UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

Lecture 12

Mid-Term Review

CS2440 – FA20
Prof Kevin Long

Exam Format

- The exam will be a quiz like a lecture quiz, on BlackBoard. It will open at 4pm this Wednesday, 7 Oct 2020. You'll have 90 minutes to complete it.
- I am leaning towards letting you begin at any time between 4 and 7, but the cutoff will be at 7pm, so if you start after 530 you are limiting yourself.
- Your exam will be force-submitted at its end time.
- If you are taking this class under CSD accommodations, you will have those for the exam as well.
- I estimate there will be 20 questions, more than you can finish in 90 minutes. That's the plan. Do not panic if you do not finish.
- There will be questions from Ch 1-2 of Patterson & Hennessey, Ch 9-10 of Stallings, App. A of P&H, and the labs. Questions on MIPS and ARM.
- Every student will have a different exam, based on questions written for just this course and just this semester.
- You will see all the questions up front.
- You will have one pass through the exam.

Tips

- BlackBoard is pretty bad about giving partial credit (like, it doesn't at all). So focus first on the questions you feel confident about, so you can make sure those generate points for you.
- Helper sites like Bucknell can help you prepare, but I'll find instructions that it cannot decode to make sure you have grasped a few key concepts.
- Collaborating on tools, favorite sites, helpful formulas, PDF helper sheets, all of that is fine.
- You must have the MIPS data sheet from dropbox available and a PDF of the book, including Appendix A.

COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

MORGAN KAUFMANN

Chapter 1

Computer Abstractions and Technology

COSC 2440 Fall 2020
Computer Organization and Architecture
Kevin B. Long
The University of Houston

*Textbook: William Stallings
Computer Organization and
Architecture10th Edition*

Chapter 9

Number Systems

COSC 2440 Fall 2020
Computer Organization and Architecture
Kevin B. Long
The University of Houston

*Textbook: William Stallings
Computer Organization and
Architecture10th Edition*

Chapter 10

Computer Arithmetic

COMPUTER ORGANIZATION AND DESIGN

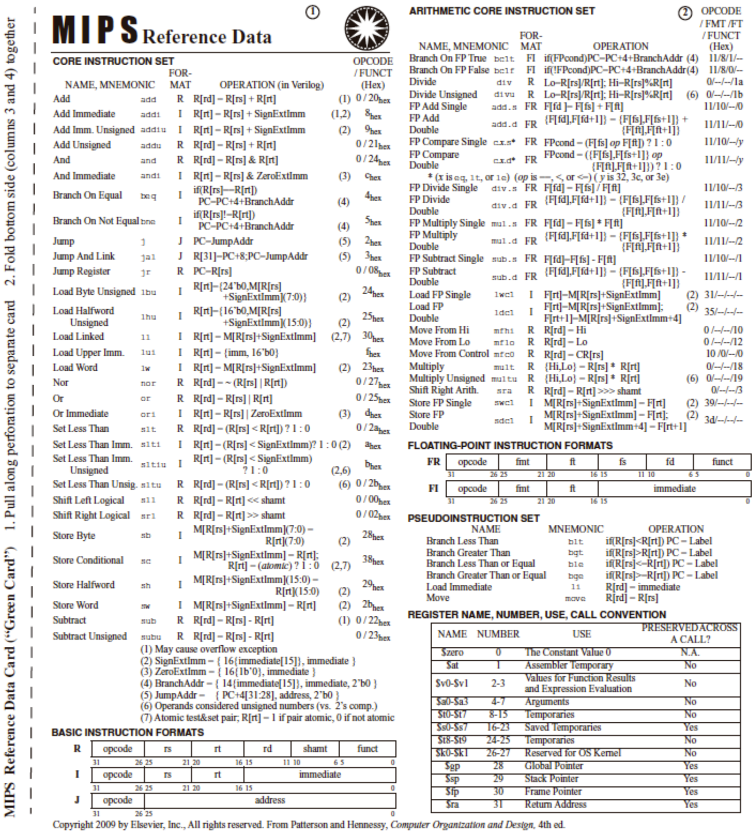
The Hardware/Software Interface

Chapter 2

Instructions: Language of the Computer

Other Resources

- Green card / MIPS reference data sheet
- Section 7.10



MIPS Reference Data Card ("Green Card") 1. Pull along perforation to separate card 2. Fold bottom side (columns 3 and 4) together

