

King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

COE 301 COMPUTER ORGANIZATION
ICS 233: COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE
Term 151 (Fall 2015-2016)
Major Exam 1
Saturday Oct. 10, 2015

Time: 120 minutes, Total Pages: 9

Name: KEY _____ **ID:** _____ **Section:** _____

Notes:

- Do not open the exam book until instructed
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated

Question	Max Points	Score
Q1	35	
Q2	25	
Total	60	

Dr. Aiman El-Maleh
Dr. Mayez Al-Muhammad

(Q1) Fill in the blank in each of the following questions:

- (1) Assuming 12-bit unsigned representation, the binary number 1111 0000 1111 is equal to the decimal number 3855.

- (2) Assuming 12-bit signed 2's complement representation, the hexadecimal number FC0 is equal to the decimal number -64.

- (2) Accessibility to hardware resources is an advantage of programming in assembly language.

- (3) Code portability is an advantage of programming in high-level language.

- (4) With a 36-bit address bus and 64-bit data bus, the maximum memory size (assuming byte addressable memory) that can be accessed by a processor is $2^{36}=64$ GByte and the maximum number of bytes that can be read or written in a single cycle is 8 Bytes.

- (5) The bandwidth mismatch between the speed of processor and the speed of main-memory is alleviated by using cache memory.

- (6) The advantage of dynamic RAM over static RAM is that it is dense and cheap but the disadvantage is that is slow as it needs refreshing.

- (7) The instruction set architecture of a processor consists of the instructions set, the programmer accessible registers and memory.

- (8) Assuming that the CPU has just read a 32-bit MIPS instruction from the address 0x00400008. Then, the address of the next instruction that this CPU is going to read is $0x00400008+4=0x0040000c$.

(9) Given a magnetic disk with the following properties:

- Time of one rotation is 8 ms
- Average seek = 8 ms, Sector = 512 bytes, Track = 200 sectors

The average time to access a block of 20 consecutive sectors is $\underline{8 + 4 + 8 * 20 / 200 = 12.8}$ ms.

(10) The pseudo instruction *neg \$s2, \$s1* (\$s2 is computed as the negative value of \$s1) is implemented by the following minimum MIPS instructions:

subu \$s2, \$0, \$s1

(11) The pseudo instruction *ble \$s2, \$s1, Next* is implemented by the following minimum MIPS instructions:

slt \$at, \$s1, \$s2
beq \$at, \$0, Next

(12) The pseudo instruction *rol \$s0, \$s0, 8* (\$s0 is rotated to the left by 8 bits and stored in \$s0) is implemented by the following minimum MIPS instructions:

srl \$at, \$s0, 24
sll \$s0, \$s0, 8
or \$s0, \$s0, \$at

(13) Assuming that \$a0 contains an Alphabetic character, the instruction *andi \$a0, \$a0, 0xdf* will guarantee that the character in \$a0 is an upper case character. Note that the ASCII code of character 'A' is 0x41 while that of character 'a' is 0x61.

(14) Assume that the instruction *bne \$t0, \$t1, NEXT* is at address 0x00400020 in the text segment, and the label NEXT is at address 0x00400010. Then, the address stored in the assembled instruction for the label NEXT is $(0x00400010 - (0x00400020 + 4)) / 4 = 0xffffb$.

- (15) Assuming that variable Array is defined as shown below:

```
Array: .byte 1, 2, -3, 4
```

After executing the following sequence of instructions, the content of the three registers is \$t1=0xffffffff, \$t2=0x000004fd and \$t3=0x04fd0201.

```
la $t0, Array
lb $t1, 2($t0)
lh $t2, 2($t0)
lw $t3, 0($t0)
```

- (16) Assuming the following data segment, and assuming that the first variable X is given the address **0x10010000**, then the addresses for variables Y and Z will be 0x10010002 and 0x10010008.

```
.data
X: .byte 1
Y: .half 2, 3
Z: .word 4
```

- (17) To multiply the **signed** content of register \$t0 by 127.75 without using multiplications and division instructions, we use the following MIPS instructions:

```
sll $t1, $t0, 7
sra $t2, $t0, 2
subu $t0, $t1, $t2
```

- (18) The condition for which the data stored in \$t0 must satisfy in order for the following MIPS fragment to branch to L1 is:

If bit 0 and bit 4 and bit 8 are all equal to 1 then branch to L1

```
ori $t1, $0, 0x111
and $t0, $t0, $t1
beq $t0, $t1, L1
```

- (19) The content of register **\$t0** after executing the following code is 1+2+3+4=0xa:

```
li $s1, 0x4321
xor $t0, $t0, $t0
Next:
andi $t1, $s1, 0xf
add $t0, $t0, $t1
srl $s1, $s1, 4
bne $s1, $0, Next
```

[25 Points]

(Q2) Write separate MIPS assembly code fragments with minimum instructions to implement each of the given requirements.

