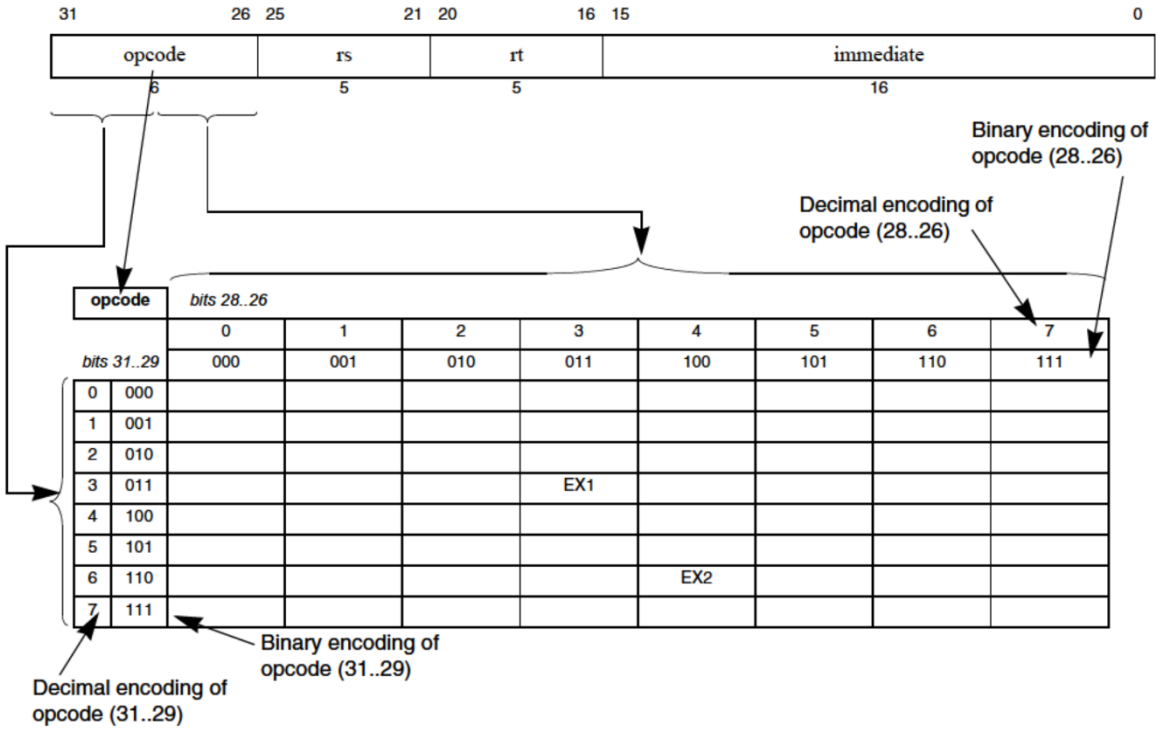
MIPS Instruction Reference

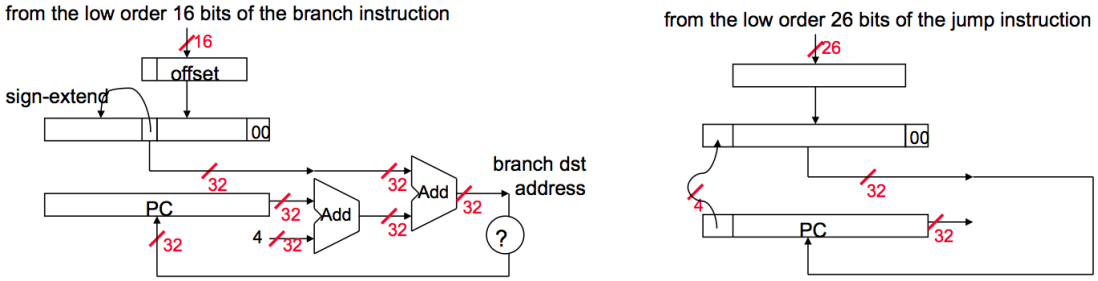
MIPS Instruction Set Summary

R-Type:	**op**	**$rs**	**$rt**	**$rd**	**shamt**	**func**
I-Type:	**op**	**$rs**	**$rt**	**imm16**		
J-Type:	**op**	**imm26**				

Sample Bit Encoding Table



Address Calculation for Branch and Jump



On the MIPS architecture, jump and branch instructions have a "delay slot". However, it is not implemented. Instead, the branch-not-taken prediction is implemented. Coprocessor support and Trap support are not implemented either.

