

Name	Register number	Usage	Preserved on call?
\$zero	0	the constant value 0	n.a.
\$v0-\$v1	2-3	values for results and expression evaluation	no
\$a0-\$a3	4-7	arguments	yes
\$t0-\$t7	8-15	temporaries	no
\$s0-\$s7	16-23	saved	yes
\$t8-\$t9	24-25	more temporaries	no
\$gp	28	global pointer	yes
\$sp	29	stack pointer	yes
\$fp	30	frame pointer	yes
\$ra	31	return address	yes

**FIGURE 3.13 MIPS register convention.** Register 1, called \$at, is reserved for the assembler (see section 3.9), and registers 26-27, called \$k0-\$k1, are reserved for the operating system.

### MIPS machine language

Name	Format	Example						Comments
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
add	R	0	2	3	1	0	32	add \$1,\$2,\$3
sub	R	0	2	3	1	0	34	sub \$1,\$2,\$3
addi	I	8	2	1	100			addi \$1,\$2,100
addu	R	0	2	3	1	0	33	addu \$1,\$2,\$3
subu	R	0	2	3	1	0	35	subu \$1,\$2,\$3
addiu	I	9	2	1	100			addiu \$1,\$2,100
mfc0	R	16	0	1	14	0	0	mfc0 \$1,\$epc
mult	R	0	2	3	0	0	24	mult \$2,\$3
multu	R	0	2	3	0	0	25	multu \$2,\$3
div	R	0	2	3	0	0	26	div \$2,\$3
divu	R	0	2	3	0	0	27	divu \$2,\$3
mfhi	R	0	0	0	1	0	16	mfhi \$1
mflo	R	0	0	0	1	0	18	mflo \$1
and	R	0	2	3	1	0	36	and \$1,\$2,\$3
or	R	0	2	3	1	0	37	or \$1,\$2,\$3
andi	I	12	2	1	100			andi \$1,\$2,100
ori	I	13	2	1	100			ori \$1,\$2,100
sll	R	0	0	2	1	10	0	sll \$1,\$2,10
srl	R	0	0	2	1	10	2	srl \$1,\$2,10
lw	I	35	2	1	100			lw \$1,100(\$2)
sw	I	43	2	1	100			sw \$1,100(\$2)
lui	I	15	0	1	100			lui \$1,100
beq	I	4	1	2	25			beq \$1,\$2,100
bne	I	5	1	2	25			bne \$1,\$2,100
slt	R	0	2	3	1	0	42	slt \$1,\$2,\$3
slti	I	10	2	1	100			slti \$1,\$2,100
sltu	R	0	2	3	1	0	43	sltu \$1,\$2,\$3
sltiu	I	11	2	1	100			sltiu \$1,\$2,100
j	J	2	2500					j 10000
jr	R	0	31	0	0	0	8	jr \$31
jal	J	3	2500					jal 10000

### MIPS instruction formats

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Main MIPS machine language. Formats and examples are shown, with values in each field: op and funct fields form the opcode (each 6 bits), rs field gives a source register (5 bits), rt is also normally a source register (5 bits), rd is the destination register (5 bits), and shamt supplies the shift amount (5 bits). The field values are all in decimal.

Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

**FIGURE 6.28** The values of the control lines are the same as in Figure 5.20 on page 361, but they have been shuffled into three groups corresponding to the last three pipeline stages.

### 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	$s1 = s2 + s3$	+ 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	$s1 = s2 + s3$	+ 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 = $s2 \times s3$	64-bit signed product in Hi, Lo
	multiply unsigned	multu \$s2,\$s3	Hi, Lo = $s2 \times s3$	64-bit unsigned product in Hi, Lo
	divide	div \$s2,\$s3	Lo = $s2 / s3$ , Hi = $s2 \bmod s3$	Lo = quotient, Hi = remainder
	divide unsigned	divu \$s2,\$s3	Lo = $s2 / s3$ , Hi = $s2 \bmod s3$	Unsigned quotient and remainder
	move from Hi	mfhi \$s1	$s1 = \text{Hi}$	Used to get copy of Hi
	move from Lo	mflo \$s1	$s1 = \text{Lo}$	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	$s1 = s2 \ll 10$	Shift left by constant
Data transfer	shift right logical	srl \$s1,\$s2,10	$s1 = s2 \gg 10$	Shift right by constant
	load word	lw \$s1,100(\$s2)	$s1 = \text{Memory}[s2+100]$	Word from memory to register
	store word	sw \$s1,100(\$s2)	$\text{Memory}[s2+100] = s1$	Word from register to memory
	load byte unsigned	lbu \$s1,100(\$s2)	$s1 = \text{Memory}[s2+100]$	Byte from memory to register
	store byte	sb \$s1,100(\$s2)	$\text{Memory}[s2+100] = s1$	Byte from register to memory
Conditional branch	load upper immediate	lui \$s1,100	$s1 = 100 * 2^{16}$	Loads constant in upper 16 bits
	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	$\$ra = \text{PC} + 4$ ; go to 10000	For procedure call

Main MIPS assembly language instruction set. The floating-point instructions are shown in Figure 4.47 on page 291. Appendix A gives the full MIPS assembly language instruction set.