- (i) [6 points] Given two arrays of words A and B with their base addresses stored in registers \$s0 and \$s1, array size N is stored in \$s2, and index i is stored \$s3, write the smallest MIPS assembly fragment for the following computation:

for (i=0; i<n; i++) if ((A[i]-B[i])*5 >=0) then A[i]= (A[i]-B[i])*5;

Solution:

```

xor $s3, $s3, $s3 # i = 0
Loop: bge $s3, $s2, EndFor # End loop if i>=n
      lw $t0, 0($s0) # $t0 = A[i]
      lw $t1, 0($s1) # $t1 = B[i]
      sub $t0, $t0, $t1 # $t0= A[i] - B[i]
      sll $t1, $t0, 2 # $t1= (A[i] - B[i])*4
      add $t1, $t1, $t0 # $t1= (A[i] - B[i])*5
      bltz $t1, skip # do not update if (A[i] - B[i])*5 < 0
      sw $t1, 0($s0) # A[i] = (A[i] - B[i])*5
Skip:  addiu $s0, $s0, 4 # update pointer to A[i+1]
      addiu $s1, $s1, 4 # update pointer to B[i+1]
      addiu $s3, $s3, 1 # update pointer i
      J Loop
EndFor:

```

Grading: 2 pts for the For, 1 pt for the If, 2 pts for the expression, and 1 pt for the store.

- (ii) [6 points] Given the following MPIS assembly fragment:

```

bne $s1, $s2, exit
bge $s2, $s3, exit
addi $s4, $s4, 5

```

Exit:

Assume that variables a, b, c, and d are stored in registers \$s1, \$s2, \$s3, and \$s4, respectively.

Fill in the Boolean expression in the following IF statement:

If (_____) then d=d +5;

Answer: if ((a == b) && (b < c)) then d=d +5;

Repeat the above question for the following MPIS assembly fragment:

```

beq $s1, $s2, process
bgt $s2, $s3, exit
ble $s3, $s4, exit
process: add $s4, $s4, $s1
Exit:

```

Fill in the Boolean expression in the following IF statement:

If (_____) then d=d +a;

Answer: if ((a ==b) || (b <= c) && (c > d)) then d=d +a;

(iii) [3 points] Write a MIPS assembly fragment for the following IF statement:

if ([(a == b) || (c== d)] && (a < c)) then b = d ;

Assume that variables a, b, c, and d are stored into registers \$s0, \$s1, \$s2, and \$s3, respectively.

Answer:

```

process:
    bge $s0, $s2, exit
    beq $s0,$s1, process
    bne $s2, $s3, exit
    Add $s1, $s3, $zero
Exit:

```

(iv) [5 points] Write a MIPS assembly fragment to count the number of occurrence of alpha characters (can be lowercase or uppercase) in a null terminating string, where the base address of the string is in register \$s0 and the count is to be in \$s1.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Answer:

```

    ori    $t2,$zero, 0x41    #Ascii of char A in t2
    ori    $t3,$zero, 0x5A    #Ascii of char Z in t3
    xor    $s1, $s1, $s1      # initialize count to zero
    move   $t0, $s0          # $t0 points to first char of string
Loop: lb    $t1, 0($t0)      # load byte into $t1
    beq    $t1, $zero, exit   # Exit if null terminating char
    andi   $t1, $t1, 0xDF     # Convert to capital (if any)
    blt    $t1, $t2, Skip     # Skip if below 0x41 (not an alpha)
    bgt    $t1, $t3, Skip     # Skip if above 0x5A (not an alpha)
    addiu  $s1, $s1, 1        # increment count

```

Skip: `addi $t0, $t0, 1` **# Increment string ptr**
J **Loop**

Exit:

Grading: -1 pt if both domains are tested.

- (v) [5 points] Write the most optimized MIPS assembly fragment for the following WHILE statement:

```
i = 0;
WHILE ( ( A[i] >= B[i]*2 ) && (i<N) ) { A[i] = A[i]- B[i]; i = i+1; }
```

Where A and B are arrays of Bytes. The base address of arrays A and B are stored into registers \$s0 and \$s1, respectively. The index i and count N are stored into registers \$s2 and \$s3.

