

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	Copy Exception PC + special regs
	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
Data transfer	divide unsigned	divu \$s2,\$s3	Lo = $\$s2 / \$s3$, Hi = $\$s2 \bmod \$s3$	Unsigned quotient and remainder
	move from Hi	mghi \$s1	$\$s1 = \text{Hi}$	Used to get copy of Hi
	move from Lo	mlo \$s1	$\$s1 = \text{Lo}$	Used to get copy of Lo
	load word	lw \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$	Word from memory to register
	store word	sw \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$	Word from register to memory
	load half unsigned	lhu \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$	Halfword memory to register
	store half	sh \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$	Halfword register to memory
	load byte unsigned	lbh \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$	Byte from memory to register
	store byte	sb \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$	Byte from register to memory
	load linked word	ll \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$	Load word as 1st half of atomic swap
Logical	store conditional word	sc \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1; \$s1 = 0$ or 1	Store word as 2nd half atomic swap
	load upper immediate	lui \$s1,100	$\$s1 = 100 \times 2^{16}$	Loads constant in upper 16 bits
	AND	AND \$s1,\$s2,\$s3	$\$s1 = \$s2 \& \$s3$	Three reg. operands; bit-by-bit AND
	OR	OR \$s1,\$s2,\$s3	$\$s1 = \$s2 \mid \$s3$	Three reg. operands; bit-by-bit OR
	NOR	NOR \$s1,\$s2,\$s3	$\$s1 = \sim (\$s2 \mid \$s3)$	Three reg. operands; bit-by-bit NOR
	AND immediate	ANDi \$s1,\$s2,100	$\$s1 = \$s2 \& 100$	Bit-by-bit AND with constant
	OR immediate	ORi \$s1,\$s2,100	$\$s1 = \$s2 \mid 100$	Bit-by-bit OR with constant
	shift left logical	sll \$s1,\$s2,10	$\$s1 = \$s2 \ll 10$	Shift left by constant
Conditional branch	shift right logical	srl \$s1,\$s2,10	$\$s1 = \$s2 \gg 10$	Shift right by constant
	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 ($\$s1 \neq \$s2$) 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	go to 10000	Jump to target address
Unconditional jump	jump register	jr \$ra	go to \$ra	For switch, procedure return
	jump and link	j al 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,\$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. Floating-point machine language instructions are shown in Figure 4.47 on page 291. Appendix A gives the full MIPS machine language.

Register Number	Alternative Name	Description
0	zero	the value 0
1	\$at	(assembler temporary) reserved by the assembler
2-3	\$v0 - \$v1	(values) from expression evaluation and function results
4-7	\$a0 - \$a3	(arguments) First four parameters for subroutine. Not preserved across procedure calls
8-15	\$t0 - \$t7	(temporaries) Caller saved if needed. Subroutines can use w/out saving. Not preserved across procedure calls
16-23	\$s0 - \$s7	(saved values) - Callee saved. A subroutine using one of these must save original and restore it before exiting. Preserved across procedure calls
24-25	\$t8 - \$t9	(temporaries) Caller saved if needed. Subroutines can use w/out saving. These are in addition to \$t0 - \$t7 above. Not preserved across procedure calls.
26-27	\$k0 - \$k1	reserved for use by the interrupt/trap handler
28	\$gp	global pointer. Points to the middle of the 64K block of memory in the static data segment.
29	\$sp	stack pointer Points to last location on the stack.
30	\$s8/\$fp	saved value / frame pointer Preserved across procedure calls
31	\$ra	return address

Service	Code in \$v0	Arguments	Results
print_int	1	\$a0 = integer to be printed	
print_float	2	\$f12 = float to be printed	
print_double	3	\$f12 = double to be printed	
print_string	4	\$a0 = address of string in memory	
read_int	5		integer returned in \$v0
read_float	6		float returned in \$v0
read_double	7		double returned in \$v0
read_string	8	\$a0 = memory address of string input buffer \$a1 = length of string buffer (n)	
sbrk	9	\$a0 = amount	address in \$v0
exit	10		
print char	11	\$a0 = character to print	
read char	12		\$v0 contains char read

