APPENDIX A

Quick Reference

Integer Instruction Set

Name	Syntax		Space/Time
Add:	add	Rd, Rs, Rt	1/1
Add Immediate:	addi	Rt, Rs, Imm	1/1
Add Immediate Unsigned:	addiu	Rt, Rs, Imm	1/1
Add Unsigned:	addu	Rd, Rs, Rt	1/1
And:	and	Rd, Rs, Rt	1/1
And Immediate:	andi	Rt, Rs, Imm	1/1
Branch if Equal:	beq	Rs, Rt, Label	1/1
Branch if Greater Than or Equal to Zero:	bgez	Rs, Label	1/1
Branch if Greater Than or Equal to Zero and Link:	bgezal	Rs, Label	1/1
Branch if Greater Than Zero:	bgtz	Rs, Label	1/1
Branch if Less Than or Equal to Zero:	blez	Rs, Label	1/1
Branch if Less Than Zero and Link:	bltzal	Rs, Label	1/1
Branch if Less Than Zero:	bltz	Rs, Label	1/1
Branch if Not Equal:	bne	Rs, Rt, Label	1/1
Divide:	div	Rs, Rt	1/38
Divide Unsigned:	divu	Rs, Rt	1/38
Jump:	j	Label	1/1
Jump and Link:	jal	Label	1/1
Jump and Link Register:	jalr	Rd, Rs	1/1
Jump Register:	jr	Rs	1/1
Load Byte:	lb	Rt, offset(Rs)	1/1
Load Byte Unsigned:	lbu	Rt, offset(Rs)	1/1
Load Halfword:	lh	Rt, offset(Rs)	1/1
Load Halfword Unsigned:	lhu	Rt, offset(Rs)	1/1
Load Upper Immediate:	lui	Rt, Imm	1/1
Load Word:	lw	Rt, offset(Rs)	1/1
Load Word Left:	lwl	Rt, offset(Rs)	1/1
Load Word Right:	lwr	Rt, offset(Rs)	1/1
Move From High:	mfhi	Rd	1/1
Move From Low:	mflo	Rd	1/1
Move to High:	mthi	Rs	1/1
Move to Low:	mtlo	Rs	1/1
Multiply:	mult	Rs, Rt	1/32
Multiply Unsigned:	multu	Rs, Rt	1/32
NOR:	nor	Rd, Rs, Rt	1/1
OR:	or	Rd, Rs, Rt	1/1
OR Immediate:	ori	Rt, Rs, Imm	1/1

Store Byte:	sb	Rt, offset(Rs)	1/1
Store Halfword:	sh	Rt, offset(Rs)	1/1
Shift Left Logical:	sll	Rd, Rt, sa	1/1
Shift Left Logical Variable:	sllv	Rd, Rt, Rs	1/1
Set on Less Than:	slt	Rd, Rt, Rs	1/1
Set on Less Than Immediate:	slti	Rt, Rs, Imm	1/1
Set on Less Than Immediate Unsigned:	sltiu	Rt, Rs, Imm	1/1
Set on Less Than Unsigned:	sltu	Rd, Rt, Rs	1/1
Shift Right Arithmetic:	sra	Rd, Rt, sa	1/1
Shift Right Arithmetic Variable:	srav	Rd, Rt, Rs	1/1
Shift Right Logical:	srl	Rd, Rt, sa	1/1
Shift Right Logical Variable:	srlv	Rd, Rt, Rs	1/1
Subtract:	sub	Rd, Rs, Rt	1/1
Subtract Unsigned:	subu	Rd, Rs, Rt	1/1
Store Word:	sw	Rt, offset(Rs)	1/1
Store Word Left:	swl	Rt, offset(Rs)	1/1
Store Right:	swr	Rt, offset(Rs)	1/1
System Call:	syscall		1/1
Exclusive OR:	xor	Rd, Rs, Rt	1/1
Exclusive OR Immediate:	xori	Rt, Rs, Imm	1/1

