MIPS Instruction Coding

Instruction Coding Formats

MIPS instructions are classified into four groups according to their coding formats:

- R-Type - This group contains all instructions that do not require an immediate value, target offset, memory address displacement, or memory address to specify an operand. This includes arithmetic and logic with all operands in registers, shift instructions, and register direct jump instructions (`jalr` and `jr`).

  All R-type instructions use opcode 000000.

- I-Type - This group includes instructions with an immediate operand, branch instructions, and load and store instructions. In the MIPS architecture, all memory accesses are handled by the main processor, so coprocessor load and store instructions are included in this group.

  All opcodes except 000000, 00001x, and 0100xx are used for I-type instructions.

- J-Type - This group consists of the two direct jump instructions (`j` and `jal`). These instructions require a memory address to specify their operand.

  J-type instructions use opcodes 00001x.

- Coprocessor Instructions - MIPS processors all have two standard coprocessors, CP0 and CP1. CP0 processes various kinds of program exceptions. CP1 is a floating point processor. The MIPS architecture makes allowance for future inclusion of two additional coprocessors, CP2 and CP3. All coprocessor instructions use opcodes 0100xx.

  Note: ALL arithmetic immediate values are sign-extended. After that, they are handled as signed or unsigned 32-bit numbers, depending upon the instruction. Signed instructions can generate an overflow exception; unsigned cannot.

R-Type Instructions (Opcode 000000)

Main processor instructions that do not require a target address, immediate value, or branch displacement use an R-type

coding format. This format has fields for specifying of up to three registers and a shift amount. For instructions that do not use all of these fields, the unused fields are coded with all 0 bits. All R-type instructions use a 000000 opcode. The operation is specified by the function field.

opcode (6)	rs (5)	rt (5)	rd (5)	sa (5)	function (6)

Alf	Instruction	Function	Opcd	Funct	Description	Numeric	Instruction	Function	Funct Hex
add	rd, rs, rt	100000	0x00	0x20	Add (with overflow)	sll	rd, rt, sa	000000	0x00
addu	rd, rs, rt	100001	0x00	0x21	Add unsigned (no overflow)	srl	rd, rt, sa	000010	0x02
and	rd, rs, rt	100100	0x00	0x24	Bitwise and	sra	rd, rt, sa	000011	0x03
break		001101	0x00	0x0D	Break (for debugging)	sllv	rd, rt, rs	000100	0x04
div	rs, rt	011010	0x00	0x1A	Divide	srlv	rd, rt, rs	000110	0x06
divu	rs, rt	011011	0x00	0x1B	Divide unsigned	srav	rd, rt, rs	000111	0x07
jalr	rd, rs	001001	0x00	0x09	Jump and link	jr	rs	001000	0x08
jr	rs	001000	0x00	0x08	Jump register	jalr	rd, rs	001001	0x09
mfhi	rd	010000	0x00	0x10	Move from HI	syscall		001100	0x0C
mflo	rd	010010	0x00	0x12	Move from LO	break		001101	0x0D
mthi	rs	010001	0x00	0x11	Move to HI	mfhi	rd	010000	0x10
mtlo	rs	010011	0x00	0x13	Move to LO	mthi	rs	010001	0x11
mult	rs, rt	011000	0x00	0x18	Multiply	mflo	rd	010010	0x12
multu	rs, rt	011001	0x00	0x19	Multiply unsigned	mtlo	rs	010011	0x13
nor	rd, rs, rt	100111	0x00	0x27	Bitwise nor	mult	rs, rt	011000	0x18
or	rd, rs, rt	100101	0x00	0x25	Bitwise or	multu	rs, rt	011001	0x19
sll	rd, rt, sa	000000	0x00	0x00	Shift left logical	div	rs, rt	011010	0x1A
sllv	rd, rt, rs	000100	0x00	0x04	Shift left logical variable	divu	rs, rt	011011	0x1B
slt	rd, rs, rt	101010	0x00	0x2A	Set on less than (signed)	add	rd, rs, rt	100000	0x20
sltu	rd, rs, rt	101011	0x00	0x2B	Set on less than unsigned	addu	rd, rs, rt	100001	0x21
sra	rd, rt, sa	000011	0x00	0x03	Shift right arithmetic	sub	rd, rs, rt	100010	0x22
srav	rd, rt, rs	000111	0x00	0x07	Shift right arithmetic variable	subu	rd, rs, rt	100011	0x23
srl	rd, rt, sa	000010	0x00	0x02	Shift right logical	and	rd, rs, rt.	100100	0x24
srlv	rd, rt, rs	000110	0x00	0x06	Shift right logical variable	or	rd, rs, rt	100101	0x25
sub	rd, rs, rt	100010	0x00	0x22	Subtract	xor	rd, rs, rt	100110	0x26
subu	rd, rs, rt	100011	0x00	0x23	Subtract unsigned	nor	rd, rs, rt	100111	0x27
syscall		001100	0x00	0x0C	System call	slt	rd, rs, rt	101010	0x2A
xor	rd, rs, rt	100110	0x00	0x26	Bitwise exclusive or	sltu	rd, rs, rt	101011	0x2B

