MIPS Reference

Machine Encoding Aids

Key

o/f	instruction/function opcodes
s/t/d	first/second/third register
a/i	shift amount/immediate

Instruction Encoding Formats

Register	000000ss sssttttt dddddaaa aaffffff
Immediate	ooooooss sssttttt iiiiiiii iiiiiiii
Jump	ooooooii iiiiiiii iiiiiiii iiiiiiii

Instruction Syntax

Encoding	Syntax	Template
Register	ArithLog	f $d, $s, $t
	DivMult	f $s, $t
	Shift	f $d, $t, a
	ShiftV	f $d, $t, $s
	JumpR	f $s
	MoveFrom	f $d
	MoveTo	f $s
Immediate	ArithLogI	o $t, $s, i
	LoadI	o $t, immed32
	Branch	o $s, $t, label
	BranchZ	o $s, label
	LoadStore	o $t, i($s)
Jump	Jump	o label
	Trap	o i

Instruction Reference

Arithmetic and Logical Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
add $d, $s, $t	$d = $s + $t	100000	ArithLog	
addu $d, $s, $t	$d = $s + $t	100001	ArithLog	
addi $t, $s, i	$t = $s + i	001000	ArithLogI	i is sign-extended
addiu $t, $s, i	$t = $s + i	001001	ArithLogI	i is sign-extended
and $d, $s, $t	$d = $s & $t	100100	ArithLog	
andi $t, $s, i	$t = $s & i	001100	ArithLogI	i is zero-extended
div $s, $t	lo = $s / $t; hi = $s % $t	011010	DivMult	
divu $s, $t	lo = $s / $t; hi = $s % $t	011011	DivMult	
mult $s, $t	hi:lo = $s * $t	011000	DivMult	
multu $s, $t	hi:lo = $s * $t	011001	DivMult	
nor $d, $s, $t	$d = ~($s | $t)	100111	ArithLog	
or $d, $s, $t	$d = $s | $t	100101	ArithLog	
ori $t, $s, i	$t = $s | i	001101	ArithLogI	i is zero-extended
sll $d, $t, a	$d = $t << a	000000	Shift	Zero is shifted in
sllv $d, $t, $s	$d = $t << $s	000100	ShiftV	Zero is shifted in
sra $d, $t, a	$d = $t >> a	000011	Shift	Sign bit is shifted in
srav $d, $t, $s	$d = $t >> $s	000111	ShiftV	Sign bit is shifted in
srl $d, $t, a	$d = $t >> a	000010	Shift	Zero is shifted in
srlv $d, $t, $s	$d = $t >> $s	000110	ShiftV	Zero is shifted in
sub $d, $s, $t	$d = $s - $t	100010	ArithLog	
subu $d, $s, $t	$d = $s - $t	100011	ArithLog	
xor $d, $s, $t	$d = $s ^ $t	100110	ArithLog	
xori $d, $s, i	$d = $s ^ i	001110	ArithLogI	i is zero-extended

Movement Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lhi $t, i	$t = i << 16	011001	LoadI	i is zero-extended
llo $t, i	$t = i	011000	LoadI	i is zero-extended
mfhi $d	$d = hi	010000	MoveFrom	
mflo $d	$d = lo	010010	MoveFrom	
mthi $s	hi = $s	010001	MoveTo	
mtlo $s	lo = $s	010011	MoveTo	

Comparison Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
slt $d, $s, $t	$d = $s < $t	101010	ArithLog	
sltu $d, $s, $t	$d = $s < $t	101001	ArithLog	
slti $t, $s, i	$d = $s < i	001010	ArithLogI	i is sign-extended
sltiu $t, $s, i	$d = $s < i	001001	ArithLogI	i is sign-extended

Branch and Jump Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
beq $s, $t, label	if ($s == $t) pc += i << 2	000100	Branch	label is a line reference in the code
bgtz $s, label	if ($s > 0) pc += i << 2	000111	BranchZ	label is a line reference in the code
blez $s, label	if ($s <= 0) pc += i << 2	000110	BranchZ	label is a line reference in the code
bne $s, $t, label	if ($s != $t) pc += i << 2	000101	Branch	label is a line reference in the code
j label	pc += i << 2	000010	Jump	label is a line reference in the code
jal label	$ra = pc; pc += i << 2	000011	Jump	label is a line reference in the code
jalr $s	$ra = pc; pc = $s	001001	JumpR	
jr $s	pc = $s	001000	JumpR	

Memory Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lb $t, i($s)	$t = MEM[$s + i]	100000	LoadStore	Sign-extends the loaded byte
lbu $t, i($s)	$t = MEM[$s + i]	100100	LoadStore	Zero-extends the loaded byte
lh $t, i($s)	$t = MEM[$s + i]	100001	LoadStore	Sign-extends the loaded bytes
lhu $t, i($s)	$t = MEM[$s + i]	100101	LoadStore	Zero-extends the loaded bytes
lw $t, i($s)	$t = MEM[$s + i]	100011	LoadStore	
sb $t, i($s)	MEM[$s + i] = $t	101000	LoadStore	Lowest order byte is stored
sh $t, i($s)	MEM[$s + i] = $t	101001	LoadStore	2 lowest order bytes are stored
sw $t, i($s)	MEM[$s + i] = $t	101011	LoadStore	

Exception and Interrupt Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
trap i	Exception	0011010	Trap	i is a trap code; implements syscall