APPENDIX D

Macro Instructions

Name	Actual Code	Space/Time
Absolute Value:		

abs Rd, Rs	addu Rd, $0, Rs
	bgez Rs, 1
	sub Rd, $0, Rs
		3/3

Branch if Equal to Zero:		
beqz Rs, Label	beq Rs, $0, Label	**1/1**
Branch if Greater than or Equal :		
bge Rs, Rt, Label	slt $at, Rs, Rt beq $at, $0, Label	**2/2**
If Reg.File[Rs] > = Reg.File[Rt] branch to Label Used to compare values represented in the two's complement number system.		
Branch if Greater than or Equal Unsigned		
bgeu Rs, Rt, Label	sltu $at, Rs, Rt beq $at, $0, Label	**2/2**
If Reg.File[Rs] > = Reg.File[Rt] branch to Label Used to compare addresses (unsigned values).		
Branch if Greater Than:		
bgt Rs, Rt, Label	slt $at, Rt, Rs bne $at, $0, Label	**2/2**
If Reg.File[Rs] > Reg.File[Rt] branch to Label Used to compare values represented in the two's complement number system.		
Branch if Greater Than Unsigned:		
bgtu Rs, Rt, Label	sltu $at, Rt, Rs bne $at, $0, Label	**2/2**
If Reg.File[Rs] > Reg.File[Rt] branch to Label Used to compare addresses (unsigned values).		
Branch if Less Than or Equal:		
ble Rs, Rt, Label	slt $at, Rt, Rs beq $at, $0, Label	**2/2**
If Reg.File[Rs] < = Reg.File[Rt] branch to Label Used to compare values represented in the two's complement number system.		

Branch if Less Than or Equal Unsigned:

bleu Rs, Rt, Label	sltu $at, Rt, Rs	**2/2**
	beq $at, $0, Label	

If Reg.File[Rs] < = Reg.File[Rt] branch to Label
Used to compare addresses (unsigned values).

Branch if Less Than:

blt Rs, Rt, Label	slt $at, Rs, Rt	**2/2**
	bne $at, $0, Label	

If Reg.File[Rs] < Reg.File[Rt] branch to Label
Used to compare values represented in the two's complement number system

Branch if Less Than Unsigned:

bltu Rs, Rt, Label	sltu $at, Rs, Rt	**2/2**
	bne $at, $0, Label	

If Reg.File[Rs] < Reg.File[Rt] branch to Label
Used to compare addresses (unsigned values).

Branch if Not Equal to Zero:

bnez Rs, Label	bne Rs, $0, Label	**1/1**

Branch Unconditional

b Label	bgez $0, Label	**1/1**

Divide:

div Rd, Rs, Rt		bne Rt, $0, ok	**4/41**
		break $0	
	ok:	div Rs, Rt	
		mflo Rd	

Divide Unsigned:

divu Rd, Rs, Rt		bne Rt, $0, ok	**4/41**
		break $0	
	ok:	divu Rs, Rt	
		mflo Rd	

Load Address:

la Rd, Label	lui $at, Upper 16-bits of Label	**2/2**
	ori Rd, $at, Lower 16-bits of Label	

Used to initialize pointers.

Load Immediate:

li Rd, value	lui $at, Upper 16-bits of value	**2/2**
	ori Rd, $at, Lower 16-bits of value	

Initialize registers with negative constants and values greater than 32767.

Load Immediate:

li Rd, value	ori Rt, $0, value	**1/1**

Initialize registers with positive constants less than 32768.

Move:

move Rd, Rs	addu Rd, $0, Rs	**1/1**
mul Rd, Rs, Rt	mult Rs, Rt	**2/33**
	mflo Rd	

Multiply (with overflow exception):

mulo Rd, Rs, Rt	mult Rs, Rt	**7/37**
	mfhi $at	
	mflo Rd	
	sra Rd, Rd, 31	
	beq $at, Rd, ok	
	break $0	
ok:	mflo Rd	

Multiply Unsigned (with overflow exception):

mulou Rd, Rs, Rt	multu Rs, Rt	**5/35**
	mfhi $at	
	beq $at, $0, ok	
ok:	break $0	
	mflo Rd	

Negate:

neg Rd, Rs	sub Rd, $0, Rs	**1/1**

Two's complement negation. An exception is generated when there is an attempt to negate the most negative value: 2,147,483,648.