I-Type Instructions (All opcodes except 000000, 00001x, and 0100xx)

I-type instructions have a 16-bit immediate field that codes an immediate operand, a branch target offset, or a displacement for a memory operand. For a branch target offset, the immediate field contains the signed difference between the address of the following instruction and the target label, with the two low order bits dropped. The dropped bits are always 0 since instructions are word-aligned.

For the `bgez`, `bgtz`, `blez`, and `bltz` instructions, the rt field is used as an extension of the opcode field.

opcode (6)	rs (5)	rt (5)	immediate (16)

Alf Instruction	Opcode	Notes	Opcd	Description	Numeric Instruction	Opcode	Notes	Opcd
addi rt, rs, immediate	001000		0x08	Add immediate (with overflow)	bltz rs, label	000001	rt=00000	0x01
addiu rt, rs, immediate	001001		0x09	Add immediate unsigned (no overflow)	bgez rs, label	000001	rt=00001	0x01
andi rt, rs, immediate	001100		0x0C	Bitwise and immediate	beq rs, rt, label	000100		0x04
beq rs, rt, label	000100		0x04	Branch on equal	bne rs, rt, label	000101		0x05
bgez rs, label	000001	rt=00001	0x01	Branch on greater than or equal to zero	blez rs, label	000110	rt=00000	0x06
bgtz rs, label	000111	rt=00000	0x07	Branch on greater than zero	bgtz rs, label	000111	rt=00000	0x07
blez rs, label	000110	rt=00000	0x06	Branch on less than or equal to zero	addi rt, rs, immediate	001000		0x08
bltz rs, label	000001	rt=00000	0x01	Branch on less than zero	addiu rt, rs, immediate	001001		0x09
bne rs, rt, label	000101		0x05	Branch on not equal	slti rt, rs, immediate	001010		0x0A
lb rt, immediate(rs)	100000		0x20	Load byte	sltiu rt, rs, immediate	001011		0x0B
lbu rt, immediate(rs)	100100		0x24	Load byte unsigned	andi rt, rs, immediate	001100		0x0C
lh rt, immediate(rs)	100001		0x21	Load halfword	ori rt, rs, immediate	001101		0x0D
lhu rt, immediate(rs)	100101		0x25	Load halfword unsigned	xori rt, rs, immediate	001110		0x0E
lui rt, immediate	001111		0x0F	Load upper immediate	lui rt, immediate	001111		0x0F
lw rt, immediate(rs)	100011		0x23	Load word	lb rt, immediate(rs)	100000		0x20
lwc1 rt, immediate(rs)	110001		0x31	Load word coprocessor 1	lh rt, immediate(rs)	100001		0x21
ori rt, rs, immediate	001101		0x0D	Bitwise or immediate	lw rt, immediate(rs)	100011		0x23
sb rt, immediate(rs)	101000		0x28	Store byte	lbu rt, immediate(rs)	100100		0x24
slti rt, rs, immediate	001010		0x09	Set on less than immediate (signed)	lhu rt, immediate(rs)	100101		0x25
sltiu rt, rs, immediate	001011		0x0B	Set on less than immediate unsigned	sb rt, immediate(rs)	101000		0x28
sh rt, immediate(rs)	101001		0x29	Store halfword	sh rt, immediate(rs)	101001		0x29
sw rt, immediate(rs)	101011		0x2B	Store word	sw rt, immediate(rs)	101011		0x2B
swc1 rt, immediate(rs)	111001		0x39	Store word coprocessor 1	lwc1 rt, immediate(rs)	110001		0x31
xori rt, rs, immediate	001110		0x0E	Bitwise exclusive or immediate	swc1 rt, immediate(rs)	111001		0x39