MIPS Reference Data

①



CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION (in Verilog)	OPCODE / FUNCT (Hex)
Add	add R	$R[rd] = R[rs] + R[rt]$	(1) 0/20 _{hex}
Add Immediate	addi I	$R[rt] = R[rs] + \text{SignExtImm}$	(1,2) 8 _{hex}
Add Imm. Unsigned	addiu I	$R[rt] = R[rs] + \text{SignExtImm}$	(2) 9 _{hex}
Add Unsigned	addu R	$R[rd] = R[rs] + R[rt]$	0/21 _{hex}
And	and R	$R[rd] = R[rs] \& R[rt]$	0/24 _{hex}
And Immediate	andi I	$R[rt] = R[rs] \& \text{ZeroExtImm}$	(3) c _{hex}
Branch On Equal	beq I	$\text{if}(R[rs] == R[rt])$ $PC = PC + 4 + \text{BranchAddr}$	(4) 4 _{hex}
Branch On Not Equal	bne I	$\text{if}(R[rs] \neq R[rt])$ $PC = PC + 4 + \text{BranchAddr}$	(4) 5 _{hex}
Jump	j J	$PC = \text{JumpAddr}$	(5) 2 _{hex}
Jump And Link	jal J	$R[31] = PC + 8; PC = \text{JumpAddr}$	(5) 3 _{hex}
Jump Register	jlr R	$PC = R[rs]$	0/08 _{hex}
Load Byte Unsigned	lbu I	$R[rt] = \{24'b0, M[R[rs]] + \text{SignExtImm}\}(7:0)$	(2) 24 _{hex}
Load Halfword Unsigned	lhu I	$R[rt] = \{16'b0, M[R[rs]] + \text{SignExtImm}\}(15:0)$	(2) 25 _{hex}
Load Linked	ll I	$R[rt] = M[R[rs] + \text{SignExtImm}]$	(2,7) 30 _{hex}
Load Upper Imm.	lui I	$R[rt] = \{\text{imm}, 16'b0\}$	f _{hex}
Load Word	lw I	$R[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 23 _{hex}
Nor	nor R	$R[rd] = \sim (R[rs] R[rt])$	0/27 _{hex}
Or	or R	$R[rd] = R[rs] R[rt]$	0/25 _{hex}
Or Immediate	ori I	$R[rt] = R[rs] \text{ZeroExtImm}$	(3) d _{hex}
Set Less Than	slt R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	0/2a _{hex}
Set Less Than Imm.	slti I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	a _{hex}
Set Less Than Imm. Unsigned	sltiu I	$R[rt] = (R[rs] < \text{SignExtImm}) ? 1 : 0$	(2,6) b _{hex}
Set Less Than Unsig.	sltu R	$R[rd] = (R[rs] < R[rt]) ? 1 : 0$	(6) 0/2b _{hex}
Shift Left Logical	sll R	$R[rd] = R[rt] \ll \text{shamt}$	0/00 _{hex}
Shift Right Logical	srl R	$R[rd] = R[rt] \gg \text{shamt}$	0/02 _{hex}
Store Byte	sb I	$M[R[rs] + \text{SignExtImm}](7:0) = R[rt](7:0)$	(2) 28 _{hex}
Store Conditional	sc I	$M[R[rs] + \text{SignExtImm}] = R[rt];$ $R[rt] = (\text{atomic}) ? 1 : 0$	(2,7) 38 _{hex}
Store Halfword	sh I	$M[R[rs] + \text{SignExtImm}](15:0) = R[rt](15:0)$	(2) 29 _{hex}
Store Word	sw I	$M[R[rs] + \text{SignExtImm}] = R[rt]$	(2) 2b _{hex}
Subtract	sub R	$R[rd] = R[rs] - R[rt]$	(1) 0/22 _{hex}
Subtract Unsigned	subu R	$R[rd] = R[rs] - R[rt]$	0/23 _{hex}

- (1) May cause overflow exception
 (2) $\text{SignExtImm} = \{16\{\text{immediate}[15]\}, \text{immediate}\}$
 (3) $\text{ZeroExtImm} = \{16\{1'b0\}, \text{immediate}\}$
 (4) $\text{BranchAddr} = \{14\{\text{immediate}[15]\}, \text{immediate}, 2'b0\}$
 (5) $\text{JumpAddr} = \{PC + 4[31:28], \text{address}, 2'b0\}$
 (6) Operands considered unsigned numbers (vs. 2's comp.)
 (7) Atomic test&set pair; $R[rt] = 1$ if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

R	opcode	rs	rt	rd	shamt	funct
	31	26 25	21 20	16 15	11 10	6 5
I	opcode	rs	rt	immediate		
	31	26 25	21 20	16 15		
J	opcode	address				
	31	26 25				