Macro instructions

Name	Syntax		Space/Time
Absolute Value:	abs	Rd, Rs	3/3
Branch if Equal to Zero:	beqz	Rs, Label	1/1
Branch if Greater Than or Equal :	bge	Rs, Rt, Label	2/2
Branch if Greater Than or Equal Unsigned:	bgeu	Rs, Rt, Label	2/2
Branch if Greater Than:	bgt	Rs, Rt, Label	2/2
Branch if Greater Than Unsigned:	bgtu	Rs, Rt, Label	2/2
Branch if Less Than or Equal:	ble	Rs, Rt, Label	2/2
Branch if Less Than or Equal Unsigned:	bleu	Rs, Rt, Label	2/2
Branch if Less Than:	blt	Rs, Rt, Label	2/2
Branch if Less Than Unsigned:	bltu	Rs, Rt, Label	2/2
Branch if Not Equal to Zero:	bnez	Rs, Label	1/1
Branch Unconditional:	b	Label	1/1
Divide:	div	Rd, Rs, Rt	4/41
Divide Unsigned:	divu	Rd, Rs, Rt	4/41
Load Address:	la	Rd, Label	2/2
Load Immediate:	li	Rd, value	2/2
Move:	move	Rd, Rs	1/1
Multiply:	mul	Rd, Rs, Rt	2/33
Multiply (with overflow exception):	mulo	Rd, Rs, Rt	7/37
Multiply Unsigned (with overflow exception):	mulou	Rd, Rs, Rt	5/35
Negate:	neg	Rd, Rs	1/1
Negate Unsigned:	negu	Rd, Rs	1/1
Nop:	nop		1/1

Not:	not	Rd, Rs	1/1
Remainder Unsigned:	remu	Rd, Rs, Rt	4/40
Rotate Left Variable:	rol	Rd, Rs, Rt	4/4
Rotate Right Variable:	ror	Rd, Rs, Rt	4/4
Remainder:	rem	Rd, Rs, Rt	4/40
Rotate Left Constant:	rol	Rd, Rs, sa	3/3
Rotate Right Constant:	ror	Rd, Rs, sa	3/3
Set if Equal:	seq	Rd, Rs, Rt	4/4
Set if Greater Than or Equal:	sge	Rd, Rs, Rt	4/4
Set if Greater Than or Equal Unsigned:	sgeu	Rd, Rs, Rt	4/4
Set if Greater Than:	sgt	Rd, Rs, Rt	1/1
Set if Greater Than Unsigned:	sgtu	Rd, Rs, Rt	1/1
Set if Less Than or Equal:	sle	Rd, Rs, Rt	4/4
Set if Less Than or Equal Unsigned:	sleu	Rd, Rs, Rt	4/4
Set if Not Equal:	sne	Rd, Rs, Rt	4/4
Unaligned Load Halfword Unsigned:	ulh	Rd, n(Rs)	4/4
Unaligned Load Halfword:	ulhu	Rd, n(Rs)	4/4
Unaligned Load Word:	ulw	Rd, n(Rs)	2/2
Unaligned Store Halfword:	ush	Rd, n(Rs)	3/3
Unaligned Store Word:	usw	Rd, n(Rs)	2/2

System I/0 Services

Service	Code in $v0	Arguments	Results
Print an Integer	**1**	$a0 = Integer Value to be Printed	
Print Float	2		
Print Double	3		
Print a String	**4**	$a0 = Address of String in Memory	
Read an Integer	**5**		Integer Returned in $v0
Read Float	6		
Read Double	7		
Read a String	**8**	$a0 = Address of Input Buffer in Memory $a1 = Length of Buffer (n)	
Sbrk	9	$a0 = amount	Address in $v0
Exit	**10**		

The system call Read Integer reads an entire line of input from the keyboard up to and including the newline. Characters following the last digit in the decimal number are ignored. Read String has the same semantics as the Unix library routine fgets. It reads up to n – 1 characters into a buffer and terminates the string with a null byte. If fewer than n – 1 characters are on the current line, Read String reads up to and including the newline and again null-terminates the string. Print String will display on the terminal the string of characters found in memory starting with the location pointed to by the address stored in $a0. Printing will stop when a null character is located in the string. Sbrk returns a pointer to a block of memory containing n additional bytes. Exit terminates the user program execution and returns control to the operating system.