MIPS/SPIM Reference Card

Example instruction Corresponding meaning

CORE INSTRUCTION SET (INCLUDING PSEUDO INSTRUCTIONS)

	MNE-	FOR-		OPCODE/
	MON-	MAT		FUNCT
NAME	IC		OPERATION (in Verilog)	(Hex)
Add	add	R	R[rd]=R[rs]+R[rt] (1)	
Add Immediate	addi	I	add $\$t1,\$s1,\$s2$ R[rt]=R[rs]+SignExtImm (1)(2)	
Add Imm. Unsigned	addiu	I	$\begin{array}{c} \text{$t1=$s1+$s2} \\ \text{$R[rt]=R[rs]+SignExtImm} \end{array} $ (2)	
Add Unsigned	addu	R	$\begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \text{addi } \$t1,\$s1,10 \end{array} \end{array} R[rd] = R[rs] + R[rt] \end{array} $	
Subtract	sub	R	$\begin{cases} s_{t1} = s_{s1+10} \end{cases} R[rd] = R[rs] - R[rt] $ (1)	l l
Subtract Unsigned	subu	R	R[rd]=R[rs]-R[rt]	0/23
And	and	R		0/24
And Immediate	andi	I	and $t1, t2$ R[rt]=R[rs]&ZeroExtImm (3)	
Nor	nor	R	$\begin{array}{c} \$t1=\$s1\bullet\$s2 \\ R[rd]=\sim(R[rs] R[rt]) \end{array}$	0/27
Or	or	R	andi \$t1,\$s1,10 R[rd]=R[rs] R[rt]	0/25
Or Immediate	ori	I	$\begin{array}{ c c c c c c } \hline & \$t1 = \$s1 \cdot 10 \\ \hline & R[rt] = R[rs] ZeroExtImm \\ \hline & (3) \\ \hline \end{array}$	
Xor	xor	R	$R[rd]=R[rs]^R[rt]$	0/26
Xor Immediate	xori	I	R[rt]=R[rs]^ZeroExtImm	e
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	sllv \$t1,\$s1,\$s2 $R[rd]=R[rt] \ll R[rs]$	0/04
Shift Right Logical Var.	srlv	R	$$t1=$s1<<$s2$ $R[rd]=R[rt]\gg R[rs]$	0/06
Shift Right Arithmetic Var.	srav	R	$R[rd]=R[rt]\gg R[rs]$	0/07
Set Less Than	slt	R	R[rd]=(R[rs] <r[rt])?1:0< td=""><td>0/2a</td></r[rt])?1:0<>	0/2a
Set Less Than Imm.	slti	I	$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	
Set Less Than Imm. Unsign.	sltiu	I	\$t1=1 if \$s1<\$s2 R[rt]=(R[rs] <signextimm)?1:0 <="" math=""> (2)(6)</signextimm)?1:0>	
Set Less Than Unsigned	sltu	R	else $$t1=0$ $R[rd]=(R[rs]< R[rt])$?1:0 (2)(0)	
Branch On Equal	beq	I	if(R[rs] == R[rt]) PC = PC + 4 + BranchAddr (4)	4
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr (4)	
Branch Less Than	blt	P	if(P[rs] < P[rt]) PC-PC (4 Propch Addr beq \$\$1,\$\$2,10	
Branch Greater Than	bgt	P	if(R[rs]>R[rt]) PC=PC+4+BranchAddr if \$s1=\$s2 PC=PC+4+(4*10)	
Branch Less Than Or Equal	ble	P	$if(R[rs] \le R[rt]) PC = PC + 4 + BranchAddr$	
Branch Greater Than Or Equal	bge	P	if(R[rs]>=R[rt]) PC=PC+4+BranchAddr	
Jump	j	J	PC=JumpAddr (5)	2
Jump And Link	jal	J	$D[21] = DC_1 A_2$ $\begin{bmatrix} 3 & 2500 \\ \end{bmatrix}$ (5)	
-			PC = 4*2500 $PC = 4*2500$	
Jump Register	jr	R	PC=R[rs] jalr \$t1	0/08
Jump And Link Register	jalr	R	R[31]=PC+4; $ra = PC; PC = $t1$	0/09
			PC=R[rs]	
Move	move	P	(move rd, rs) R[rd]=R[rs]	
Load Byte	lb	I	$R[rt] = \{24'b0, M[R[rs] + SignExtImm](7:0)\} $ (3)	20
Load Byte Unsigned	lbu	I		24
Load Halfword	lh	I	$s1 = Mem[100+ss2]$ $R[rt] = \{16'b0, M[R[rs] + SignExtImm](15:0)\}$ (3)	
Load Halfword Unsigned	lhu	I	$\begin{cases} 1 \text{ui } \$s1,100 \\ \text{R[rt]} = \{16\text{'b0}, \text{M[R[rs]} + \text{ZeroExtImm]}(15\text{:0}) \} \end{cases} $ (2)	
Load Upper Imm.	lui	I	\$\s1 = \frac{100*2^16}{\text{R[rt]} = \text{\text{imm}}, 16'b0\}	f
Load Word	lw	I	R[rt]=M[R[rs]+SignExtImm] (2)	23
Load Immediate	li	P	R[rd]=immediate	
Load Address	la	P	R[rd]=immediate	
Store Byte	sb	I	M[R[rs]+SignExtImm] (7:0)=R[rt](7:0) $Sw $s1,100 ($s2)$ (2)	28
Store Halfword	sh	I	$M[R[rs]+SignExtImm] (15:0)=R[rt](15:0) \begin{bmatrix} sw & $s1,100 & ($s2) \\ Mem[100+$s2]=$s1 \end{bmatrix} $ (2)	
Store Word	SW	I	M[R[rs]+SignExtImm]=R[rt] (2)	
·				

