Ryerson University
Department of Electrical, Computer and Biomedical Engineering
COE 608–Computer Organization and Architecture

Midterm Test **March 7, 2019**

Name: _____ **Student Number:** _____ **Sec:----------**

Time limit: 2 hours Examiners: N. Mekhiel

Notes:
a) Closed book.
b) No calculators.
c) Answer all questions **in the space provided**. Total Marks= 50

Q1- Assume the following C code :-

```
for(i=0; i<=2000; i++) {
int temp;
temp= X[i];
X[i]=Y[i];
Y[i]=temp;
                    }
```

Assume that $S0 has the address of X[0] and $S1 has the address of Y[0] and i is in $S2.

1.1-(10 Marks) Write the above code using MIPS instructions.

```
addi $s2, $0, 0 ;   i = 0                    } initialize i (2)
addi $s3, $0, 2001; s3 = 2001

Loop: sll t0, s2, 2;   4 i                   } address of X[i], Y[i] (3)
      add t1, s0, t0;  &X[i]
      add t2, s1, t0;  &Y[i]

      lw t3, 0(t1) ;   t3 = X[i]             } SWAP (4)
      lw t4, 0(t2);    t4 = Y[i]
      sw t4, 0(t1);    X[i] = Y[i]
      sw t3, 0(t2);    Y[i] = X[i]
      addi s2, s2, 1 ;  i = i+1              } inc decision } 2
      bne s2, s3, Loop

Exit;
```

10

Name:_____ Section:_____

1.2 (2 Marks) How many instructions are executed during running this code.

$N = 2001 * (9 \mid +2 = 18011$

2

1.3 (3 Marks) Find the performance of above code in MIPS 2 GHz processor, assuming that arithmetic instruction takes 1 cycle, data transfer instruction takes 4 cycles, conditional branch takes 2 cycles and jump takes 1 cycle.

$T = (2001 * (4 \times 1 + 4 \times 4 + 1 \times 2) + 2 \times 1) \cdot 5\ ns$

$= 22{,}012 \approx 22\ \mu s$

3

1.4(5 marks) Find average CPI for the above code

$CPI = \frac{4 \times 1 + 4 \times 4 + 1 \times 2}{9} = \frac{22}{9} = 2.45$

5

1.5 (5 Marks) Find performance speed up for above code if data transfer instruction takes 1 cycle using cache

$T_{new} = (2001 * (4 \times 1 + 4 \times 1 + 2 \times 1) + 2 \times 1) \times 1.5\ ns$

