MPC General Purpose Compare and Branch Instructions

You will need a basic understanding of the general-purpose compare and branch instructions for In-lab 2. For most of your labs, you can use simplified mnemonic compare and branch instructions. In lab 2, we will give you a program to debug that uses simplified compare, simplified branch and general-purpose compare/branch instructions.

Directing Program Flow in Assembly

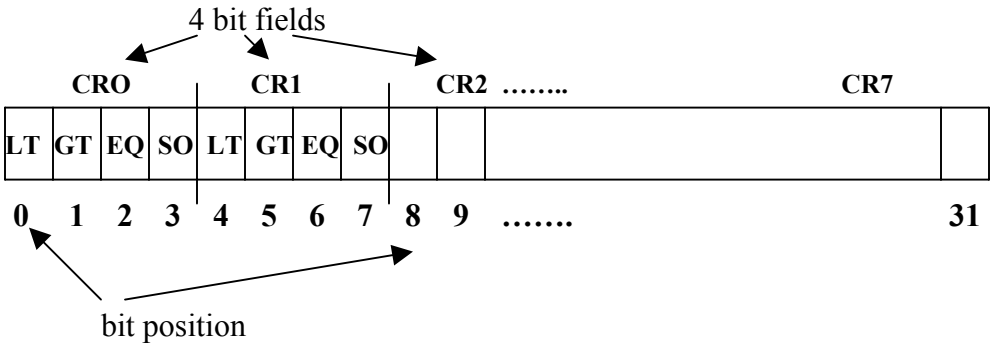
Directing program flow in assembly is a two-step process. First, a test is made usually by comparing two values. Second, program flow is directed from the results of the test.

The arguments of the compare may be signed or unsigned integers. If the arguments are signed integers, it is called an arithmetic compare. If the arguments are integers, the compare is called a logical compare. The results of the compare are stored in special register called a condition register. The results are usually stored as less then (LT), greater then (GT), equal (EQ).

The results of the test or compare are interpreted by a conditional branch instruction. A conditional branch control instruction reads the CR and directs program flow accordingly.

MPC Condition Register

The MPC CR is composed of 8, 4 bit fields. Each 4 bit field contains the results of a compare. The results are bit encoded as; less then (LT), greater then (GT), equal (EQ) and summation overflow (SO). The results are represented in the MPC CR as shown:



General Purpose Compare Instructions

There are four general-purpose compare instructions:

1. **cmp**, compare
2. **cmpi**, compare immediate
3. **cmpl**, compare logical
4. **cmpli**, compare logical immediate

The cmp and cmpi are arithmetic compares. The cmp compares two registers while the cmpi compares a register with a constant or immediate value. The cmpl and cmpli are similar except they perform a logical comparison. They all have similar syntax and operation. See the White book for more details. The cmp has the following syntax:

cmp crf, ra, rb

Where crf is one of the 4 bit condition register fields and ra and rb are general purpose registers. If:

ra < rb, the **LT** bit of the respective crf field is set (=1)
ra > rb, the **GT** bit of the respective crf field is set (=1)
ra = rb, the **EQ** bit of the respective crf field is set (=1)

The **crf** fields are coded **0 –7**.

Typically, the crf field is omitted when using simplified mnemonic compares and the result is placed in the CR0 field. For example,

cmplw ra, rb

Compares 2 unsigned integers in ra and rb and places the result in the cr0 field.

General Purpose Conditional Branch, bc

The general purpose conditional branch has the following syntax:

bc bo, bi, label

Where bo is encoded such that a **4** means the branch condition should be interpreted as **false** and **12** means the condition should be interpreted as **true**. The field bi specifies which **bit** (not field) in the condition register should be used as the branch condition. Label specifies the branch location if the condition is meet.

The bo field allows for the implementation of >= or <=.

For example,

bc, 12, 4, exit

will branch to exit if the results of the associated compare are ra < rb.

However, if the same condition is tested false or logical not of ra < rb, then the branch will occur if ra >= rb. Testing false condition requires the bo field be 4, thus the instruction reads:

bc 4, 4, exit

Simplified mnemonic branch instructions evaluate the CR0 field unless otherwise specified and are much easier to read and code then the bc. For example, from page F-13 of simplified mnemonics appendix:

Branch Semantics	L
	bc Relative
Branch if less than	blt
Branch if less than or equal	ble
Branch if equal	beq
Branch if greater than or equal	bge
Branch if greater than	bgt
Branch if not less than	bnl
Branch if not equal	bne
Branch if not greater than	bng