MIPS Instruction Reference Data
Connor Baker, January 20th 2017

Core Instruction Set

Name	Mnemonic	Format	Operation (in Verilog)	OPCODE/Funct (Hex)
Add	add	R	R[rd]=R[rs]+R[rt]	(1) $0/20_{16}$
Add Immediate	addi	I	R[rt]=R[rs]+SigExtImm	$(1,2)$ 8_{16}
Add Imm. Unsigned	addiu	I	R[rt]=R[rs]+SigExtImm	(2) 9_{16}
Add Unsigned	addu	R	R[rd]=R[rs]+R[rt]	$0/21_{16}$
And	and	R	R[rd]=R[rs]&R[rt]	(1) $0/24_{16}$
And Immediate	andi	I	R[rt]=R[rs]&ZeroExtImm	(3) C_{16}
Branch On Equal	beq	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4) 4_{16}
Branch On Not Equal	bne	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4) 5_{16}
Jump	j	J	PC=JumpAddr	(5) 2_{16}
Jump And Link	jal	J	R[31]=PC+8; PC=JumpAddr	(5) 3_{16}
Jump Register	jr	R	PC=R[rs]	$0/08_{16}$
Load Byte Unsigned	lbu	I	R[rt]={24'b0,M[R[rs]] +SignExtImm](7:0)}	(2) 24_{16}

Name	Mnemonic	Format	Operation (in Verilog)	OPCODE/Funct (Hex)
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs]] +SignExtImm](15:0)}	(2) 25_{16}
Load Linked	ll	I	R[rt]=M[R[rs]] +SignExtImm]	$(2, 7)$ 30_{16}
Load Upper Imm.	lui	I	R[rt]={imm, 16'b0}	F_{16}
Load Word	lw	I	R[rt]=M[R[rs] +SignExtImm]	(2) 23_{16}
Nor	nor	R	R[rd]=$\sim$ ($R[rs]$ | $R[rt]$)	$0/27_{16}$
Or	or	R	R[rd]=(R[rs] | R[rt])	$0/25_{16}$
Or Immediate	ori	I	R[rt]=R[rs] | ZeroExtImm	(3) D_{16}
Set Less Than	slt	R	R[rd]=(R[rs]<R[rt])? 1 : 0	$0/2A_{16}$
Set Less Than Imm.	slti	I	R[rt]=(R[rs]<SignExtImm) ? 1 : 0	(2) A_{16}
Set Less Than Imm. Unsigned	sltiu	I	R[rt]=(R[rs]<SignExtImm) ? 1 : 0	$(2, 6)$ B_{16}
Set Less Than Unsigned	sltu	R	R[rd]=(R[rs]<R[rt]) ? 1 : 0	(6) $0/2B_{16}$
Shift Left Logical	sll	R	R[rd]=R[rt]<<shamt	$0/00_{16}$
Shift Right Logical	srl	R	R[rd]=R[rt]>>>shamt	$0/02_{16}$

Name	Mnemonic	Format	Operation (in Verilog)	OPCODE/Funct (Hex)
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) =R[rt](7:0)	(2) 28_{16}
Store Conditional	sc	I	M[R[rs]+SignExtImm]=R[rt]; R[rt]=(*atomic*) ? 1 : 0	$(2, 7)$ 38_{16}
Store Halfword	sh	I	M[R[rs]+SignExtImm](15:0) =R[rt](15:0)	(2) 29_{16}
Store Word	sw	I	M[R[rs]+SignExtImm]=R[rt]	(2) $2B_{16}$
Subtract	sub	R	R[rd]=R[rs]-R[rt]	(1) $0/22_{16}$
Subtract Unsigned	subu	R	R[rd]=R[rs]-R[rt]	$0/23_{16}$

Basic Instruction Formats

Arithmetic Core Instruction Set

Name	Mnemonic	Format	Operation	OPCODE/FMT/FT/Funct (Hex)
Branch On FP True	bolt	FI	if(FPcond)PC=PC+4 +BranchAddr	(4) 11/8/1/ − −
Branch On FP False	bolf	FI	if(!FPcond)PC=PC+4 +BranchAddr	(4) 11/8/0/ − −
Divide	div	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	0/ − −/ − −/la
Divide Unsigned	divu	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6) 0/ − −/ − −/la

Name	`Mnemonic`	Format	Operation	OPCODE/FMT/FT/Funct (Hex)
FP Add Single	`add.s`	FR	F[fd]=F[fs]+F[ft]	$11/10/--/0$
FP Add Double	`add.d`	FR	{F[fd], F[fd+1]} = {F[fs]+F[fs+1]} + {F[ft]+F[ft+1]}	$11/11/--/0$
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	$11/11/--/y$

Floating-Point Instruction Formats

Pseudoinstruction Set

Register Name, Number, Use, Call Convention

Opcodes, Base Conversion, ASCII Symbols

IEEE 754 Floating-Point Standard

IEEE Single and Double Precision Formats

Memory Allocation

Stack Frame

Data Alignment

Exception Control Registers: Cause and Status

Exception Codes

Size Prefixes

*(x is `eq`, `lt`, or `le`) (*op* is ==, <, or ≤) (y is 32, 3C, or 3E)

(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+4[31:28], 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 pair not atomic