

MIPS/SPIM Reference Card

Rev. A., CP 20150417

CORE INSTRUCTION SET (INCLUDING A FEW PSEUDO-INSTRUCTIONS)

NAME	MNEUMONIC	FORMAT	OPERATION (Notes)	OPCODE/ FUNCT(Hex)
Add	add	R	$R[rd]=R[rs]+R[rt]$ (1)	0/20
Add Immediate	addi	I	$R[rt]=R[rs]+SignExtImm$ (1)(2)	08
Add Imm. Unsigned	addiu	I	$R[rt]=R[rs]+SignExtImm$ (2)	09
Add Unsigned	addu	R	$R[rd]=R[rs]+R[rt]$ (2)	0/21
Subtract	sub	R	$R[rd]=R[rs]-R[rt]$ (1)	0/22
Subtract Unsigned	subu	R	$R[rd]=R[rs]-R[rt]$	0/23
And	and	R	$R[rd]=R[rs]\&R[rt]$	0/24
And Immediate	andi	I	$R[rt]=R[rs]\&ZeroExtImm$ (3)	0C
Nor	nor	R	$R[rd]=\sim(R[rs])\mid R[rt]$	0/27
Or	or	R	$R[rd]=R[rs]\mid R[rt]$	0/25
Or Immediate	ori	I	$R[rt]=R[rs]\mid ZeroExtImm$ (3)	0D
Xor	xor	R	$R[rd]=R[rs]\wedge R[rt]$	0/26
Xor Immediate	xori	I	$R[rt]=R[rs]\wedge ZeroExtImm$	0E
Shift Left Logical	sll	R	$R[rd]=R[rt] \ll shamt$	0/00
Shift Right Logical	srl	R	$R[rd]=R[rt] \gg shamt$	0/02
Shift Right Arithmetic	sra	R	$R[rd]=R[rt] \gg shamt$	0/03
Shift Left Logical Var.	sllv	R	$R[rd]=R[rt] \ll R[rs]$	0/04
Shift Right Logical Var.	srlv	R	$R[rd]=R[rt] \ll R[rs]$	0/06
Shift Right Arithmetic Var.	srav	R	$R[rd]=R[rt] \gg R[rs]$	0/07
Set Less Than	slt	R	$R[rd]=(R[rs]<R[rt])?1:0$	0/2A
Set Less Than Imm.	slti	I	$R[rt]=(R[rs]<SignExtImm)?1:0$ (2)	0A
Set Less Than Imm. Unsign.	sltiu	I	$R[rt]=(R[rs]<SignExtImm)?1:0$ (2)(6)	0B
Set Less Than Unsigned	sltu	R	$R[rd]=(R[rs]<R[rt])?1:0$ (6)	0/2B
Branch On Equal	beq	I	if($R[rs]=R[rt]$) $PC=PC+4+BranchAddr$ (4)	04
Branch On Not Equal	bne	I	if($R[rs]\neq R[rt]$) $PC=PC+4+BranchAddr$ (4)	05
Branch Less Than	blt	P	if($R[rs]<R[rt]$) $PC=PC+4+BranchAddr$	
Branch Greater Than	bgt	P	if($R[rs]>R[rt]$) $PC=PC+4+BranchAddr$	
Branch Less Than Or Equal	ble	P	if($R[rs]\leq R[rt]$) $PC=PC+4+BranchAddr$	
Branch Greater Than Or Equal	bge	P	if($R[rs]\geq R[rt]$) $PC=PC+4+BranchAddr$	
Jump	j	J	$PC=JumpAddr$ (5)	02
Jump And Link	jal	J	$R[31]=PC+4;(5) PC=JumpAddr$	03
Jump Register	jr	R	$PC=R[rs] R[31]=PC+4;$	0/08
Jump And Link Register	jalr	R	$PC=R[rs]$	0/09
Move	move	P	$R[rd]=R[rs]$	
Load Byte	lb	I	$R[rt]=signextend(M[R[rs]+SignExtImm](7:0))$ (2)	20
Load Byte Unsigned	lbu	I	$R[rt]=\{24'b0, M[R[rs]+SignExtImm](7:0)\}$ (2)	24
Load Halfword	lh	I	$R[rt]=signextend(M[R[rs]+SignExtImm](15:0))$ (2)	21
Load Halfword Unsigned	lhu	I	$R[rt]=\{16'b0, M[R[rs]+SignExtImm](15:0)\}$ (2)	25
Load Upper Imm.	lui	I	$R[rt]=\{immediate, 16'b0\}$	0F
Load Word	lw	I	$R[rt]=M[R[rs]+SignExtImm]$ (2)	23
Load Immediate	li	I	$R[rd]=immediate$	
Load Address	la	P	$R[rd]=immediate$	
Store Byte	sb	I	$M[R[rs]+SignExtImm]=R[rt](7:0)$ (2)	28
Store Halfword	sh	I	$M[R[rs]+SignExtImm]=R[rt](15:0)$ (2)	29
Store Word	sw	I	$M[R[rs]+SignExtImm]=R[rt]$ (2)	2B

REGISTERS

NAME	NMBR	USE	STORE?
\$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
\$f0-\$f31	0-31	Floating Point Registers	Yes

(1) May cause overflow exception

(2) $SignExtImm = \{16\{immediate[15]\}, immediate\}$

(3) $ZeroExtImm = \{16\{1b'0\}, immediate\}$

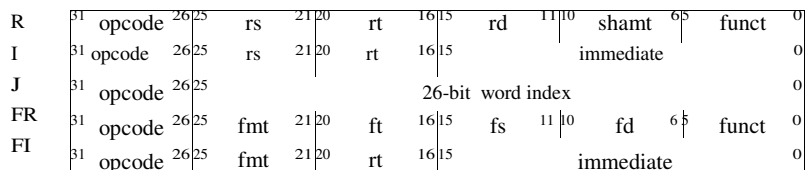
(4) $BranchAddr = \{14\{immediate[15]\}, immediate, 2'b0\}$

(5) $JumpAddr = \{PC[31:28], address, 2'b0\}$

(6) Operands considered unsigned numbers (vs. 2 s comp.)