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	No
		Expression Evaluation	
\$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}
- (4) JumpAddr = {PC[31:28], address, 2'b0}
- (6) Operands considered unsigned numbers (vs. 2 s comp.)

BASIC INSTRUCTION FORMATS, FLOATING POINT INSTRUCTION FORMATS

R	31 opcode 26 25	rs	21 20	rt	1615	rd	11110	shamt	65	funct	0
I	³¹ opcode ²⁶ ²⁵	rs	21 20	rt	1615			immediat	te		σ
J	31 opcode 26 25				in	nmedi	ate				σ
FR	31 opcode 26 25	fmt	21 20	ft	1615	fs	11110	fd	65	funct	0
FI	31 opcode 26 ²⁵	fmt	21 20	rt	1615			immediat	te		0

FORMAT=P: PSEUDO-INSTRUCTION

ARITHMETIC CORE INSTRUCTION SET

	MNE-	FOR-		OPCODE/		
	MON-	MAT		FMT/FT/		
NAME	IC		OPERATION (in Verilog)			
Divide	div	R	Lo=R[rs]/R[rt]; div \$s2,\$s3	0/–/–/1a		
			Hi=R[rs]%R[rt] $ \frac{\text{div $32,$33}}{\text{LO} = \$s2/\$s3} $			
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; $HI = $s2 \mod $s3$ Note: use with (6)	0/–/–/1b		
			Hi=R[rs]%R[rt] mul \$\$2,\$\$3 mfhi and mflo			
Multiply	mult	R	$\{Hi,Lo\}=R[rs]*R[rt] \qquad Hi:LO=\$s2*\$s3$	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] <i>op</i> 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	11/11/–/y		
			*(x is eq, 1t or 1e) (op is ==, < or <=) (y is 32, 3c or 3e)			
FP Add Single	add.s	FR	F[fd]=F[fs]+F[ft]	11/10/–/0		
FP Divide Single	div.s	FR	F[fd]=F[fs]/F[ft] Note: FP usage similar to	11/10/–/3		
FP Multiply Single	mul.s	FR	F[fd]=F[fs]*F[ft] corresponding INTEGER usage	11/10/–/2		
FP Subtract Single	sub.s	FR	F[fd]=F[fs]-F[ft]	11/10/–/1		
FP 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	div.d	FR	${F[fd],F[fd+1]}={F[fs],F[fs+1]}/{F[ft],F[ft+1]}$	11/11/–/3		
FP Multiply Double	mul.d	FR	${F[fd],F[fd+1]}={F[fs],F[fs+1]}*{F[ft],F[ft+1]}$	11/11/–/2		
FP 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	mfhi \$s1 R[rd]=Hi	0/-/-/10		
Move From Lo	mflo	R	\$s1 = HI	0/-/-/12		
Move From Control	mfc0	R	R[rd]=CR[rs] also mfc1, mtc1	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]; (2)	35/-/-/-		
			F[rt+1]=M[R[rs]+SignExtImm+4]			
Store FP Single	swc1	I	M[R[rs]+SignExtImm]=F[rt] (2)	39/–/–/–		
Store FP Double	sdc1	I	M[R[rs]+SignExtImm]=F[rt]; (2)	3d/-/-/-		
			M[R[rs]+SignExtImm+4]=F[rt+1]			

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 b_1,\ldots,b_n	Store the <i>n</i> values in successive bytes of memory
double d_1,\ldots,d_n	Store the n floating-point double precision numbers in successive memory locations
.float f_1,\ldots,f_1	Store the n floating-point single precision numbers in successive memory locations
.half h_1,\ldots,h_n	Store the n 16-bit quantities in successive memory halfwords
word w_1,\ldots,w_n	Store the <i>n</i> 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				