ARITHMETIC CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION	OPCODE / FUNCT (Hex)
Branch On FP True	bc1t FI	$\text{if}(FPcond) PC = PC + 4 + \text{BranchAddr}$	(4) 11/8/1/1
Branch On FP False	bc1f FI	$\text{if}(!FPcond) PC = PC + 4 + \text{BranchAddr}$	(4) 11/8/0/1
Divide	div R	$Lo = R[rs]/R[rt]; Hi = R[rs]\%R[rt]$	0/0/1/1a
Divide Unsigned	divu R	$Lo = R[rs]/R[rt]; Hi = R[rs]\%R[rt]$	(6) 0/0/1/1b
FP Add Single	add.s FR	$F[fd] = F[fs] + F[ft]$	11/10/0/0
FP Add Double	add.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} + \{F[ft], F[ft+1]\}$	11/11/0/0
FP Compare Single	c.x.s* FR	$FPcond = (F[fs] op F[ft]) ? 1 : 0$	11/10/0/y
FP Compare Double	c.x.d* FR	$FPcond = (\{F[fs], F[fs+1]\} op \{F[ft], F[ft+1]\}) ? 1 : 0$	11/11/0/y
* (x is eq, lt, or le) (op is ==, <, or <=) (y is 32, 3c, or 3e)			
FP Divide Single	div.s FR	$F[fd] = F[fs] / F[ft]$	11/10/1/3
FP Divide Double	div.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} / \{F[ft], F[ft+1]\}$	11/11/1/3
FP Multiply Single	mul.s FR	$F[fd] = F[fs] * F[ft]$	11/10/1/2
FP Multiply Double	mul.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} * \{F[ft], F[ft+1]\}$	11/11/1/2
FP Subtract Single	sub.s FR	$F[fd] = F[fs] - F[ft]$	11/10/1/1
FP Subtract Double	sub.d FR	$\{F[fd], F[fd+1]\} = \{F[fs], F[fs+1]\} - \{F[ft], F[ft+1]\}$	11/11/1/1
Load FP Single	lwc1 I	$F[rt] = M[R[rs] + \text{SignExtImm}]$	(2) 31/1/1/1
Load FP Double	ldc1 I	$F[rt+1] = M[R[rs] + \text{SignExtImm}]; F[rt] = M[R[rs] + \text{SignExtImm} + 4]$	(2) 35/1/1/1
Move From Hi	mhi R	$R[rd] = Hi$	0/1/1/10
Move From Lo	mlo R	$R[rd] = Lo$	0/1/1/12
Move From Control	mfc0 R	$R[rd] = CR[rs]$	10/0/0/0
Multiply	mult R	$\{Hi, Lo\} = R[rs] * R[rt]$	0/1/1/18
Multiply Unsigned	multu R	$\{Hi, Lo\} = R[rs] * R[rt]$	(6) 0/1/1/19
Shift Right Arith.	sra R	$R[rd] = R[rt] \gg \text{shamt}$	0/1/1/3
Store FP Single	swc1 I	$M[R[rs] + \text{SignExtImm}] = F[rt]$	(2) 39/1/1/1
Store FP Double	sdc1 I	$M[R[rs] + \text{SignExtImm}] = F[rt]; M[R[rs] + \text{SignExtImm} + 4] = F[rt+1]$	(2) 3d/1/1/1

FLOATING-POINT INSTRUCTION FORMATS

FR	opcode	fmt	ft	fs	fd	funct
	31	26 25	21 20	16 15	11 10	6 5
FI	opcode	fmt	ft	immediate		
	31	26 25	21 20	16 15		

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	blt	$\text{if}(R[rs] < R[rt]) PC = \text{Label}$
Branch Greater Than	bgt	$\text{if}(R[rs] > R[rt]) PC = \text{Label}$
Branch Less Than or Equal	ble	$\text{if}(R[rs] \leq R[rt]) PC = \text{Label}$
Branch Greater Than or Equal	bge	$\text{if}(R[rs] \geq R[rt]) PC = \text{Label}$
Load Immediate	li	$R[rd] = \text{immediate}$
Move	move	$R[rd] = R[rs]$

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
\$zero	0	The Constant Value 0	N.A.
\$at	1	Assembler Temporary	No
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation	No
\$a0-\$a3	4-7	Arguments	No
\$t0-\$t7	8-15	Temporaries	No
\$s0-\$s7	16-23	Saved Temporaries	Yes
\$t8-\$t9	24-25	Temporaries	No
\$k0-\$k1	26-27	Reserved for OS Kernel	No
\$gp	28	Global Pointer	Yes
\$sp	29	Stack Pointer	Yes
\$fp	30	Frame Pointer	Yes
\$ra	31	Return Address	Yes