MIPS Reference Data ①

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	`add`	R	R[rd] = R[rs] + R[rt]	(1)	$0 / 20_{hex}$
Add Immediate	`addi`	I	R[rt] = R[rs] + SignExtImm	(1,2)	8_{hex}
Add Imm. Unsigned	`addiu`	I	R[rt] = R[rs] + SignExtImm	(2)	9_{hex}
Add Unsigned	`addu`	R	R[rd] = R[rs] + R[rt]		$0 / 21_{hex}$
And	`and`	R	R[rd] = R[rs] & R[rt]		$0 / 24_{hex}$
And Immediate	`andi`	I	R[rt] = R[rs] & ZeroExtImm	(3)	c_{hex}
Branch On Equal	`beq`	I	if(R[rs]==R[rt]) PC=PC+4+BranchAddr	(4)	4_{hex}
Branch On Not Equal	`bne`	I	if(R[rs]!=R[rt]) PC=PC+4+BranchAddr	(4)	5_{hex}
Jump	`j`	J	PC=JumpAddr	(5)	2_{hex}
Jump And Link	`jal`	J	R[31]=PC+8;PC=JumpAddr	(5)	3_{hex}
Jump Register	`jr`	R	PC=R[rs]		$0 / 08_{hex}$
Load Byte Unsigned	`lbu`	I	R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}	(2)	24_{hex}
Load Halfword Unsigned	`lhu`	I	R[rt]={16'b0,M[R[rs] +SignExtImm](15:0)}	(2)	25_{hex}
Load Linked	`ll`	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30_{hex}
Load Upper Imm.	`lui`	I	R[rt] = {imm, 16'b0}		f_{hex}
Load Word	`lw`	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23_{hex}
Nor	`nor`	R	R[rd] = ~ (R[rs] | R[rt])		$0 / 27_{hex}$
Or	`or`	R	R[rd] = R[rs] | R[rt]		$0 / 25_{hex}$
Or Immediate	`ori`	I	R[rt] = R[rs] | ZeroExtImm	(3)	d_{hex}
Set Less Than	`slt`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0		$0 / 2a_{hex}$
Set Less Than Imm.	`slti`	I	R[rt] = (R[rs] < SignExtImm)? 1 : 0	(2)	a_{hex}
Set Less Than Imm. Unsigned	`sltiu`	I	R[rt] = (R[rs] < SignExtImm) ? 1 : 0	(2,6)	b_{hex}
Set Less Than Unsig.	`sltu`	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	(6)	$0 / 2b_{hex}$
Shift Left Logical	`sll`	R	R[rd] = R[rt] << shamt		$0 / 00_{hex}$
Shift Right Logical	`srl`	R	R[rd] = R[rt] >> shamt		$0 / 02_{hex}$
Store Byte	`sb`	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28_{hex}
Store Conditional	`sc`	I	M[R[rs]+SignExtImm] = R[rt]; R[rt] = (*atomic*) ? 1 : 0	(2,7)	38_{hex}
Store Halfword	`sh`	I	M[R[rs]+SignExtImm](15:0) = R[rt](15:0)	(2)	29_{hex}
Store Word	`sw`	I	M[R[rs]+SignExtImm] = R[rt]	(2)	$2b_{hex}$
Subtract	`sub`	R	R[rd] = R[rs] - R[rt]	(1)	$0 / 22_{hex}$
Subtract Unsigned	`subu`	R	R[rd] = R[rs] - R[rt]		$0 / 23_{hex}$

