], Smem
Smem = HI(saturate(uns(rnd(ACx))))	MOV [uns(] [rnd(]HI[(saturate](ACx)[)))], Smem
Smem = HI(saturate(uns(rnd(ACx << Tx))))	MOV [uns(] [rnd(]HI[(saturate](ACx << Tx)[)))], Smem
Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	MOV [uns(] [rnd(]HI[(saturate](ACx << #SHIFTW)[)))], Smem
dbl(Lmem) = ACx	MOV ACx, dbl(Lmem)
dbl(Lmem) = saturate(uns(ACx))	MOV [uns(]saturate(ACx)[)], dbl(Lmem)
HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	MOV ACx >> #1, dual(Lmem)

Xmem = LO(ACx),
Ymem = HI(ACx)
	MOV ACx, Xmem, Ymem

Store Accumulator Pair Content to Memory	**MOV: Store Accumulator Pair Content to Memory**
Lmem = pair(HI(ACx))	MOV pair(HI(ACx)), dbl(Lmem)
Lmem = pair(LO(ACx))	MOV pair(LO(ACx)), dbl(Lmem)
Store Accumulator, Auxiliary, or Temporary Register Content to Memory	**MOV: Store Accumulator, Auxiliary, or Temporary Register Content to Memory**
Smem = src	MOV src, Smem
high_byte(Smem) = src	MOV src, high_byte(Smem)
low_byte(Smem) = src	MOV src, low_byte(Smem)
Store Auxiliary or Temporary Register Pair Content to Memory	**MOV: Store Auxiliary or Temporary Register Pair Content to Memory**
Lmem = pair(TAx)	MOV pair(TAx), dbl(Lmem)

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	**Mnemonic Syntax**
Store CPU Register Content to Memory	**MOV: Store CPU Register Content to Memory**
Smem = BK03	MOV BK03, Smem
Smem = BK47	MOV BK47, Smem
Smem = BKC	MOV BKC, Smem
Smem = BSA01	MOV BSA01, Smem
Smem = BSA23	MOV BSA23, Smem
Smem = BSA45	MOV BSA45, Smem
Smem = BSA67	MOV BSA67, Smem
Smem = BSAC	MOV BSAC, Smem
Smem = BRC0	MOV BRC0, Smem
Smem = BRC1	MOV BRC1, Smem
Smem = CDP	MOV CDP, Smem
Smem = CSR	MOV CSR, Smem
Smem = DP	MOV DP, Smem
Smem = DPH	MOV DPH, Smem
Smem = PDP	MOV PDP, Smem
Smem = SP	MOV SP, Smem
Smem = SSP	MOV SSP, Smem
Smem = TRN0	MOV TRN0, Smem
Smem = TRN1	MOV TRN1, Smem
dbl(Lmem) = RETA	MOV RETA, dbl(Lmem)

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Store Extended Auxiliary Register Content to Memory	**MOV: Store Extended Auxiliary Register Content to Memory**
dbl(Lmem) = XAsrc	MOV XAsrc, dbl(Lmem)
Subtract Conditionally	**SUBC: Subtract Conditionally**
subc(Smem, ACx, ACy)	SUBC Smem, [ACx,] ACy
Subtraction	**SUB: Subtraction**
dst = dst – src	SUB [src,] dst
dst = dst – k4	SUB k4, dst
dst = src – K16	SUB K16, [src,] dst
dst = src – Smem	SUB Smem, [src,] dst
dst = Smem – src	SUB src, Smem, dst
ACy = ACy – (ACx << Tx)	SUB ACx << Tx, ACy
ACy = ACy – (ACx << #SHIFTW)	SUB ACx << #SHIFTW, ACy
ACy = ACx – (K16 << #16)	SUB K16 << #16, [ACx,] ACy
ACy = ACx – (K16 << #SHFT)	SUB K16 << #SHFT, [ACx,] ACy
ACy = ACx – (Smem << Tx)	SUB Smem << Tx, [ACx,] ACy
ACy = ACx – (Smem << #16)	SUB Smem << #16, [ACx,] ACy
ACy = (Smem << #16) – ACx	SUB ACx, Smem << #16, ACy
ACy = ACx – uns(Smem) – BORROW	SUB [uns(]Smem[)], BORROW, [ACx,] ACy
ACy = ACx – uns(Smem)	SUB [uns(]Smem[)], [ACx,] ACy
ACy = ACx – (uns(Smem) << #SHIFTW)	SUB [uns(]Smem[)] << #SHIFTW, [ACx,] ACy
ACy = ACx – dbl(Lmem)	SUB dbl(Lmem), [ACx,] ACy
ACy = dbl(Lmem) – ACx	SUB ACx, dbl(Lmem), ACy

