School of Electrical Engineering and Computer Science

CptS 260 - Introduction to Computer Architecture
Fall 2019

Midterm #1
September 25, 2019
Duration: 50 minutes

NAME: SOLUTION. ID:

	Total Points	Earned
Problem 1	15	
Problem 2	15	
Problem 3	20	
Problem 4	15	
Problem 5	10	
Problem 6	20	

Notes:

- You may bring a calculator to the test. No other resources are allowed! In particular, NO textbook, lecture notes, internet access, smartphone usage, etc. are allowed!
- Make sure to write your name and WSU ID down on the first page
- Show your work for each question.
- MIPS reference data is provided!

1. **(15 points)** Assume for a given program, 60% of the executed instructions are of Class A, 25% are of Class B, and 15% are of Class C. Furthermore, assume that an instruction in Class A requires 2 cycles, an instruction in Class B requires 4 cycles, and an instruction in Class C requires 5 to complete. Compute the overall CPI for this program.

Class	%	CPI
A	60	2
B	25	4
C	15	5

$$CPI_{avg} = \sum \%_i * CPI_i$$

$$= \frac{60}{100} \cdot 2 + \frac{25}{100} \cdot 4 + \frac{15}{100} \cdot 5$$

$$= 1.2 + 1 + 0.75 = \boxed{2.95}$$

2. **(15 Points)** Give the 16-bit unsigned *binary* and *hexadecimal* representations of the decimal number (2060).

/2	
2060	0
1030	0
515	1
257	1
128	0
64	0
32	0
16	0
8	0
4	0
2	0
1	1
0	

16 bit binary $\longrightarrow$ 0000 | 1000 | 0000 | 1100

8 4 2 1

$\downarrow$

hex $\longrightarrow$ $(080C)_{16}$

10 A
11 B
12 C

3. **(20 Points)** Consider the decimal number (-18.625). Write down binary representation of this number using the IEEE 745 single precision format. Clearly specify "Sign", "Exponent" and "Mantissa" fields of the single precision representation.

Sign bit: 1

$(18)_{10} \rightarrow (10010)_2$

	×2
0.625	1
0.25	0
0.5	1
1	

$(0.65)_{10} \rightarrow (101)_2$

$(-1)^1(10010.101)$

$(-1)^1(1.\underbrace{0010101}_{\text{Mantissa}}) \times 2^4$ ← exponent

$\text{Exp} + \text{Bias} = 4 + 127$

$= (131)_{10}$

$\downarrow$

$(10000011)_2$

Final answer: (IEEE Format

$\underbrace{1}_{\text{sign bit}} \quad \underbrace{100\,000\,11}_{\text{8 bit exponent}} \quad \underbrace{0010101 \ldots\ldots 0000}_{\text{23 bits Mantissa}}$

4. **(15 points)**
   a. What decimal number is represented by the following single precision float?

1 | 10000100 | 01100000000000000000000

negative

exp: 10000100 → $2^7 + 2^2 = 132$

7 6 5 4 3 2 1 0

$-127 = \boxed{5}$

(2) $(-1)^1 (1.0110) \times 2^5$

-1 -2

(10) $(-1)^1 (1 + 2^{-2} + 2^{-3}) \times 2^5 = (-1)^1 (1.375) \times 2^5$

$= -44$

b. How many bits are <u>needed</u> to represent 436(Hex) in binary

0 | 100 | 0011 | 0110

$\boxed{\text{11 bits}}$ needed

c. Show the 2's complement *binary* (8 bits) representation of $-102_{(10)}$

102	0
51	1
25	1
12	0
6	0
3	1
1	1
0	

0 1 1 0 0 1 1 0

1 0 0 1 1 0 0 1 flip

\+ 1 +1

$\boxed{10011010_{(2)}}$

5. **(10 points)**

a. For the following MIPS assembly instructions above, what is a corresponding C/JAVA statement?

$$add\ f, g, h$$

$$add\ f, i, f$$

f = g+h;

f = f+i; //OR f += i;

b. What is the binary instruction sequence for the following MIPS instruction:

$$sll\ \$t2, \$t0, 4$$

↳ shamt

\$t0 - 8 (rt)
\$t2 - 10 (rd)

R-type instruction:

opcode	rs	rt	rd	shamt	func
000000	00000	01000	00101	00100	00000

6. **(20 points)**
Assume 185 and 122 are unsigned 8-bit decimal integers. Calculate 185 - 122. Is there overflow?

We will do the subtraction directly without using 2's complement.

$$
\begin{array}{lcr}
 & & \text{1 11 11} \rightarrow \text{Borrow} \\
(185)_{10} & \rightarrow & 1011\ 1001 \\
-\ (122)_{10} & \rightarrow & -\ 0111\ 1010 \\
\hline
 & & 0011\ 1111
\end{array}
$$

No overflow since result is +ve & MSB is zero

Rules for sub

0 - 0 = 0, borrow = 0 (no borrow)
1 - 0 = 1, borrow = 0
0 - 1 = 1, borrow = 1
1 - 1 = 0, borrow = 0

Bonus Question: (5 points)

Using the IEEE 754 single precision floating point format, write down the bit pattern that would represent -1/4. Can you represent -1/4 exactly?

$$\frac{1}{4} \rightarrow \frac{1}{2^2} \rightarrow 1 \times 2^{-2} = .01$$

$$1 \quad \underbrace{0\,1\,1\,1\,1\,1\,0\,1}_{exp} \quad \underbrace{0\,0\,0\,0\ldots 0\,0}_{23\text{ bit}} \quad = \quad \begin{array}{c} exp + bias \\ -2 + 127 = \underline{125} \end{array}$$

$(125)_{10}$

$\rightarrow 01111101$

and <u>yes</u> you can represent -1/4 exactly.

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

(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 (31–26)	rs (25–21)	rt (20–16)	rd (15–11)	shamt (10–6)	funct (5–0)
I	opcode (31–26)	rs (25–21)	rt (20–16)	immediate (15–0)		
J	opcode (31–26)	address (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 (31–26)	fmt (25–21)	ft (20–16)	fs (15–11)	fd (10–6)	funct (5–0)
FI	opcode (31–26)	fmt (25–21)	ft (20–16)	immediate (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