(1) May cause overflow exception
(2) SignExtImm = { 16{immediate[15]}, immediate }
(3) ZeroExtImm = { 16{1b'0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC+4[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2's comp.)
(7) Atomic test&set pair; R[rt] = 1 if pair atomic, 0 if not atomic

BASIC INSTRUCTION FORMATS

R	opcode	rs	rt	rd	shamt	funct
	31 26	25 21	20 16	15 11	10 6	5 0
I	opcode	rs	rt	immediate		
	31 26	25 21	20 16	15 0		
J	opcode	address				
	31 26	25 0				

ARITHMETIC CORE INSTRUCTION SET ②

NAME, MNEMONIC		FOR-MAT	OPERATION		OPCODE / FMT /FT / FUNCT (Hex)
Branch On FP True	`bc1t`	FI	if(FPcond)PC=PC+4+BranchAddr	(4)	11/8/1/--
Branch On FP False	`bc1f`	FI	if(!FPcond)PC=PC+4+BranchAddr	(4)	11/8/0/--
Divide	`div`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]		0/--/--/1a
Divide Unsigned	`divu`	R	Lo=R[rs]/R[rt]; Hi=R[rs]%R[rt]	(6)	0/--/--/1b
FP Add Single	`add.s`	FR	F[fd]= F[fs] + F[ft]		11/10/--/0
FP Add Double	`add.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} + {F[ft],F[ft+1]}		11/11/--/0
FP Compare Single	`c.x.s*`	FR	FPcond = (F[fs] *op* F[ft]) ? 1 : 0		11/10/--/y
FP Compare Double	`c.x.d*`	FR	FPcond = ({F[fs],F[fs+1]} *op* {F[ft],F[ft+1]}) ? 1 : 0		11/11/--/y
* (*x* is `eq`, `lt`, or `le`) (*op* is ==, <, or <=) (*y* is 32, 3c, or 3e)					
FP Divide Single	`div.s`	FR	F[fd] = F[fs] / F[ft]		11/10/--/3
FP Divide Double	`div.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} / {F[ft],F[ft+1]}		11/11/--/3
FP Multiply Single	`mul.s`	FR	F[fd] = F[fs] * F[ft]		11/10/--/2
FP Multiply Double	`mul.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} * {F[ft],F[ft+1]}		11/11/--/2
FP Subtract Single	`sub.s`	FR	F[fd]=F[fs] - F[ft]		11/10/--/1
FP Subtract Double	`sub.d`	FR	{F[fd],F[fd+1]} = {F[fs],F[fs+1]} - {F[ft],F[ft+1]}		11/11/--/1
Load FP Single	`lwc1`	I	F[rt]=M[R[rs]+SignExtImm]	(2)	31/--/--/--
Load FP Double	`ldc1`	I	F[rt]=M[R[rs]+SignExtImm]; F[rt+1]=M[R[rs]+SignExtImm+4]	(2)	35/--/--/--
Move From Hi	`mfhi`	R	R[rd] = Hi		0 /--/--/10
Move From Lo	`mflo`	R	R[rd] = Lo		0 /--/--/12
Move From Control	`mfc0`	R	R[rd] = CR[rs]		10 /0/--/0
Multiply	`mult`	R	{Hi,Lo} = R[rs] * R[rt]		0/--/--/18
Multiply Unsigned	`multu`	R	{Hi,Lo} = R[rs] * R[rt]	(6)	0/--/--/19
Shift Right Arith.	`sra`	R	R[rd] = R[rt] >>> shamt		0/--/--/3
Store FP Single	`swc1`	I	M[R[rs]+SignExtImm] = F[rt]	(2)	39/--/--/--
Store FP Double	`sdc1`	I	M[R[rs]+SignExtImm] = F[rt]; M[R[rs]+SignExtImm+4] = F[rt+1]	(2)	3d/--/--/--

