7.3 MIPSzy instruction summary

Table 7.3.1: MIPSzy Instruction summary.

Instruction	Format	Description	Example
lw	`lw $a, 0($b)`	Load word: Copies data from memory at address $b to register $a.	`lw $t3, 0($t6)`
sw	`sw $a, 0($b)`	Store word: Copies data from register $a to memory at address $b.	`sw $t1, 0($t3)`
lw (with offset)	`lw $a, C($b)`	Load word: Copies data from memory at address $b + C to register $a.	`lw $t3, 20($t6)`
sw (with offset)	`sw $a, C($b)`	Store word: Copies data from register $a to memory at address $b + C.	`sw $t1, -4($t3)`
addi	`addi $a, $b, C`	Add immediate: Adds register $b and the immediate	`addi $t3, $t2, 7`

		value C, and writes the sum into register $a.	
add	`add $a, $b, $c`	Add: Computes the sum of registers $b and $c, and writes the sum into register $a.	`add $t4, $t1, $t2`
sub	`sub $a, $b, $c`	Subtract: Subtracts $c from $b, and writes the difference into register $a.	`sub $t3, $t2, $t5`
mul	`mul $a, $b, $c`	Multiply: Multiplies register $b and $c, and writes the lower 32-bits of the product into register $a. mul is a pseudoinstruction implemented using mult and mflo.	`mul $t3, $t2, $t1`
mult	`mult $a, $b`	Multiply: Multiplies register $a and $b, writing the 64-bit result to special register $LO and $HI.	`mult $t3, $t5`
mflo	`mflo $a`	Move from LO	`mflo $t2`

		register: Copies value held in special register $LO to register $a.	
beq	`beq $a, $b, BLabel`	Branch on equal: Branches to the instruction at BLabel if the values held in $a and $b are equal. Otherwise, instruction immediately after beq is executed.	`beq $t3, $t2, SumEq5`
bne	`bne $a, $b, BLabel`	Branch on not equal: Branches to the instruction at BLabel if the values held in $a and $b are not equal. Otherwise, instruction immediately after bne is executed.	`bne $t4, $t5, GuessNeqCorrect`
slt	`slt $a, $b, $c`	Set on less than: Write 1 to register $a if value held in register $b is less than value held in	`slt $t1, $t5, $t6`

		register $c, and otherwise writes 0.	
j	`j JLabel`	Jump: Causes execution to continue with the instruction at JLabel.	`j CalcTip`
jal	`jal JLabel`	Jump and link: Stores the address of the next instruction to register $ra, but continues execution with the instruction at JLabel.	`jal CalcTip`
jr	`jr $a`	Jump register: Causes execution to continue with the instruction at address $a.	`jr $t3`

Table 7.3.2: MIPSzy machine instructions.

Assembly	Machine
lw $t0, 0($t1)	100011 01001 01000 0000000000000000

sw $t0, 0($t1)	101011 01001 01000 0000000000000000
addi $t0, $t1, 15	001000 01001 01000 0000000000001111
add $t0, $t1, $t2	000000 01001 01010 01000 00000 100000
sub $t0, $t1, $t2	000000 01001 01010 01000 00000 100010
mult $t1, $t2	000100 01001 01010 00000 00000 011000
mflo $t0	000000 00000 00000 01000 00000 010010
beq $t1, $t2, BLabel	000100 01001 01010 0000000000000010
bne $t1, $t2, BLabel	000101 01001 01010 0000000000000010
slt $t0, $t1, $t2	000000 01001 01010 01000 00000 101010
j JLabel	000010 00000000000000000000000101
jal JLabel	000011 00000000000000100000000101
jr $t1	000000 01001 00000 00000 00000 001000

Assume BLabel becomes an immediate of 2, and JLabel 5. Creating immediates for branches/jumps is in another section.

$t0, $t1, and $t2 are used for registers. Other registers could be used.

addi's immediate value is shown as 15. That value is arbitrary.

Exploring further:

- The MIPS32 Instruction Set v6.05 (from Imagination Technologies)

Provide feedback on this section