SPIM Instruction Set

Instructions and PseudoInstructions

This list of MIPS instructions and pseudoinstructions is not complete. Notably missing are some of the Floating Point and other coprocessor instructions.

• - Indicates an actual MIPS instruction. Others are SPIM PseudoInstructions.

	Instruction		Function
•	add	Rd, Rs, Rt	Rd = Rs + Rt (signed)
•	addu	Rd, Rs, Rt	Rd = Rs + Rt (unsigned)
•	addi	Rd, Rs, Imm	Rd = Rs + Imm (signed)
•	sub	Rd, Rs, Rt	Rd = Rs - Rt (signed)
•	subu	Rd, Rs, Rt	Rd = Rs - Rt (unsigned)
•	div	Rs, Rt	lo = Rs/Rt, hi = Rs **mod** Rt (integer division, signed)
•	divu	Rs, Rt	lo = Rs/Rt, hi = Rs **mod** Rt (integer division, unsigned)
	div	Rd, Rs, Rt	Rd = Rs/Rt (integer division, signed)
	divu	Rd, Rs, Rt	Rd = Rs/Rt (integer division, unsigned)
	rem	Rd, Rs, Rt	Rd = Rs **mod** Rt (signed)
	remu	Rd, Rs, Rt	Rd = Rs **mod** Rt (unsigned)
	mul	Rd, Rs, Rt	Rd = Rs * Rt (signed)
•	mult	Rs, Rt	hi, lo = Rs * Rt (signed, hi = high 32 bits, lo = low 32 bits)
•	multu	Rd, Rs	hi, lo = Rs * Rt (unsigned, hi = high 32 bits, lo = low 32 bits)
•	and	Rd, Rs, Rt	Rd = Rs • Rt
•	andi	Rd, Rs, Imm	Rd = Rs • Imm
	neg	Rd, Rs	Rd = -(Rs)
•	nor	Rd, Rs, Rt	Rd = (Rs + Rt)'
	not	Rd, Rs	Rd = (Rs)'
•	or	Rd, Rs, Rt	Rd = Rs + Rt
•	ori	Rd, Rs, Imm	Rd = Rs + Imm
•	xor	Rd, Rs, Rt	Rd = Rs ⊕ Rt
•	xori	Rd, Rs, Imm	Rd = Rs ⊕ Imm
•	sll	Rd, Rt, Sa	Rd = Rt left shifted by Sa bits
•	sllv	Rd, Rs, Rt	Rd = Rt left shifted by Rs bits
•	srl	Rd, Rs, Sa	Rd = Rt right shifted by Sa bits
•	srlv	Rd, Rs, Rt	Rd = Rt right shifted by Rs bits
	move	Rd, Rs	Rd = Rs
•	mfhi	Rd	Rd = hi
•	mflo	Rd	Rd = lo
	li	Rd, Imm	Rd = Imm
•	lui	Rt, Imm	Rt[31:16] = Imm, Rt[15:0] = 0
•	lb	Rt, Address(Rs)	Rt = byte at M[Address + Rs] (sign extended)
•	sb	Rt, Address(Rs)	Byte at M[Address + Rs] = Rt (sign extended)
•	lw	Rt, Address(Rs)	Rt = word at M[Address + Rs]
•	sw	Rt, Address(Rs)	Word at M[Address + Rs] = Rt
•	slt	Rd, Rs, Rt	Rd = 1 if Rs < Rt, Rd = 0 if Rs ≥ Rt (signed)

- slti Rd, Rs, Imm Rd = 1 if Rs < Imm, Rd = 0 if Rs ≥ Imm (signed)
- sltu Rd, Rs, Rt Rd = 1 if Rs < Rt, Rd = 0 if Rs ≥ Rt (unsigned)

- beq Rs, Rt, Label Branch to Label if Rs = Rt
  beqz Rs, Label Branch to Label if Rs = 0
  bge Rs, Rt, Label Branch to Label if Rs ≥ Rt (signed)
- bgez Rs, Label Branch to Label if Rs ≥ 0 (signed)
- bgezal Rs, Label Branch to Label and Link if Rs ≥ Rt (signed)
  bgt Rs, Rt, Label Branch to Label if Rs > Rt (signed)
  bgtu Rs, Rt, Label Branch to Label if Rs > Rt (unsigned)
- bgtz Rs, Label Branch to Label if Rs > 0 (signed)
  ble Rs, Rt, Label Branch to Label if Rs ≤ Rt (signed)
  bleu Rs, Rt, Label Branch to Label if Rs ≤ Rt (unsigned)
- blez Rs, Label Branch to Label if Rs ≤ 0 (signed)
- bgezal Rs, Label Branch to Label and Link if Rs ≥ 0 (signed)
- bltzal Rs, Label Branch to Label and Link if Rs < 0 (signed)
  blt Rs, Rt, Label Branch to Label if Rs < Rt (signed)
  bltu Rs, Rt, Label Branch to Label if Rs < Rt (unsigned)
- bltz Rs, Label Branch to Label if Rs < 0 (signed)
- bne Rs, Rt, Label Branch to Label if Rs ≠ Rt
  bnez Rs, Label Branch to Label if Rs ≠ 0