$= 10\ \mu s$

$speedup = \frac{22}{10} = 2.2$ Times

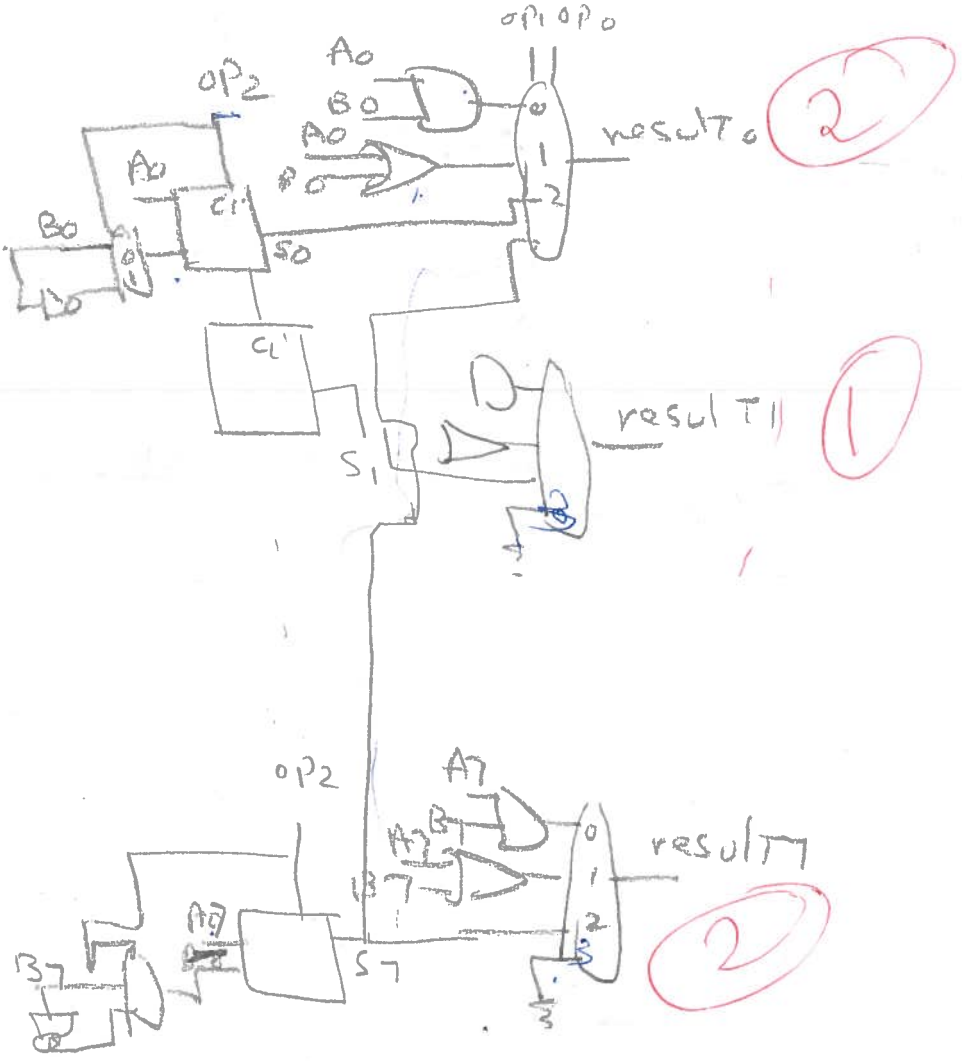
5

Name:_____ Section:_____

Q2.1 (5 Marks) Design 8 bit ALU that performs the following operations based on op2,op1,op0 as:-

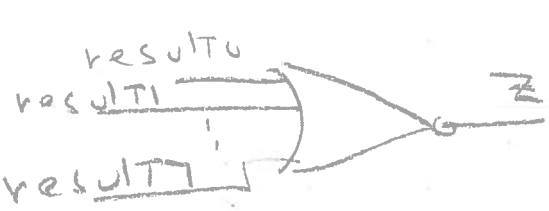
op2	op1	op0	operation
0	0	0	AND
0	0	1	OR
0	1	0	ADD
1	1	0	SUB
1	1	1	SLT

5//



Name:_____ Section:_____

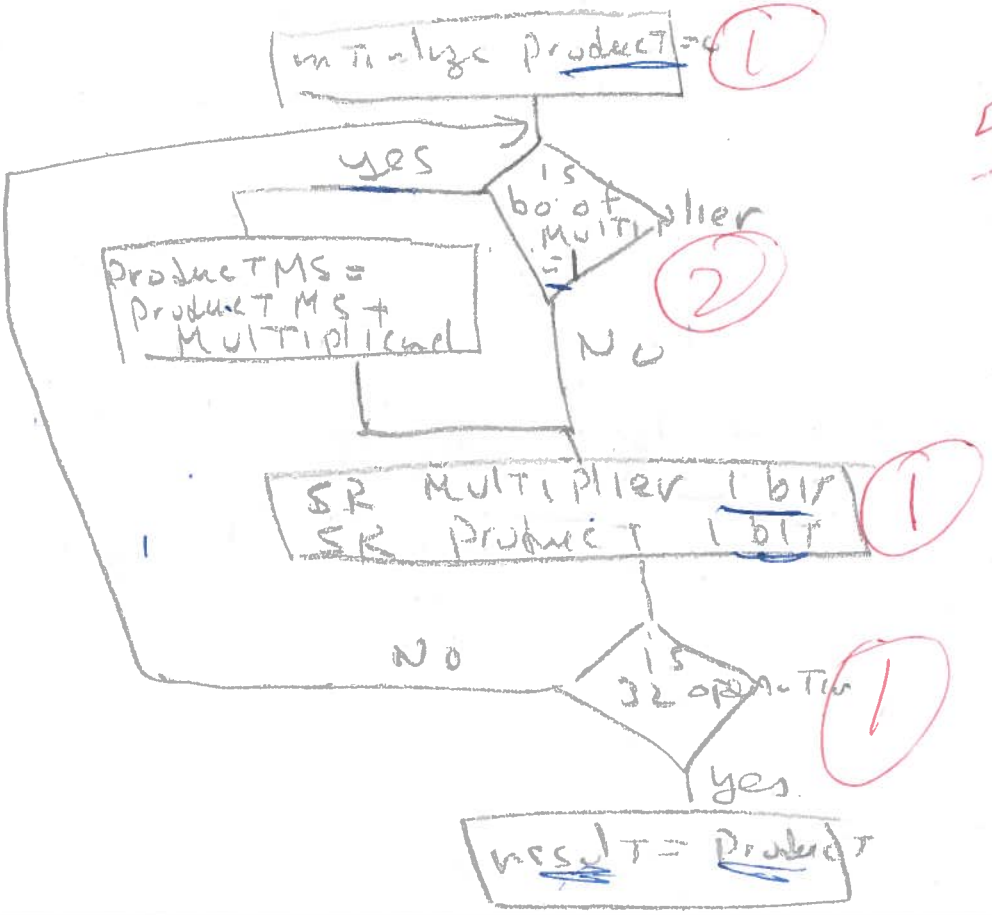
Q2.2(3 marks) Design a circuit for the ALU to detect if the two inputs are equal then an output Z=1 and specify what operation should be selected



