

MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$s1,\$s2,\$s3	\$s1 = \$s2 + \$s3	Three operands; overflow detected
	subtract	sub \$s1,\$s2,\$s3	\$s1 = \$s2 - \$s3	Three operands; overflow detected
	add immediate	addi \$s1,\$s2,100	\$s1 = \$s2 + 100	+ constant; overflow detected
	add unsigned	addu \$s1,\$s2,\$s3	\$s1 = \$s2 + \$s3	Three operands; overflow undetected
	subtract unsigned	subu \$s1,\$s2,\$s3	\$s1 = \$s2 - \$s3	Three operands; overflow undetected
	add immediate unsigned	addiu \$s1,\$s2,100	\$s1 = \$s2 + 100	+ constant; overflow undetected
	move from coprocessor register	mfc0 \$s1,\$epc	\$s1 = \$epc	Used to copy Exception PC plus other special registers
	multiply	mult \$s2,\$s3	Hi, Lo = \$s2 × \$s3	64-bit signed product in Hi, Lo
	multiply unsigned	multu \$s2,\$s3	Hi, Lo = \$s2 × \$s3	64-bit unsigned product in Hi, Lo
	divide	div \$s2,\$s3	Lo = \$s2 / \$s3, Hi = \$s2 mod \$s3	Lo = quotient, Hi = remainder
	divide unsigned	divu \$s2,\$s3	Lo = \$s2 / \$s3, Hi = \$s2 mod \$s3	Unsigned quotient and remainder
	move from Hi	mfhi \$s1	\$s1 = Hi	Used to get copy of Hi
move from Lo	mflo \$s1	\$s1 = Lo	Used to get copy of Lo	
Logical	and	and \$s1,\$s2,\$s3	\$s1 = \$s2 & \$s3	Three reg. operands; logical AND
	or	or \$s1,\$s2,\$s3	\$s1 = \$s2 \$s3	Three reg. operands; logical OR
	and immediate	andi \$s1,\$s2,100	\$s1 = \$s2 & 100	Logical AND reg, constant
	or immediate	ori \$s1,\$s2,100	\$s1 = \$s2 100	Logical OR reg, constant
	shift left logical	sll \$s1,\$s2,10	\$s1 = \$s2 << 10	Shift left by constant
	shift right logical	srl \$s1,\$s2,10	\$s1 = \$s2 >> 10	Shift right by constant
Data transfer	load word	lw \$s1,100(\$s2)	\$s1 = Memory[\$s2+100]	Word from memory to register
	store word	sw \$s1,100(\$s2)	Memory[\$s2 + 100] = \$s1	Word from register to memory
	load byte unsigned	lbu \$s1,100(\$s2)	\$s1 = Memory[\$s2 + 100]	Byte from memory to register
	store byte	sb \$s1,100(\$s2)	Memory[\$s2 + 100] = \$s1	Byte from register to memory
	load upper immediate	lui \$s1,100	\$s1 = 100 * 2 ¹⁶	Loads constant in upper 16 bits
Conditional branch	branch on equal	beq \$s1,\$s2,25	if (\$s1 == \$s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne \$s1,\$s2,25	if (\$s1 != \$s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt \$s1,\$s2,\$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1 = 0	Compare less than; two's complement
	set less than immediate	slti \$s1,\$s2,100	if (\$s2 < 100) \$s1 = 1; else \$s1=0	Compare < constant; two's complement
	set less than unsigned	sltu \$s1,\$s2,\$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1=0	Compare less than; natural numbers
	set less than immediate unsigned	sltiu \$s1,\$s2,100	if (\$s2 < 100) \$s1 = 1; else \$s1 = 0	Compare < constant; natural numbers
Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	jal 2500	\$ra = PC + 4; go to 10000	For procedure call