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
ACx = (Xmem << #16) – (Ymem << #16)	SUB Xmem, Ymem, ACx
Subtraction with Parallel Store Accumulator Content to Memory	**SUB::MOV: Subtraction with Parallel Store Accumulator Content to Memory**

ACy = (Xmem << #16) – ACx,
Ymem = HI(ACy << T2)
	SUB Xmem << #16, ACx, ACy
	:: MOV HI(ACy << T2), Ymem

Swap Accumulator Content	**SWAP: Swap Accumulator Content**
swap(ACx, ACy)	SWAP ACx, ACy
Swap Accumulator Pair Content	**SWAPP: Swap Accumulator Pair Content**
swap(pair(AC0), pair(AC2))	SWAPP AC0, AC2
Swap Auxiliary Register Content	**SWAP: Swap Auxiliary Register Content**
swap(ARx, ARy)	SWAP ARx, ARy
Swap Auxiliary Register Pair Content	**SWAPP: Swap Auxiliary Register Pair Content**
swap(pair(AR0), pair(AR2))	SWAPP AR0, AR2
Swap Auxiliary and Temporary Register Content	**SWAP: Swap Auxiliary and Temporary Register Content**
swap(ARx, Tx)	SWAP ARx, Tx
Swap Auxiliary and Temporary Register Pair Content	**SWAPP: Swap Auxiliary and Temporary Register Pair Content**
swap(pair(ARx), pair(Tx))	SWAPP ARx, Tx
Swap Auxiliary and Temporary Register Pairs Content	**SWAP4: Swap Auxiliary and Temporary Register Pairs Content**
swap(block(AR4), block(T0))	SWAP4 AR4, T0

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Swap Temporary Register Content	**SWAP: Swap Temporary Register Content**
swap(Tx, Ty)	SWAP Tx, Ty
Swap Temporary Register Pair Content	**SWAPP: Swap Temporary Register Pair Content**
swap(pair(T0), pair(T2))	SWAPP T0, T2
Test Accumulator, Auxiliary, or Temporary Register Bit	**BTST: Test Accumulator, Auxiliary, or Temporary Register Bit**
TCx = bit(src, Baddr)	BTST Baddr, src, TCx
Test Accumulator, Auxiliary, or Temporary Register Bit Pair	**BTSTP: Test Accumulator, Auxiliary, or Temporary Register Bit Pair**
bit(src, pair(Baddr))	BTSTP Baddr, src
Test Memory Bit	**BTST: Test Memory Bit**
TCx = bit(Smem, src)	BTST src, Smem, TCx
TCx = bit(Smem, k4)	BTST k4, Smem, TCx
Test and Clear Memory Bit	**BTSTCLR: Test and Clear Memory Bit**
TCx = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TCx
Test and Complement Memory Bit	**BTSTNOT: Test and Complement Memory Bit**

TCx = bit(Smem, k4),
cbit(Smem, k4)
	BTSTNOT k4, Smem, TCx

Table 7−1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Test and Set Memory Bit	**BTSTSET: Test and Set Memory Bit**
TCx = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TCx

Index

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265