MIPS Reference Sheet

Arithmetic and Logical Instructions

Instruction	Operation
add $d, $s, $t	`$d = $s + $t`
addu $d, $s, $t	`$d = $s + $t`
addi $t, $s, i	`$t = $s + SE(i)`
addiu $t, $s, i	`$t = $s + SE(i)`
and $d, $s, $t	`$d = $s & $t`
andi $t, $s, i	`$t = $s & ZE(i)`
div $s, $t	`lo = $s / $t; hi = $s % $t`
divu $s, $t	`lo = $s / $t; hi = $s % $t`
mult $s, $t	`hi:lo = $s * $t`
multu $s, $t	`hi:lo = $s * $t`
nor $d, $s, $t	`$d = ~($s | $t)`
or $d, $s, $t	`$d = $s | $t`
ori $t, $s, i	`$t = $s | ZE(i)`
sll $d, $t, a	`$d = $t << a`
sllv $d, $t, $s	`$d = $t << $s`
sra $d, $t, a	`$d = $t >> a`
srav $d, $t, $s	`$d = $t >> $s`
srl $d, $t, a	`$d = $t >>> a`
srlv $d, $t, $s	`$d = $t >>> $s`
sub $d, $s, $t	`$d = $s - $t`
subu $d, $s, $t	`$d = $s - $t`
xor $d, $s, $t	`$d = $s ^ $t`
xori $d, $s, i	`$d = $s ^ ZE(i)`

Constant-Manipulating Instructions

Instruction	Operation
lhi $t, i	`HH($t) = i`
llo $t, i	`LH($t) = i`

Comparison Instructions

Instruction	Operation
slt $d, $s, $t	`$d = ($s < $t)`
sltu $d, $s, $t	`$d = ($s < $t)`
slti $t, $s, i	`$t = ($s < SE(i))`
sltiu $t, $s, i	`$t = ($s < SE(i))`

Branch Instructions

Instruction	Operation
beq $s, $t, label	`if ($s == $t) pc += i << 2`
bgtz $s, label	`if ($s > 0) pc += i << 2`
blez $s, label	`if ($s <= 0) pc += i << 2`
bne $s, $t, label	`if ($s != $t) pc += i << 2`

Jump Instructions

Instruction	Operation
j label	`pc += i << 2`
jal label	`$31 = pc; pc += i << 2`
jalr $s	`$31 = pc; pc = $s`
jr $s	`pc = $s`

Load Instructions

Instruction	Operation
lb $t, i($s)	`$t = SE (MEM [$s + i]:1)`
lbu $t, i($s)	`$t = ZE (MEM [$s + i]:1)`
lh $t, i($s)	`$t = SE (MEM [$s + i]:2)`
lhu $t, i($s)	`$t = ZE (MEM [$s + i]:2)`
lw $t, i($s)	`$t = MEM [$s + i]:4`

Store Instructions

Instruction	Operation
sb $t, i($s)	`MEM [$s + i]:1 = LB ($t)`
sh $t, i($s)	`MEM [$s + i]:2 = LH ($t)`
sw $t, i($s)	`MEM [$s + i]:4 = $t`

Data Movement Instructions

Instruction	Operation
mfhi $d	`$d = hi`
mflo $d	`$d = lo`
mthi $s	`hi = $s`
mtlo $s	`lo = $s`

Exception and Interrupt Instructions

Instruction	Operation
trap 1	Print integer value in `$4`
trap 5	Read integer value into `$2`
trap 10	Terminate program execution
trap 101	Print ASCII character in `$4`
trap 102	Read ASCII character into `$2`

Note: Detailed encoding reference on reverse.

Instruction Encodings

Register	`000000ss sssttttt dddddaaa aaffffff`
Immediate	`oooooss sssttttt iiiiiiii iiiiiiii`
Jump	`oooooii iiiiiiii iiiiiiii iiiiiiii`

Instruction Syntax

Syntax	Template	Encoding	Comments
ArithLog	`f $d, $s, $t`	Register	
DivMult	`f $s, $t`	Register	
Shift	`f $d, $t, a`	Register	
ShiftV	`f $d, $t, $s`	Register	
JumpR	`f $s`	Register	
MoveFrom	`f $d`	Register	
MoveTo	`f $s`	Register	
ArithLogI	`o $t, $s, i`	Immediate	
LoadI	`o $t, immed32`	Immediate	`i` is high or low 16 bits of `immed32`
Branch	`o $s, $t, label`	Immediate	`i` is calculated as `(label-(current+4))>>2`
BranchZ	`o $s, label`	Immediate	`i` is calculated as `(label-(current+4))>>2`
LoadStore	`o $t, i($s)`	Immediate	
Jump	`o label`	Jump	`i` is calculated as `label<<2`
Trap	`o i`	Jump	

Opcode Table

Instruction	Opcode/Function	Syntax
add	100000	ArithLog
addu	100001	ArithLog
addi	001000	ArithLogI
addiu	001001	ArithLogI
and	100100	ArithLog
andi	001100	ArithLogI
div	011010	DivMult
divu	011011	DivMult
mult	011000	DivMult
multu	011001	DivMult
nor	100111	ArithLog
or	100101	ArithLog
ori	001101	ArithLogI
sll	000000	Shift
sllv	000100	ShiftV
sra	000011	Shift
srav	000111	ShiftV
srl	000010	Shift
srlv	000110	ShiftV
sub	100010	ArithLog
subu	100011	ArithLog
xor	100110	ArithLog
xori	001110	ArithLogI
lhi	011001	LoadI
llo	011000	LoadI

Instruction	Opcode/Function	Syntax
slt	101010	ArithLog
sltu	101001	ArithLog
slti	001010	ArithLogI
sltiu	001001	ArithLogI
beq	000100	Branch
bgtz	000111	BranchZ
blez	000110	BranchZ
bne	000101	Branch
j	000010	Jump
jal	000011	Jump
jalr	001001	JumpR
jr	001000	JumpR
lb	100000	LoadStore
lbu	100100	LoadStore
lh	100001	LoadStore
lhu	100101	LoadStore
lw	100011	LoadStore
sb	101000	LoadStore
sh	101001	LoadStore
sw	101011	LoadStore
mfhi	010000	MoveFrom
mflo	010010	MoveFrom
mthi	010001	MoveTo
mtlo	010011	MoveTo
trap	011010	Trap

Note: Operation details on reverse.