FLOATING-POINT INSTRUCTION FORMATS

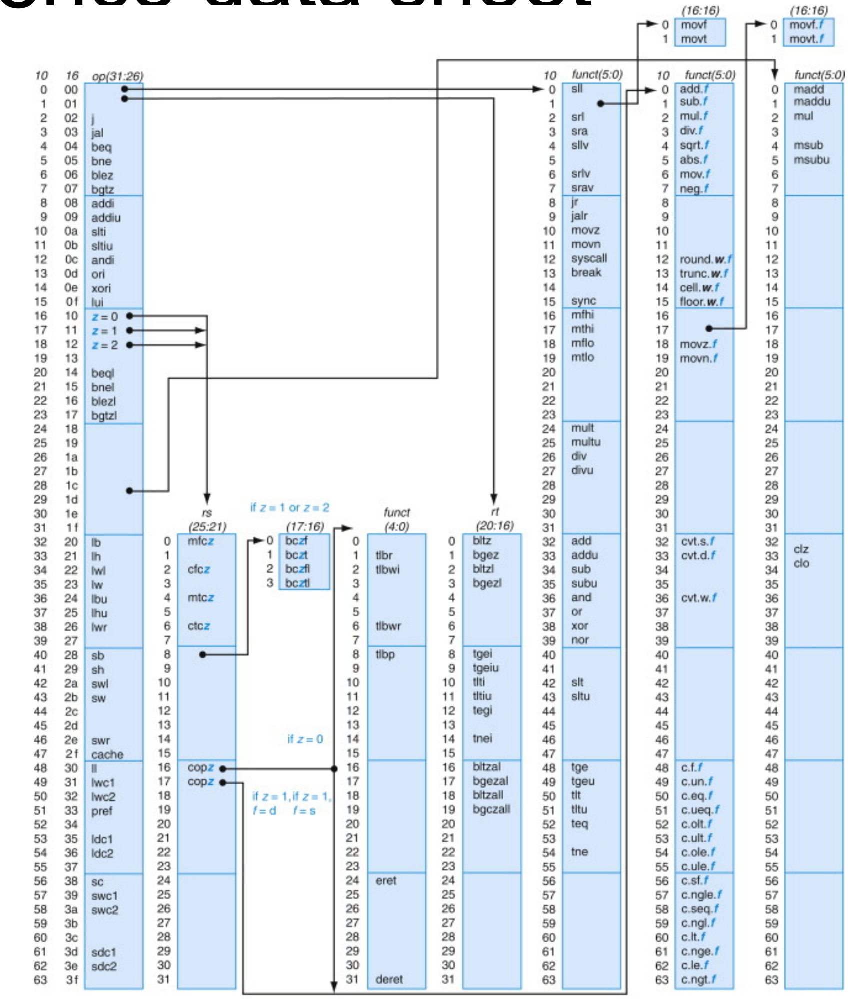
FR	opcode	fmt	ft	fs	fd	funct
	31 26	25 21	20 16	15 11	10 6	5 0
FI	opcode	fmt	ft	immediate		
	31 26	25 21	20 16	15 0		

PSEUDOINSTRUCTION SET

NAME	MNEMONIC	OPERATION
Branch Less Than	`blt`	if(R[rs]<R[rt]) PC = Label
Branch Greater Than	`bgt`	if(R[rs]>R[rt]) PC = Label
Branch Less Than or Equal	`ble`	if(R[rs]<=R[rt]) PC = Label
Branch Greater Than or Equal	`bge`	if(R[rs]>=R[rt]) PC = Label
Load Immediate	`li`	R[rd] = immediate
Move	`move`	R[rd] = R[rs]

