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 = high4(pc).(i<<2)`
jal label	`$31 = pc; pc = high4(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`

Assembler Directives

Directives	Operation
.word *N*	Place *N* in the next word of the assembled module.
.space *N*	Leave *N* bytes blank (0) in the assembled module.
.globl N	Label `N` should be visible from other MIPS files.

Note: Detailed encoding reference on reverse.

Instruction Encodings

Register	`000000ss ssstttt dddddaaa aaffffff`
Immediate	`oooooss ssstttt iiiiiiii iiiiiiii`
Jump	`ooooooii 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	Instruction	Opcode/Function	Syntax
add	100000	**ArithLog**	**slt**	101010	**ArithLog**
addu	100001	**ArithLog**	**sltu**	101001	**ArithLog**
addi	001000	**ArithLogI**	**slti**	001010	**ArithLogI**
addiu	001001	**ArithLogI**	**sltiu**	001011	**ArithLogI**
and	100100	**ArithLog**	**beq**	000100	**Branch**
andi	001100	**ArithLogI**	**bgtz**	000111	**BranchZ**
div	011010	**DivMult**	**blez**	000110	**BranchZ**
divu	011011	**DivMult**	**bne**	000101	**Branch**
mult	011000	**DivMult**	**j**	000010	**Jump**
multu	011001	**DivMult**	**jal**	000011	**Jump**
nor	100111	**ArithLog**	**jalr**	001001	**JumpR**
or	100101	**ArithLog**	**jr**	001000	**JumpR**
ori	001101	**ArithLogI**	**lb**	100000	**LoadStore**
sll	000000	**Shift**	**lbu**	100100	**LoadStore**
sllv	000100	**ShiftV**	**lh**	100001	**LoadStore**
sra	000011	**Shift**	**lhu**	100101	**LoadStore**
srav	000111	**ShiftV**	**lw**	100011	**LoadStore**
srl	000010	**Shift**	**sb**	101000	**LoadStore**
srlv	000110	**ShiftV**	**sh**	101001	**LoadStore**
sub	100010	**ArithLog**	**sw**	101011	**LoadStore**
subu	100011	**ArithLog**	**mfhi**	010000	**MoveFrom**
xor	100110	**ArithLog**	**mflo**	010010	**MoveFrom**
xori	001110	**ArithLogI**	**mthi**	010001	**MoveTo**
lhi	011001	**LoadI**	**mtlo**	010011	**MoveFrom**
llo	011000	**LoadI**	**trap**	011010	**Trap**

Note: Operation details on reverse.