opcode = 110 subtract

3/

Q2.3 (5 Marks) Draw a **FLOW CHART** for a 32 bit simple Multiplication Algorithm that uses a 32 bit ALU



5/

Name:_____ Section:_____

Q2.4 (5 Marks) Apply the above algorithm for the following:

Operation ADD OR SHIFT	Product 0 0 0 0 0 0 0 0	Multiplicand 0 1 0 1	Multiplier 0 1 0 1
ADD	0101		
SR	0101 0000		0101
SR	0001 0100		0101
ADD	0101		
	0110 0100		0
SR	0011 0010		
SR	0001 1001		
	+1		

5/7

Q2.5- (7 Marks) Determine the IEEE754 FP the following number:-

-13.625

S = 1

$13 = 1101.101 = 1.101101 \times 2^3$

2 | .625	
1 | .250	
0 | .500	
1 | .000	

(1) (3) (3)

1	10000010	10110100000000000000000	6

$E = 127 + 3 + 130$

7/7

Name:_____ Section:_____

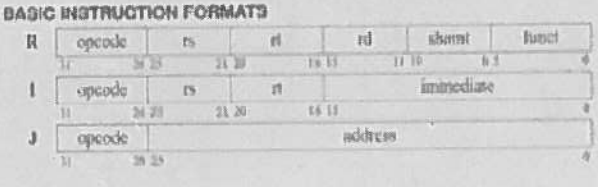
MIPS Reference Data

CORE INSTRUCTION SET

NAME	MNE-MON-IC	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+4;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)	0 / 24hex
Load Halfword Unsigned	lhu	I	R[rt]={16'b0,M[R[rs]+SignExtImm](15:0)}	(2)	0 / 25hex
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		fhex
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	0 / 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 Unsigned	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[rs] >> shamt		0 / 02hex
Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = R[rt](7:0)	(2)	28hex
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{1'b0}, immediate }
(4) BranchAddr = { 14{immediate[15]}, immediate, 2'b0 }
(5) JumpAddr = { PC[31:28], address, 2'b0 }
(6) Operands considered unsigned numbers (vs. 2's comp.)

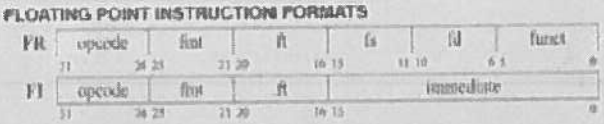
BASIC INSTRUCTION FORMATS



ARITHMETIC CORE INSTRUCTION SET

NAME	MNE-MON-IC	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
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



PSEUDO INSTRUCTION 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