J-Type Instructions (Opcode 00001x)

The only J-type instructions are the jump instructions `j` and `jal`. These intructions require a 26-bit coded address field to specify the target of the jump. The coded address is formed from the bits at positions 27 to 2 in the binary representation of the address. The bits at positions 1 and 0 are always 0 since instructions are word-aligned.

When a J-type instruction is executed, a full 32-bit jump target address is formed by concatenating the high order four bits of the PC (the address of the instruction following the jump), the 26 bits of the target field, and two 0 bits.

opcode (6)	target (26)

Instruction		Opcode	Target	Opcd	Description
j	label	000010	coded address of label	0x02	Jump
jal	label	000011	coded address of label	0x03	Jump and link

Coprocessor Instructions (Opcode 0100xx)

The only instructions that are described here are the floating point instructions that are common to all processors in the MIPS family. All coprocessor instructions instructions use opcode 0100xx. The last two bits specify the coprocessor number. Thus all floating point instructions use opcode 010001.

The instruction is broken up into fields of the same sizes as in the R-type instruction format. However, the fields are used in different ways.

Most floating point intructions use the format field to specify a numerical coding format: single precision (.s), double precision (.d), or fixed point (.w). The `mfc1` and `mtc1` instructions uses the format field as an extension of the function field; one operand specifies a coprocessor floating point register (fx) and the other, a MIPS general purpose register (rx).

opcode (6)	format (5)	ft (5)	fs (5)	fd (5)	function (6)

Alf Instruction		Opcode	Format	Funct	Opc	Form	Func	Description	Numeric Instruction Function	Funct Hex
add.s	fd, fs, ft	010001	10000	000000	0x11	0x10	0x00	FP add single		
cvt.s.w	fd, fs	010001	10100	100000	0x11	0x14	0x20	Convert to single precision FP from integer		
cvt.w.s	fd, fs	010001	10000	100100	0x11	0x10	0x24	Convert to integer from single precision FP		
div.s	fd, fs, ft	010001	10000	000011	0x11	0x10	0x03	FP divide single		
mfc1	rd, fs	010001	00000	000000	0x11	0x00	0x00	move from coprocessor 1 (FP)		
mov.s	fd, fs	010001	10000	000110	0x11	0x10	0x06	move FP single precision FP		
mtc1	rs, fd	010001	00100	000000	0x11	0x04	0x00	move to coprocessor 1 (FP)		
mul.s	fd, fs, ft	010001	10000	000010	0x11	0x10	0x02	FP multiply single		
sub.s	fd, fs, ft	010001	10000	000001	0x11	0x10	0x01	FP subtract single		

Page URL: http://www.cs.sunysb.edu/~lw/spim/MIPSinstHex.pdf from http://www.d.umn.edu/~gshute/spimsal/talref.html
Page Author: extensions by Larry Wittie from original no-hex instruction lists by Gary Shute
Last Modified: Saturday, 18-Sep-2010 from original of Tuesday, 26-Jun-2007 12:33:40 CDT
Comments to: lw AT ic DOT sunysb DOT edu (Original instruction list author was gshute@d.umn.edu)