

MIPS Instruction Types

Type	format (bits)					
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

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)
------------	--------	--------	--------	--------	--------------

Instruction	Function
add	rd, rs, rt 100000
addu	rd, rs, rt 100001
and	rd, rs, rt 100100
break	001101
div	rs, rt 011010
divu	rs, rt 011011
jalr	rd, rs 001001
jr	rs 001000
mfhi	rd 010000
mflo	rd 010010
mthi	rs 010001
mtlo	rs 010011
mult	rs, rt 011000
multu	rs, rt 011001
nor	rd, rs, rt 100111
or	rd, rs, rt 100101
sll	rd, rt, sa 000000
sllv	rd, rt, rs 000100
slt	rd, rs, rt 101010
sltu	rd, rs, rt 101011
sra	rd, rt, sa 000011
srav	rd, rt, rs 000111
srl	rd, rt, sa 000010
srlv	rd, rt, rs 000110
sub	rd, rs, rt 100010
subu	rd, rs, rt 100011
syscall	001100
xor	rd, rs, rt 100110

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)
------------	--------	--------	----------------

Instruction	Opcode	Notes
addi	rt, rs, immediate	001000
addiu	rt, rs, immediate	001001
andi	rt, rs, immediate	001100
beq	rs, rt, label	000100
bgez	rs, label	000001 rt = 00001
bgtz	rs, label	000111 rt = 00000
blez	rs, label	000110 rt = 00000
bltz	rs, label	000001 rt = 00000
bne	rs, rt, label	000101
lb	rt, immediate(rs)	100000
lbu	rt, immediate(rs)	100100
lh	rt, immediate(rs)	100001
lhu	rt, immediate(rs)	100101
lui	rt, immediate	001111
lw	rt, immediate(rs)	100011
lwcl	rt, immediate(rs)	110001
ori	rt, rs, immediate	001101
sb	rt, immediate(rs)	101000
slti	rt, rs, immediate	001010
sltiu	rt, rs, immediate	001011
sh	rt, immediate(rs)	101001
sw	rt, immediate(rs)	101011
swcl	rt, immediate(rs)	111001
xori	rt, rs, immediate	001110

J-Type Instructions (Opcode 00001x)

The only J-type instructions are the jump instructions `j` and `jal`. These instructions 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
<code>j</code> label	000010	coded address of label
<code>jal</code> label	000011	coded address of label

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 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 instructions 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 use the format field as an extension of the function field.

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

Instruction	Function	Format
<code>add.s</code>	fd, fs, ft	000000 10000
<code>cvt.s.w</code>	fd, fs, ft	100000 10100
<code>cvt.w.s</code>	fd, fs, ft	100100 10000
<code>div.s</code>	fd, fs, ft	000011 10000
<code>mfc1</code>	ft, fs	000000 00000
<code>mov.s</code>	fd, fs	000110 10000
<code>mtc1</code>	ft, fs	000000 00100
<code>mul.s</code>	fd, fs, ft	000010 10000
<code>sub.s</code>	fd, fs, ft	000001 10000

OPCODE map

Table of opcodes for all instructions:

	000	001	010	011	100	101	110	111
000	R-type		<code>j</code>	<code>jal</code>	<code>beq</code>	<code>bne</code>	<code>blez</code>	<code>bgtz</code>
001	<code>addi</code>	<code>addiu</code>	<code>slti</code>	<code>sltiu</code>	<code>andi</code>	<code>ori</code>	<code>xori</code>	
010								
011	<code>llo</code>	<code>lhi</code>	<code>trap</code>					
100	<code>lb</code>	<code>lh</code>		<code>lw</code>	<code>lbu</code>	<code>lhu</code>		
101	<code>sb</code>	<code>sh</code>		<code>sw</code>				
110								
111								

FUNC map of R-type instructions

Table of function codes for register-format instructions:

	000	001	010	011	100	101	110	111
000	<code>sll</code>		<code>srl</code>	<code>sra</code>	<code>sllv</code>		<code>srlv</code>	<code>srav</code>
001	<code>jr</code>	<code>jalr</code>						
010	<code>mfhi</code>	<code>mthi</code>	<code>mflo</code>	<code>mtlo</code>				
011	<code>mult</code>	<code>multu</code>	<code>div</code>	<code>divu</code>				
100	<code>add</code>	<code>addu</code>	<code>sub</code>	<code>subu</code>	<code>and</code>	<code>or</code>	<code>xor</code>	<code>nor</code>
101			<code>slt</code>	<code>sltu</code>				
110								
111								

Some R-type instructions grouped by their familiarity:

Instruction	Function
<code>add</code>	rd, rs, rt 100000
<code>addu</code>	rd, rs, rt 100001
<code>sub</code>	rd, rs, rt 100010
<code>subu</code>	rd, rs, rt 100011
<code>and</code>	rd, rs, rt 100100
<code>or</code>	rd, rs, rt 100101
<code>xor</code>	rd, rs, rt 100110
<code>nor</code>	rd, rs, rt 100111
<code>slt</code>	rd, rs, rt 101010
<code>sltu</code>	rd, rs, rt 101011
<code>sll</code>	rd, rt, sa 000000
<code>srl</code>	rd, rt, sa 000010
<code>sllv</code>	rd, rt, rs 000100
<code>srlv</code>	rd, rt, rs 000110
<code>sra</code>	rd, rt, sa 000011
<code>srav</code>	rd, rt, rs 000111