

# MIPS Not-So-Quick Reference version 1.0

This is a **rough draft**. On an exam, you'd probably be given something more condensed, like the quick reference PDF. For a complete reference, see pages 13-20 of the SPIM Documentation. These are most of the commands we will come across in class this semester. "\$reg" refers to any register.

## Registers

Name	Numbers	Usage	Comment
\$zero	\$0	Constant 0	
\$at	\$1	Reserved	
\$v0-\$v1 \$a0-\$a3	\$2-\$3 \$4-\$7	Return values Arguments 1-4	from subroutines Use for passing args to subroutines
\$t0-\$t7	\$8-\$15	Temporary	Not preserved across procedure call
\$s0-\$s7	\$16-\$23	Saved Temporary	Preserved across procedure call
\$t8-\$t9	\$24-\$25	Temporary	Not preserved across procedure call
\$k0-\$k1	\$26-\$27	Reserved	
\$gp	\$28	Global pointer	points to global area
\$sp	\$29	Stack pointer	
\$fp	\$30	Frame pointer	
\$ra	\$31	Return address	use with jal

## Math/Logic Register Instructions

Format:	operation Rdest,Rsrc1,Rsrc2	
Examples:	add \$t1,\$t4,\$t5 or \$t3,\$t4,\$t5	\$t1=\$t4+\$t5 \$t3 = \$t4 OR \$t5
Instructions:	add, sub, and, or, xor, nor	

## Math/Logic Immediate Instructions

Format:	operation Rdest,Rsrc,Imm	
Examples:	addi \$t1,\$t4,18 ori \$t3,\$t4,0x0080	\$t1=\$t4+18 \$t3 = \$t4 OR 0x0080
Instructions:	addi, subi, andi, ori, xori	

## Shift Instructions (Register)

Format:	operation Rdest,Rsrc1,Rsrc2
Examples:	sra \$t1,\$t3,\$t5 \$t1 = arithmetic right shift \$t5 bits
	sllv \$t3,\$t4,\$t5 \$t3 = \$t4 logical left shift \$t5 bits
	ror \$t3,\$t4,\$t5 \$t3 = \$t4 circular right shift \$t5 bits
Instructions:	sllv, srlv, sra, rol, ror

## Shift Instructions (Immediate)

Format:	operation Rdest,Rsrc1,Rsrc2
Examples:	sra \$t1,\$t3,8 \$t1 = \$t3 arithmetic right shift 8 bits
	sll \$t3,\$t4,2 \$t3 = \$t4 logical left shift 2 bits
	ror \$t3,\$t4,12 \$t3 = \$t4 circular right shift 12 bits
Instructions:	sll, srl, sra, rol, ror

Note: you can use this syntax with the register-level shift instructions and the assembler will figure it out from the context.

## Loading/Moving Data

move \$t1,\$t2	\$t1=\$t2	move between registers
li \$t1,1234	\$t1=1234	load immediate
la \$t5, <i>address</i>	\$t5= <i>address</i>	load address
lw \$t5, <i>address</i>	\$t5=M[ <i>address</i> ]	load word
sw \$t5, <i>address</i>	M[ <i>address</i> ]=\$t5	store word

In the above, *address* may take on 3 different forms:

<b>label</b>	address is a <b>label</b> in program
\$reg	\$reg contains address
<b>label</b> (\$reg)	address is <b>label</b> + \$reg

## Comparison Instructions (Two Registers)

Note: Src2 can be a register or immediate

Format:	operation Rsrc1,Src2,label	
Examples:	slt \$t1,\$t3,\$t5	if (\$t1 <= \$t3) \$t5=1 else \$t5=0
Examples:	sne \$t1,\$t3,burdell	if (\$t1 != \$t3) \$t5=1 else \$t5=0
Instructions:	slt, sle, sgt sge, seq, sne	

## Branch Instructions (Two Registers)

Note: Src2 can be a register or immediate

Format:	operation Rsrc1,Src2,label	
Examples:	ble \$t1,\$t3,george	if (\$t1 <= \$t3) goto label george
Examples:	bne \$t1,\$t3,burdell	if (\$t1 != \$t3) goto label burdell
Instructions:	blt, ble, bgt bge, beq, bne	

## Branch Instructions (Comp w/Zero)

Format:	operation Rsrc,label	
Examples:	blez \$t1,buzz	if (\$t1 <= 0) goto label buzz
Instructions:	blez, blz, bgtz bgez, beqz, bnez	

**Unconditional Branch:** b label

## Jump Instructions

j label	jump to label
jal label	jump to label and save address of next instruction in \$ar
jalr \$reg	jump to address in \$reg and save address of next instruction in \$ar
jr \$reg	jump to instruction whose address is in \$reg