Negate Unsigned:

negu Rd, Rs	subu Rd, $0, Rs	**1/1**

Nop:

nop	or $0, $0, $0	**1/1**

Used to solve problems with hazards in the pipeline.

Not:

not Rd, Rs	nor Rd, Rs, $0	**1/1**

A bit-wise Boolean complement.

Remainder:

rem Rd, Rs, Rt	bne Rt, $0, 8	**4/40**
	break $0	
	div Rs, Rt	
	mfhi Rd	

Remainder Unsigned:

remu Rd, Rs, Rt		bne Rt, $0, ok	**4/40**
		break $0	
	ok:	divu Rs, Rt	
		mfhi Rd	

Rotate Left Variable:

rol Rd, Rs, Rt		subu $at, $0, Rt	**4/4**
		srlv $at, Rs, $at	
		sllv Rd, Rs, Rt	
		or Rd, Rd, $at	

The lower 5-bits in Rt specifys the shift amount.

Rotate Right Variable:

ror Rd, Rs, Rt		subu $at, $0, Rt	**4/4**
		sllv $at, Rs, $at	
		srlv Rd, Rs, Rt	
		or Rd, Rd, $at	

Rotate Left Constant:

rol Rd, Rs, sa		srl $at, Rs, 32-sa	**3/3**
		sll Rd, Rs, sa	
		or Rd, Rd, $at	

Rotate Right Constant:

ror Rd, Rs, sa		sll $at, Rs, 32-sa	**3/3**
		srl Rd, Rs, sa	
		or Rd, Rd, $at	

Set if Equal:

seq Rd, Rs, Rt		beq Rt, Rs, yes	**4/4**
		ori Rd, $0, 0	
		beq $0, $0, skip	
	yes:	ori Rd, $0, 1	
	skip:		

Set if Greater Than or Equal:

sge Rd, Rs, Rt		bne Rt, Rs, yes	**4/4**
		ori Rd, $0, 1	
		beq $0, $0, skip	
	yes:	slt Rd, Rt, Rs	
	skip:		

Set if Greater Than or Equal Unsigned:

sgeu Rd, Rs, Rt		bne Rt, Rs, yes	**4/4**
		ori Rd, $0, 1	
		beq $0, $0, skip	
	yes:	sltu Rd, Rt, Rs	
	skip:		

Set if Greater Than:

sgt Rd, Rs, Rt	slt Rd, Rt, Rs	**1/1**

Set if Greater Than Unsigned:

sgtu Rd, Rs, Rt	sltu Rd, Rt, Rs	**1/1**

Set if Less Than or Equal:

sle Rd, Rs, Rt	bne Rt, Rs, yes	**4/4**
	ori Rd, $0, 1	
	beq $0, $0, skip	
yes:	slt Rd, Rs, Rt	
skip:		

Set if Less Than or Equal Unsigned:

sleu Rd, Rs, Rt	bne Rt, Rs, yes	**4/4**
	ori Rd, $0, 1	
	beq $0, $0, skip	
yes:	sltu Rd, Rs, Rt	
skip:		

Set if Not Equal:

sne Rd, Rs, Rt	beq Rt, Rs, yes	**4/4**
	ori Rd, $0, 1	
	beq $0, $0, skip	
yes:	ori Rd, $0, 0	
skip:		

Unaligned Load Halfword Unsigned:

ulh Rd, 3(Rs)	lb Rd, 4(Rs)	**4/4**
	lbu $at, 3(Rs)	
	sll Rd, Rd, 8	
	or Rd, Rd, $at	

Unaligned Load Halfword:

ulhu Rd, 3(Rs)	lbu Rd, 4(Rs)	**4/4**
	lbu $at, 3(Rs)	
	sll Rd, Rd, 8	
	or Rd, Rd, $at	

Unaligned Load Word:

ulw Rd, 3(Rs	lwl Rd, 6(Rs)	**2/2**
	lwr Rd, 3(Rs)	

Unaligned Store Halfword:

ush Rd, 3(Rs)	sb Rd, 3(Rs)	**3/3**
	srl $at, Rd, 8	
	sb $at, 4(Rs)	

Unaligned Store Word:

usw Rd, 3(Rs)	swl Rd, 6(Rs)	**2/2**
	swr Rd, 3(Rs)	