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes



10	16	op(31:26)	funct(5:0)	funct(5:0)	funct(5:0)
0	00		sll	add.f	madd
1	01			sub.f	maddu
2	02	j	srl	mul.f	mul
3	03	jal	sra	div.f	
4	04	beq	sllv	sqrt.f	msub
5	05	bne		abs.f	msubu
6	06	blez	srlv	mov.f	
7	07	bgtz	srav	neg.f	
8	08	addi	jr		
9	09	addiu	jalr		
10	0a	slti	movz		
11	0b	sltiu	movn		
12	0c	andi	syscall	round.w.f	
13	0d	ori	break	trunc.w.f	
14	0e	xori		cell.w.f	
15	0f	lui	sync	floor.w.f	
16	10	z = 0	mfhi		
17	11	z = 1	mthi		
18	12	z = 2	mflo	movz.f	
19	13		mtlo	movn.f	
20	14	beql			
21	15	bnel			
22	16	blezl			
23	17	bgtzl			
24	18		mult		
25	19		multu		
26	1a		div		
27	1b		divu		
28	1c				
29	1d				
30	1e				
31	1f				
32	20	lb	add	cvt.s.f	clz
33	21	lh	addu	cvt.d.f	clo
34	22	lwl	sub		
35	23	lw	subu		
36	24	lbu	and	cvt.w.f	
37	25	lhu	or		
38	26	lwr	xor		
39	27		nor		
40	28	sb			
41	29	sh			
42	2a	swl	slt		
43	2b	sw	sltu		
44	2c				
45	2d				
46	2e	swr			
47	2f	cache			
48	30	ll	tge	c.f.f	
49	31	lwc1	tgeu	c.un.f	
50	32	lwc2	tlt	c.eq.f	
51	33	pref	tltu	c.ueq.f	
52	34		teq	c.olt.f	
53	35	ldc1		c.ult.f	
54	36	ldc2	tne	c.ole.f	
55	37			c.ule.f	
56	38	sc		c.sf.f	
57	39	swc1		c.ngle.f	
58	3a	swc2		c.seq.f	
59	3b			c.ngl.f	
60	3c			c.lt.f	
61	3d	sdc1		c.nge.f	
62	3e	sdc2		c.le.f	
63	3f			c.ngt.f	

	rs (25:21)	funct (4:0)	rt (20:16)
0	mfcz		bltz
1		tlbr	bgez
2	cfcz	tlbwi	bltzl
3			bgezl
4	mtcz		
5			
6	ctcz	tlbwr	
7			
8		tlbp	tgei
9			tgeiu
10			tlti
11			tltiu
12			tegi
13			
14			tnei
15			
16	copz		bltzal
17	copz		bgezal
18			bltzall
19			bgczall
20			
21			
22			
23			
24		eret	
25			
26			
27			
28			
29			
30			
31		deret	

	(17:16)
0	bczf
1	bczt
2	bczfl
3	bcztl

	(16:16)	(16:16)
0	movf	movf.f
1	movt	movt.f

Topics

- 8 Great Ideas
- Calculating processing time with multiple CPUs
- Wafer die counts based on size of wafer and size of each die and % lost due to edges
- Wafer yield based on random defect distribution
- Average CPI across multiple types of instructions given IC and CPI

Topics

- Execution time based on IC and CPI and CPU speed
- Change in execution time given a change of execution time of a given type of instruction
- Knowing register types (temporary, for passing arguments, getting values back, ones found on the stack, etc.)
- Big Endian vs Little Endian
- Range of numbers given n bits and the class (unsigned (pos) integers, 2’s complement, sign+magnitude)

Topics

- Converting a decimal number to an arbitrary base
- Identifying what instruction type a MIPS command is
- Decoding a hex value to a MIPS command and vice-versa, passing through binary
- Knowing when to sign-extend and what that means
- Shifting left and right

Topics

- Basic changes like inverting numbers, reversing bits, multiplying or dividing by powers of two
- Knowing what a word is and how big it is
- Knowing when to use j, jal, jr
- Knowing a little about atomic operations and when to use them
- Knowing how sw and lw will change memory and registers