

# Register Use Policy Conventions

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

<b><u>I/O Call</u></b>	<b><u>Code</u></b>	<b><u>Arguments or Result</u></b>
print int	\$v0=1	\$a0 = integer
print float	\$v0=2	\$f12 = oat
print double	\$v0=3	\$f12 = double
print string	\$v0=4	\$a0 = string
read int	\$v0=5	Integer (in \$v0 )
read foat	\$v0=6	float (in \$f0 )
read double	\$v0=7	double (in \$f0 )
read string	\$v0=8	\$a0 = buffer, \$a1 = length
sbrk	\$v0=9	\$a0 = amount,address in \$v0
exit	\$v0= 0	

# Basic MIPS Instructions

## MIPS operands

Name	Example	Comments
32 registers	$\$s0-\$s7$ , $\$t0-\$t9$ , $\$zero$ , $\$a0-\$a3$ , $\$v0-\$v1$ , $\$gp$ , $\$fp$ , $\$sp$ , $\$ra$ , $\$at$	Fast locations for data. In MIPS, data must be in registers to perform arithmetic. MIPS register $\$zero$ always equals 0. Register $\$at$ is reserved for the assembler to handle large constants.
$2^{30}$ memory words	Memory[0], Memory[4], ..., Memory[4294967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers, such as those saved on procedure calls.

## MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $\$s1$ , $\$s2$ , $\$s3$	$\$s1 = \$s2 + \$s3$	Three operands; data in registers
	subtract	sub $\$s1$ , $\$s2$ , $\$s3$	$\$s1 = \$s2 - \$s3$	Three operands; data in registers
	add immediate	addi $\$s1$ , $\$s2$ , 100	$\$s1 = \$s2 + 100$	Used to add constants
Data transfer	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	lb $\$s1$ , 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
	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; for beq, bne
	set less than immediate	slti $\$s1$ , $\$s2$ , 100	if ( $\$s2 < 100$ ) $\$s1 = 1$ ; else $\$s1 = 0$	Compare less than constant
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

### 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,\$cpc
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	100			beq \$1,\$2,100
bne	I	5	1	2	100			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
situ	R	0	2	3	1	0	43	situ \$1,\$2,\$3
sltiu	I	11	2	1	100			sltiu \$1,\$2,100
j	J	2	10000					j 10000
jr	R	0	31	0	0	0	8	jr \$1
jal	J	3	10000					jal 10000

### MIPS machine language

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

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. Floating-point machine language instructions are shown in Figure 4.44 on page 241. Appendix A gives the full MIPS machine language.

### MIPS operands

Name	Example	Comments
32 registers	\$0, \$1, \$2, . . . \$31, Hi, Lo	Fast locations for data. In MIPS, data must be in registers to perform arithmetic. MIPS register \$0 always equals 0. Register \$1 is reserved for the assembler to handle pseudoinstructions and large constants. Hi and Lo are 32-bit registers containing the results of multiply and divide.
2 <sup>30</sup> memory words	Memory[0], Memory[4], . . . , Memory[4294967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers, such as those saved on procedure calls.

### MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add \$1,\$2,\$3	$\$1 = \$2 + \$3$	3 operands; exception possible
	subtract	sub \$1,\$2,\$3	$\$1 = \$2 - \$3$	3 operands; exception possible
	add immediate	addi \$1,\$2,100	$\$1 = \$2 + 100$	+ constant; exception possible
	add unsigned	addu \$1,\$2,\$3	$\$1 = \$2 + \$3$	3 operands; no exceptions
	subtract unsigned	subu \$1,\$2,\$3	$\$1 = \$2 - \$3$	3 operands; no exceptions
	add imm. unsign.	addiu \$1,\$2,100	$\$1 = \$2 + 100$	+ constant; no exceptions
	Move fr. copr. reg.	mfc0 \$1,\$epc	$\$1 = \$epc$	Used to get exception PC
	multiply	mult \$2,\$3	Hi, Lo = $\$2 \times \$3$	64-bit signed product in Hi, Lo
	multiply unsigned	multu \$2,\$3	Hi, Lo = $\$2 \times \$3$	64-bit unsigned product in Hi, Lo
	divide	div \$2,\$3	Lo = $\$2 \div \$3$ , Hi = $\$2 \bmod \$3$	Lo = quotient, Hi = remainder
	divide unsigned	divu \$2,\$3	Lo = $\$2 \div \$3$ , Hi = $\$2 \bmod \$3$	Unsigned quotient and remainder
	Move from Hi	mfhi \$1	$\$1 = \text{Hi}$	Used to get copy of Hi
Move from Lo	mflo \$1	$\$1 = \text{Lo}$	Use to get copy of Lo	
Logical	and	and \$1,\$2,\$3	$\$1 = \$2 \& \$3$	3 register operands; logical AND
	or	or \$1,\$2,\$3	$\$1 = \$2   \$3$	3 register operands; logical OR
	and immediate	and \$1,\$2,100	$\$1 = \$2 \& 100$	Logical AND register, constant
	or immediate	or \$1,\$2,100	$\$1 = \$2   100$	Logical OR register, constant
	shift left logical	sll \$1,\$2,10	$\$1 = \$2 \ll 10$	Shift left by constant
	shift right logical	srl \$1,\$2,10	$\$1 = \$2 \gg 10$	Shift right by constant
Data transfer	load word	lw \$1,100(\$2)	$\$1 = \text{Memory}[\$2+100]$	Data from memory to register
	store word	sw \$1,100(\$2)	$\text{Memory}[\$2+100] = \$1$	Data from register to memory
	load upper imm.	lui \$1,100	$\$1 = 100 \times 2^{16}$	Loads constant in upper 16 bits
Conditional branch	branch on equal	beq \$1,\$2,100	if ( $\$1 == \$2$ ) go to PC+4+100	Equal test; PC relative branch
	branch on not eq.	bne \$1,\$2,100	if ( $\$1 \neq \$2$ ) go to PC+4+100	Not equal test; PC relative
	set on less than	slt \$1,\$2,\$3	if ( $\$2 < \$3$ ) $\$1=1$ ; else $\$1=0$	Compare less than; 2's complement
	set less than imm.	slti \$1,\$2,100	if ( $\$2 < 100$ ) $\$1=1$ ; else $\$1=0$	Compare < constant; 2's comp.
	set less than uns.	sltu \$1,\$2,\$3	if ( $\$2 < \$3$ ) $\$1=1$ ; else $\$1=0$	Compare less than; natural number
set l.t. imm. uns.	sltiu \$1,\$2,100	if ( $\$2 < 100$ ) $\$1=1$ ; else $\$1=0$	Compare < constant; natural	
Unconditional jump	jump	j 10000	go to 10000	Jump to target address
	jump register	jr \$31	go to \$31	For switch, procedure return
	jump and link	jal 10000	$\$31 = \text{PC} + 4$ ; go to 10000	For procedure call

Main MIPS assembly language instruction set. The floating-point instructions are shown in Figure 4.44 on page 241. Appendix A gives the full MIPS assembly language instruction set.