

## Common MIPS instructions.

Notes: *op, funct, rd, rs, rt, imm, address, shamt* refer to fields in the instruction format. The program counter PC is assumed to point to the next instruction (usually 4 + the address of the current instruction). M is the byte-addressed main memory.

Assembly instruction	Instr. format	<i>op</i> <i>op/funct</i>	Meaning	Comments
add <i>\$rd, \$rs, \$rt</i>	R	0/32	$\$rd = \$rs + \$rt$	Add contents of two registers
sub <i>\$rd, \$rs, \$rt</i>	R	0/34	$\$rd = \$rs - \$rt$	Subtract contents of two registers
addi <i>\$rt, \$rs, imm</i>	I	8	$\$rt = \$rs + imm$	Add signed constant
addu <i>\$rd, \$rs, \$rt</i>	R	0/33	$\$rd = \$rs + \$rt$	Unsigned, no overflow
subu <i>\$rd, \$rs, \$rt</i>	R	0/35	$\$rd = \$rs - \$rt$	Unsigned, no overflow
addiu <i>\$rt, \$rs, imm</i>	I	9	$\$rt = \$rs + imm$	Unsigned, no overflow
mfc0 <i>\$rt, \$rd</i>	R	16	$\$rt = \$rd$	<i>rd</i> = coprocessor register (e.g. epc, cause, status)
mult <i>\$rs, \$rt</i>	R	0/24	Hi, Lo = $\$rs * \$rt$	64 bit signed product in Hi and Lo
multu <i>\$rs, \$rt</i>	R	0/25	Hi, Lo = $\$rs * \$rt$	64 bit unsigned product in Hi and Lo
div <i>\$rs, \$rt</i>	R	0/26	Lo = $\$rs / \$rt$ , Hi = $\$rs \bmod \$rt$	
divu <i>\$rs, \$rt</i>	R	0/27	Lo = $\$rs / \$rt$ , Hi = $\$rs \bmod \$rt$ (unsigned)	
mfhi <i>\$rd</i>	R	0/16	$\$rd = \text{Hi}$	Get value of Hi
mflo <i>\$rd</i>	R	0/18	$\$rd = \text{Lo}$	Get value of Lo
and <i>\$rd, \$rs, \$rt</i>	R	0/36	$\$rd = \$rs \& \$rt$	Logical AND
or <i>\$rd, \$rs, \$rt</i>	R	0/37	$\$rd = \$rs   \$rt$	Logical OR
andi <i>\$rt, \$rs, imm</i>	I	12	$\$rt = \$rs \& imm$	Logical AND, unsigned constant
ori <i>\$rt, \$rs, imm</i>	I	13	$\$rt = \$rs   imm$	Logical OR, unsigned constant
sll <i>\$rd, \$rt, shamt</i>	R	0/0	$\$rd = \$rt \ll shamt$	Shift left logical (shift in zeros)
srl <i>\$rd, \$rt, shamt</i>	R	0/2	$\$rd = \$rt \gg shamt$	Shift right logical (shift in zeros)
lw <i>\$rt, imm(\$rs)</i>	I	35	$\$rt = M[\$rs + imm]$	Load word from memory
sw <i>\$rt, imm(\$rs)</i>	I	43	$M[\$rs + imm] = \$rt$	Store word in memory
lbu <i>\$rt, imm(\$rs)</i>	I	37	$\$rt = M[\$rs + imm]$	Load a single byte, set bits 8-31 of <i>\$rt</i> to zero
sb <i>\$rt, imm(\$rs)</i>	I	41	$M[\$rs + imm] = \$rt$	Store byte (bits 0-7 of <i>\$rt</i> ) in memory
lui <i>\$rt, imm</i>	I	15	$\$rt = imm * 2^{16}$	Load constant in bits 16-31 of register <i>\$rt</i>
beq <i>\$rs, \$rt, imm</i>	I	4	if( $\$rs == \$rt$ ) PC = PC + <i>imm</i> (PC always points to next instruction)	
bne <i>\$rs, \$rt, imm</i>	I	5	if( $\$rs \neq \$rt$ ) PC = PC + <i>imm</i> (PC always points to next instruction)	
slt <i>\$rd, \$rs, \$rt</i>	R	0/42	if( $\$rs < \$rt$ ) $\$rd = 1$ ; else $\$rd = 0$	
slti <i>\$rt, \$rs, imm</i>	I	10	if( $\$rs < imm$ ) $\$rt = 1$ ; else $\$rt = 0$	
sltu <i>\$rd, \$rs, \$rt</i>	R	0/43	if( $\$rs < \$rt$ ) $\$rd = 1$ ; else $\$rd = 0$ (unsigned numbers)	
sltiu <i>\$rt, \$rs, imm</i>	I	11	if( $\$rs < imm$ ) $\$rt = 1$ ; else $\$rt = 0$ (unsigned numbers)	
j <i>destination</i>	J	2	PC = <i>address</i> *4	Jump to <i>destination</i> , <i>address</i> = <i>destination</i> /4
jal <i>destination</i>	J	3	$\$ra = \text{PC}$ ; PC = <i>address</i> *4 (Jump and link, <i>address</i> = <i>destination</i> /4)	
jr <i>\$rs</i>	R	0/8	PC = <i>\$rs</i>	Jump to address stored in register <i>\$rs</i>

## MIPS registers

Name	Number	Usage
\$zero	0	constant 0
\$at	1	reserved for assembler
\$v0 - \$v1	2-3	expression evaluation and function results
\$a0 - \$a3	4-7	arguments
\$t0 - \$t7	8-15	temporary, saved by caller
\$s0 - \$s7	16-23	temporary, saved by called function
\$t8 - \$t9	24-25	temporary, saved by caller
\$k0 - \$k1	26-27	reserved for kernel (OS)
\$gp	28	points to middle of a 64K block in the data segment
\$sp	29	stack pointer (top of stack)
\$fp	30	frame pointer (beginning of current frame)
\$ra	31	return address
Hi, Lo	-	store partial result of mult and div operations
PC	-	contains the address of the next instruction to be fetched (this is not a real MIPS register, and is only used to define instructions)
status	-	register 12 in coprocessor 0, stores interrupt mask and enable bits
cause	-	register 13 in coprocessor 0, stores exception type and pending interrupt bits
epc	-	register 14 in coprocessor 0, stores address of instruction causing exception

## MIPS Instruction formats

Format	Bits 31-26	Bits 25-21	Bits 20-16	Bits 15-11	Bits 10-6	Bits 5-0
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	imm		
J	op	address				

## MIPS Assembler syntax

```

                                # This is a comment
                                # Store following data in the data
                                # segment
items:                          # This is a label connected to the
                                # next address in the current segment
                                # Stores values 1 and 2 in next two
                                # words
hello:   .asciiz "Hello"       # Stores null-terminated string in
                                # memory
                                # Store following instructions in
                                # the text segment
main:    lw $t0, items($zero)  # Instruction that uses a label to
                                # address data

A:       .word 0:10            # Stores 10 words initialized to 0

```