OPCODE Map,

Table of opcodes for all instructions:

H / L	000	001	010	011	100	101	110	111
000	R-type	RegIm	j	jal	beq	bne	blez	bgtz
001	addi	addiu	slti	sltiu	andi	ori	xori	lui
010	COP0	COP1	COP2	COP3	beql	bnel	blezl	bgtzl
011	(llo)*	(lhi)*	(user)*	(user)*	(user)*	(user)*	(user)*	(user)*
100	lb	lh	lwl	lw	lbu	lhu	lwr	
101	sb	sh	swl	sw			swr	cache
110	ll	lwc1	lwc2	pref		ldc1	ldc2	
111	sc	swc1	swc2			sdc1	sdc2	

RegImm (I-Type) with $rt:

H / L	000	001	010	011	100	101	110	111
00	bltz	bgez	bltzl	bgezl				
01	tgei	tgeiu	tlti	tltiu	teqi		tnei	
10	bltzal	bgezal	bltzall	bgezall				
11								

FUNC Map of R-type instructions,

Table of function codes for register-format instructions:

H / L	000	001	010	011	100	101	110	111
000	sll		srl	sra	sllv		srlv	srav
001	jr	jalr	movz	movn	syscall	break		sync
010	mfhi	mthi	mflo	mtlo				
011	mult	multu	div	divu				
100	add	addu	sub	subu	and	or	xor	nor
101			slt	sltu				
110	tge	tgeu	tlt	tltu	teq		tne	
111								

Abbr:

o: op $s: $rs $t: $rt $d: $rd sa: shamt f: func i: imm16 i26: imm26

* Instructions in the shaded cells are not implemented.

Arithmetic and Logical Instructions

Instruction	Opcode/Function	Syntax	Operation
add	100000	f $d, $s, $t	$d = $s + $t
addu	100001	f $d, $s, $t	$d = $s + $t
addi	001000	o $t, $s, i	$t = $s + SE(i)
addiu	001001	o $t, $s, i	$t = $s + SE(i)
and	100100	f $d, $s, $t	$d = $s & $t
andi	001100	o $t, $s, i	$t = $s & ZE(i)
div	011010	f $s, $t	lo = $s / $t; hi = $s % $t
divu	011011	f $s, $t	lo = $s / $t; hi = $s % $t
mult	011000	f $s, $t	hi:lo = $s * $t
multu	011001	f $s, $t	hi:lo = $s * $t
nor	100111	f $d, $s, $t	$d = ~($s | $t)
or	100101	f $d, $s, $t	$d = $s | $t
ori	001101	o $t, $s, i	$t = $s | ZE(i)
sll	000000	f $d, $t, sa	$d = $t << a
sllv	000100	f $d, $t, $s	$d = $t << $s
sra	000011	f $d, $t, sa	$d = $t >>> a
srav	000111	f $d, $t, $s	$d = $t >>> $s
srl	000010	f $d, $t, sa	$d = $t >> a
srlv	000110	f $d, $t, $s	$d = $t >> $s
sub	100010	f $d, $s, $t	$d = $s - $t
subu	100011	f $d, $s, $t	$d = $s - $t
xor	100110	f $d, $s, $t	$d = $s ^ $t
xori	001110	o $t, $s, i	$t = $s ^ ZE(i)

Comparison Instructions

Instruction	Opcode/Function	Syntax	Operation
slt	101010	f $d, $s, $t	$d = ($s < $t)
sltu	101011	f $d, $s, $t	$d = ($s < $t)
slti	001010	o $d, $s, i	$t = ($s < SE(i))
sltiu	001011	o $d, $s, i	$t = ($s < ZE(i))

Branch Instructions

Instruction	Opcode/Function	Syntax	Operation
beq	000100	o $s, $t, label	if ($s == $t) pc += i << 2
bgtz	000111	o $s, label	if ($s > 0) pc += i << 2
blez	000110	o $s, label	if ($s <= 0) pc += i << 2
bne	000101	o $s, $t, label	if ($s != $t) pc += i << 2
bltz	(00000)	r $s, label	if ($s < 0) pc += i << 2
bgez	(00001)	r $s, label	if ($s >= 0) pc += i << 2

Jump Instructions

Instruction	Opcode/Function	Syntax	Operation
j	000010	o label	pc = pc4 | i26 << 2
jal	000011	o label	$31 = pc; pc = pc4 | i26 << 2
jalr	001001	f labelR	$31 = pc; pc = $s
jr	001000	f labelR	pc = $s

Store Instructions

Instruction	Opcode/Function	Syntax	Operation
sb	101000	o $t, i ($s)	MEM [$s + i]:1 = LB ($t)
sh	101001	o $t, i ($s)	MEM [$s + i]:2 = LH ($t)
sw	101011	o $t, i ($s)	MEM [$s + i]:4 = $t

Load Instructions

Instruction	Opcode/Function	Syntax	Operation
lb	100000	o $t, i ($s)	$t = SE (MEM [$s + i]:1)
lbu	100100	o $t, i ($s)	$t = ZE (MEM [$s + i]:1)
lh	100001	o $t, i ($s)	$t = SE (MEM [$s + i]:2)
lhu	100101	o $t, i ($s)	$t = ZE (MEM [$s + i]:2)
lw	100011	o $t, i ($s)	$t = MEM [$s + i]:4

Data Movement Instructions

Instruction	Opcode/Function	Syntax	Operation
mfhi	010000	f $d	$d = hi
mflo	010010	f $d	$d = lo
mthi	010001	f $s	hi = $s
mtlo	010011	f $s	lo = $s