- j Label Jump to Label unconditionally
- jal Label Jump to Label and link unconditionally
- jr Rs Jump to location in Rs unconditionally
- jalr Label Jump to location in Rs and link unconditionally

System I/O Services: syscall

Service	Code in $v0	Argument(s)	Result(s)
Print Integer	1	$a0 = number to be printed	
Print Float	2	$f12 = number to be printed	
Print Double	3	$f12 = number to be printed	
Print String	4	$a0 = address of string in memory	
Read Integer	5		number returned in $v0
Read Float	6		number returned in $f0
Read Double	7		number returned in $f0
Read String	8	$a0 = address of input buffer in memory	
		$a1 = length of buffer (n)	
Sbrk	9	$a0 = amount	address in $v0
Exit	10		

Registers

By convention, many MIPS registers have special purpose uses. To help clarify this, SPIM defines aliases for each register that represent its purpose. The following table lists these aliases and the commonly accepted uses for the registers.

Register	Number	Usage
zero	0	Constant 0
at	1	Reserved for the assembler
v0	2	Used for return values from function calls.
v1	3	
a0	4	Used to pass arguments to procedures and functions.
a1	5	
a2	6	
a3	7	
t0	8	Temporary (Caller-saved, need not be saved by called procedure)
t1	9	
t2	10	
t3	11	
t4	12	
t5	13	
t6	14	
t7	15	
s0	16	Saved temporary (Callee-saved, called procedure must save and restore)
s1	17	
s2	18	
s3	19	
s4	20	
s5	21	
s6	22	
s7	23	
t8	24	Temporary (Caller-saved, need not be saved by called procedure)
t9	25	
k0	26	Reserved for OS kernel
k1	27	
gp	28	Pointer to global area
sp	29	Stack pointer
fp	30	Frame pointer
ra	31	Return address for function calls.

Decimal	Hex	Octal	Binary	Decimal	Hex	Octal	Binary
0	0	0	0000	8	8	10	1000
1	1	1	0001	9	9	11	1001
2	2	2	0010	10	A	12	1010
3	3	3	0011	11	B	13	1011
4	4	4	0100	12	C	14	1100
5	5	5	0101	13	D	15	1101
6	6	6	0110	14	E	16	1110
7	7	7	0111	15	F	17	1111

Floating Point Instructions and PseudoInstructions

The MIPS floating point coprocessor is coprocessor number 1. It operates on single precision (32-bit) and double precision (64-bit) floating point numbers. This coprocessor has its own registers, numbered $f0-$f31. Even numbered registers are used for double precision operations.

• - Indicates an actual MIPS instruction. Others are SPIM pseudo instructions.

	Instruction		Function
•	abs.d	FRdest, FRsrc	FRdest = | FRsrc |
•	abs.s	FRdest, FRsrc	FRdest = | FRsrc |
•	add.d	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 + FRsrc2
•	add.s	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 + FRsrc2
•	c.eq.d	FRsrc1, FRsrc2	Set flag on FRsrc1 = FRsrc2
•	c.eq.s	FRsrc1, FRsrc2	Set flag on FRsrc1 = FRsrc2
•	c.le.d	FRsrc1, FRsrc2	Set flag on FRsrc1 ≤ FRsrc2
•	c.le.s	FRsrc1, FRsrc2	Set flag on FRsrc1 ≤ FRsrc2
•	c.lt.d	FRsrc1, FRsrc2	Set flag on FRsrc1 < FRsrc2
•	c.lt.s	FRsrc1, FRsrc2	Set flag on FRsrc1 < FRsrc2
•	cvt.d.s	FRdest, FRsrc	Double ← Single
•	cvt.d.w	FRdest, FRsrc	Double ← Integer
•	cvt.s.d	FRdest, FRsrc	Single ← Double
•	cvt.s.w	FRdest, FRsrc	Single ← Integer
•	cvt.w.d	FRdest, FRsrc	Integer ← Double
•	cvt.w.s	FRdest, FRsrc	Integer ← Single
•	div.d	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 / FRsrc2
•	div.s	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 / FRsrc2
	l.d	FRdest, address	FRdest = address
	l.s	FRdest, address	FRdest = address
•	mov.d	FRdest, FRsrc	FRdest = FRsrc
•	mov.s	FRdest, FRsrc	FRdest = FRsrc
•	mul.d	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 * FRsrc2
•	mul.s	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 * FRsrc2
•	neg.d	FRdest, FRsrc	FRdest = - FRsrc
•	neg.s	FRdest, FRsrc	FRdest = - FRsrc
	s.d	FRdest, address	address ← FRdest
	s.s	FRdest, address	address ← FRdest
•	sub.d	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 – FRsrc2
•	sub.s	FRdest, FRsrc1, FRsrc2	FRdest = FRsrc1 – FRsrc2
•	bc1t	Label	Branch conditional; if flag is True; to Label
•	bc1f	Label	Branch conditional; if flag is False; to Label (1 denotes coprocessor 1, the floating point coprocessor)
•	mfc1	Rdest, FPsrc	Move contents of register FRsrc to CPU register Rdest
•	mtc1	Rsrc, FPdest	Move contents of CPU register Rsrc to register FPdest