BASIC INSTRUCTION FORMATS,

FLOATING POINT INSTRUCTION FORMATS



ARITHMETIC CORE INSTRUCTION SET

NAME	MNE- MON- IC	FOR- MAT	OPERATION (in Verilog)	OPCODE/ FMT/FT/ FUNCT
Divide	div	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	0/-/-1a
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6) 0/-/-1b
Multiply	mult	R	{Hi,Lo}=R[rs]*R[rt]	0/-/-18
Multiply Unsigned	multu	R	{Hi,Lo}=R[rs]*R[rt]	(6) 0/-/-19
Branch On FP True	bc1t	FI	if(FPCond) PC=PC+4+BranchAddr	(4) 11/8/1/-
Branch On FP False	bc1f	FR	if(!FPCond) PC=PC+4+BranchAddr	(4) 11/8/0/-
FP Compare Single	c.x.s*	FR	FPCond=(F[fs] op F[ft])?1:0	11/10/-/y
FP Compare Double	c.x.d*	FR	FPCond=((F[fs],F[fs+1]) op {F[ft],F[ft+1]})?1:0 *(x is eq, lt or le) (op is ==, < or <=) (y is 32, 3c or 3e)	11/11/-/y
FP Add Single	add.s	FR	F[fd]=F[fs]+F[ft]	11/10/-/0
FP Divide Single FP	div.s	FR	F[fd]=F[fs]/F[ft]	11/10/-/3
Multiply Single FP	mul.s	FR	F[fd]=F[fs]*F[ft]	11/10/-/2
Subtract Single FP	sub.s	FR	F[fd]=F[fs]-F[ft]	11/10/-/1
Add Double	add.d	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}+{F[ft],F[ft+1]}	11/11/-/0
FP Divide Double FP	div.d	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}/{F[ft],F[ft+1]}	11/11/-/3
Multiply Double FP	mul.d	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}*{F[ft],F[ft+1]}	11/11/-/2
Subtract Double	sub.d	FR	{F[fd],F[fd+1]}={F[fs],F[fs+1]}-{F[ft],F[ft+1]}	11/11/-/1
Move From Hi	mfhi	R	R[rd]=Hi	0/-/-10
Move From Lo	mflo	R	R[rd]=Lo	0/-/-12
Move From Control	mfc0	R	R[rd]=CR[rs]	16/0/-/0
Load FP Single	lwc1	I	F[rt]=M[R[rs]+SignExtImm]	(2) 31/-/-/-
Load FP Double	ldc1	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2) 35/-/-/-
Store FP Single	swc1	I	M[R[rs]+SignExtImm]=F[rt]	(2) 39/-/-/-
Store FP Double	sdc1	I	M[R[rs]+SignExtImm]=F[rt]; M[R[rs]+SignExtImm+4]=F[rt+1]	(2) 3d/-/-/-

ASSEMBLER DIRECTIVES

.data [addr]*	Subsequent items are stored in the data segment
.kdata [addr]*	Subsequent items are stored in the kernel data segment
.ktext [addr]*	Subsequent items are stored in the kernel text segment
.text [addr]*	Subsequent items are stored in the text * starting at [addr] if specified
.ascii str	Store string str in memory, but do not null-terminate it
.asciiz str	Store string str in memory and null-terminate it
.byte b1,...,bn	Store the n values in successive bytes of memory
.double d1,...,dn	Store the n floating-point double precision numbers in successive memory locations
.float f1,...,fn	Store the n floating-point single precision numbers in successive memory locations
.half h1,...,hn	Store the n 16-bit quantities in successive memory halfwords
.word w1,...,wn	Store the n 32-bit quantities in successive memory words
.space n	Allocate n bytes of space in the current segment
.extern symsize	Declare that the datum stored at sym is size bytes large and is a global label
.globl sym	Declare that label sym is global and can be referenced from other files
.align n	Align the next datum on a 2 ⁿ byte boundary, until the next .data or .kdata directive
.set at	Tells SPIM to complain if subsequent instructions use \$at
.set noat	prevents SPIM from complaining if subsequent instructions use \$at

SYSCALLS

SERVICE	\$v0	ARGS	RESULT
print_int	1	integer \$a0	
print_float	2	float \$f12	
print_double	3	double \$f12/\$f13	
print_string	4	string \$a0	
read_int	5		integer (in \$v0)
read_float	6		float (in \$f0)
read_double	7		double (in \$f0)
read_string	8	buf \$a0, buflen \$a1	
sbrk	9	amount \$a	address (in \$v0)
exit	10		

EXCEPTION CODES

Number	Name	Cause of Exception
0	Int	Interrupt (hardware)
4	AdEL	Address Error Exception (load or instruction fetch)
5	AdES	Address Error Exception (store)
6	IBE	Bus Error on Instruction Fetch
7	DBE	Bus Error on Load or Store
8	Sys	Syscall Exception
9	Bp	Breakpoint Exception
10	RI	Reserved Instruction Exception
11	CpU	Coprocessor Unimplemented
12	Ov	Arithmetic Overflow Exception
13	Tr	Trap
15	FPE	Floating Point Exception

[1] Patterson, David A; Hennessy, John J.: Computer Organization and Design, 3rd Edition. Morgan Kaufmann Publishers. San Francisco, 2005.