MIPS R3000 Instruction Set Summary

MIPS Operands

Name	Example	Comments
32 registers	$0, $1, $2,..., $31	Fast location for data. In MIPS, data must be in registers to perform arithmetic. MIPS register $0 always equal 0. Register $1 is reserved for the assembler to handle pseudo instructions and large constants
2^{30} memory words	Memory[0], Memory[4],..., Memory[4293967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers, such as those saved on procedure calls

MIPS Assembler Instructions

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $1,$2,$3	$1 = $2 + $3	3 operands; exception possible
	subtract	sub $1,$2,$3	$1 = $2 - $3	3 operands; exception possible
	add immediate	addi $1,$2,100	$1 = $2 + 100	+ constant; exception possible
	add unsigned	addu $1,$2,$3	$1 = $2 + $3	3 operands; exception possible
	subtract unsigned	subi $1,$2,$3	$1 = $2 - $3	3 operands; exception possible
	add immediate unsigned	addi $1,$2,100	$1 = $2 + 100	+ constant; exception possible
	Move from coprocessor register	mfc0 $1,$epc	$1 = $epc	Used to get of Exception PC
Logical	and	and $1,$2,$3	$1 = $2 & $3	3 register operands; Logical AND
	or	or $1,$2,$3	$1 = $2 | $3	3 register operands; Logical OR
	and immediate	and $1,$2,100	$1 = $2 & 100	Logical AND register, constant
	or immediate	or $1,$2,100	$1 = $2 | 100	Logical OR register, constant
	shift left logical	sll $1,$2,10	$1 = $2 << 10	Shift left by constant

	shift right logical	srl $1,$2,10	$1 = $2 >> 10	Shift right by constant
Data transfer	load word	lw $1, (100)$2	$1 = Memory[$2+100]	Data from memory to register
	store word	sw $1, (100)$2	Memory[$2+100] = $1	Data from memory to register
	load upper immediate	lui $1,100	$1 = 100 * 2^{16}	Load constant in upper 16bits
Conditional branch	branch on equal	beq $1,$2,100	if ($1 == $2) go to PC+4+100	Equal test; PC relative branch
	branch on not equal	bne $1,$2,100	if ($1 != $2) go to PC+4+100	Not equal test; PC relative
	set on less than	slt $1,$2,$3	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; 2`s complement
	set less than immediate	slti $1,$2,100	if ($2 < 100) $1 = 1; else $1 = 0	Compare < constant; 2`s complement
	set less than unsigned	sltu $1,$2,$3	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; natural number
	set less than immediate unsigned	sltiu $1,$2,100	if ($2 < 100) $1 = 1; else $1 = 0	Compare constant; natural number
Unconditional jump	jump	j 10000	goto 10000	Jump to target address
	jump register	j $31	goto $31	For switch, procedure return
	jump and link	jal 10000	$31 = PC + 4;go to 10000	For procedure call

MIPS Floating-Point Operands

Name	Example	Comments
32 floating-point registers	$f0, $f1, $f2,..., $f31	MIPS floating point register are used in pairs for double precision numbers. Odd numbered registers cannot be used for arithemetic or branch, just for data transfer of the right "half" of double precision register pairs.
2^{30} memory words	Memory[0], Memory[4],..., Memory[4293967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers, such as those saved on procedure calls

MIPS Floating-Point Instructions

Category	Instruction	Example	Meaning	Comments
				Floating-Point add

Arithmetic	FP add single	add.s $f2,$f4,$f6	$f2 = $f4 + $f6	(single precision)
	FP subtract single	sub.s $f2,$f4,$f6	$f2 = $f4 - $f6	Floating-Point sub (single precision)
	FP multiply single	mul.s $f2,$f4,$f6	$f2 = $f4 * $f6	Floating-Point multiply (single precision)
	FP divide single	div.s $f2,$f4,$f6	$f2 = $f4 / $f6	Floating-Point divide (single precision)
	FP add double	add.d $f2,$f4,$f6	$f2 = $f4 + $f6	Floating-Point add (double precision)
	FP.dubtract double	.dub.d $f2,$f4,$f6	$f2 = $f4 - $f6	Floating-Point sub (double precision)
	FP multiply double	mul.d $f2,$f4,$f6	$f2 = $f4 * $f6	Floating-Point multiply (double precision)
	FP divide double	div.d $f2,$f4,$f6	$f2 = $f4 / $f6	Floating-Point divide (double precision)
Data transfer	load word coprocessor 1	lwc1 $f1,100($2)	$f1 = Memory[$2+100]	32-bit data to FP register
	store word coprocessor 1	swc1 $f1,100($2)	Memory[$2+100] = $f1	32-bit data to memory
Arithmetic	branch on FP true	bc1t 100	if (cond == 1) go to PC+4+100	PC relative branch if FP condition
	branch on FP false	bc1f 100	if (cond == 0) go to PC+4+100	PC relative branch if not condition
	FP compare single (eq,ne,lt,le,gt,ge)	c.lt.s $f2,$f4	if ($f2 < $f4) cond=1; else cond=0	Floating-point compare less than single precision
	FP compare double (eq,ne,lt,le,gt,ge)	c.lt.d $f2,$f4	if ($f2 < $f4) cond=1; else cond=0	Floating-point compare less than double precision