Constant-Manipulating Instructions

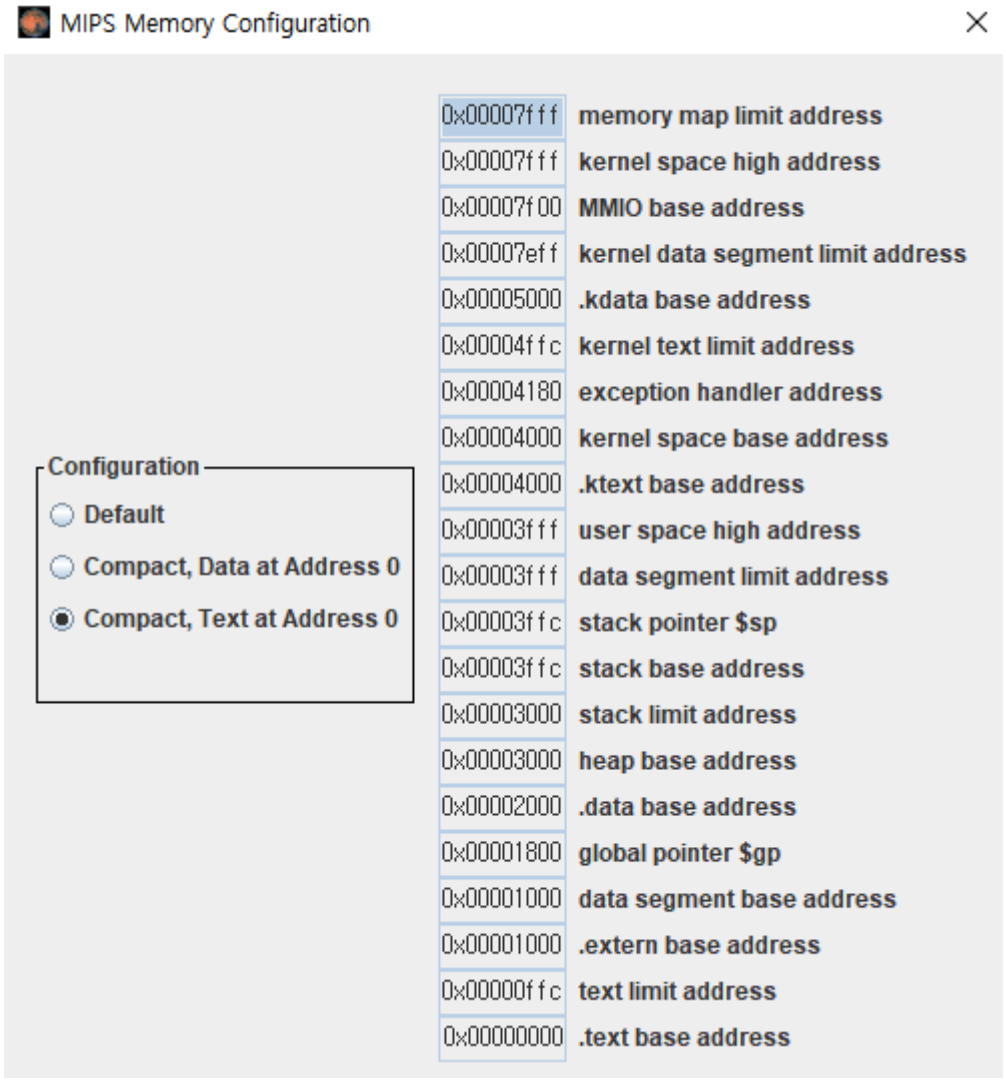
Instruction	Opcode/Function	Syntax	Operation
lui	001111	o $t, i	$t = i << 16

Exception and Interrupt Instructions

Instruction	Opcode/Function	Syntax	Operation
syscall	001100	f (i << 6)	Dependent on OS;
break	001101	f	Dependent on OS; Just break signal in this implementation.

User defined Instructions (Test purpose)

Instruction	Opcode/Function	Syntax	Operation
lhi	011001	o $t, i	HH ($t) = i
llo	011000	o $t, i	LH ($t) = i


MIPS Memory Configuration
Configuration
Default
Compact, Data at Address 0
Compact, Text at Address 0
0x00007fff memory map limit address
0x00007fff kernel space high address
0x00007f00 MMIO base address
0x00007eff kernel data segment limit address
0x00005000 .kdata base address
0x00004ffc kernel text limit address
0x00004180 exception handler address
0x00004000 kernel space base address
0x00004000 .ktext base address
0x00003fff user space high address
0x00003fff data segment limit address
0x00003ffc stack pointer $sp
0x00003ffc stack base address
0x00003000 stack limit address
0x00003000 heap base address
0x00002000 .data base address
0x00001800 global pointer $gp
0x00001000 data segment base address
0x00001000 .extern base address
0x00000ffc text limit address
0x00000000 .text base address