Answer:

```

xor   $s2, $s2, $s2      # index i = 0
Loop: bge   $s2, $s3, exit # exit if i >= N
lb    $t0, 0 ($s0)      # $t0 = A[i] as byte
lb    $t1, 0 ($s1)      # $t1 = B[i] as byte
sll   $t2, $t1, 1       # t2 =B[i]*2
blt   $t0, $t2, exit   # exit if A[i] < B[i]*2
sub   $t0, $t0, $t1     # $t0= A[i]-B[i];
sb    $t0, 0 ($s0)     # A[i] = A[i]-B[i];
addiu $s2, $s2, 1      # i = i+1
addiu $s0, $s0, 1      # update ptr to A[i+1]
addiu $s1, $s1, 1      # update ptr to B[i+1]
J     Loop
```

Exit:

MIPS Instructions:

Instruction	Meaning	R-Type Format					
add \$s1, \$s2, \$s3	$\$s1 = \$s2 + \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x20
addu \$s1, \$s2, \$s3	$\$s1 = \$s2 + \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x21
sub \$s1, \$s2, \$s3	$\$s1 = \$s2 - \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x22
subu \$s1, \$s2, \$s3	$\$s1 = \$s2 - \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x23

Instruction	Meaning	R-Type Format					
and \$s1, \$s2, \$s3	$\$s1 = \$s2 \& \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x24
or \$s1, \$s2, \$s3	$\$s1 = \$s2 \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x25
xor \$s1, \$s2, \$s3	$\$s1 = \$s2 \wedge \$s3$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x26
nor \$s1, \$s2, \$s3	$\$s1 = \sim(\$s2 \$s3)$	op = 0	rs = \$s2	rt = \$s3	rd = \$s1	sa = 0	f = 0x27

Instruction	Meaning	R-Type Format					
sll \$s1, \$s2, 10	$\$s1 = \$s2 \ll 10$	op = 0	rs = 0	rt = \$s2	rd = \$s1	sa = 10	f = 0
srl \$s1, \$s2, 10	$\$s1 = \$s2 \gg 10$	op = 0	rs = 0	rt = \$s2	rd = \$s1	sa = 10	f = 2
sra \$s1, \$s2, 10	$\$s1 = \$s2 \gg 10$	op = 0	rs = 0	rt = \$s2	rd = \$s1	sa = 10	f = 3
sllv \$s1, \$s2, \$s3	$\$s1 = \$s2 \ll \$s3$	op = 0	rs = \$s3	rt = \$s2	rd = \$s1	sa = 0	f = 4
srlv \$s1, \$s2, \$s3	$\$s1 = \$s2 \gg \$s3$	op = 0	rs = \$s3	rt = \$s2	rd = \$s1	sa = 0	f = 6
srav \$s1, \$s2, \$s3	$\$s1 = \$s2 \gg \$s3$	op = 0	rs = \$s3	rt = \$s2	rd = \$s1	sa = 0	f = 7

Instruction	Meaning	I-Type Format				
addi \$s1, \$s2, 10	$\$s1 = \$s2 + 10$	op = 0x8	rs = \$s2	rt = \$s1	imm ¹⁶ = 10	
addiu \$s1, \$s2, 10	$\$s1 = \$s2 + 10$	op = 0x9	rs = \$s2	rt = \$s1	imm ¹⁶ = 10	
andi \$s1, \$s2, 10	$\$s1 = \$s2 \& 10$	op = 0xc	rs = \$s2	rt = \$s1	imm ¹⁶ = 10	
ori \$s1, \$s2, 10	$\$s1 = \$s2 10$	op = 0xd	rs = \$s2	rt = \$s1	imm ¹⁶ = 10	
xori \$s1, \$s2, 10	$\$s1 = \$s2 \wedge 10$	op = 0xe	rs = \$s2	rt = \$s1	imm ¹⁶ = 10	
lui \$s1, 10	$\$s1 = 10 \ll 16$	op = 0xf	0	rt = \$s1	imm ¹⁶ = 10	

Instruction	Meaning	Format				
j label	jump to label	op ⁶ = 2	imm ²⁶			
beq rs, rt, label	branch if (rs == rt)	op ⁶ = 4	rs ⁵	rt ⁵	imm ¹⁶	
bne rs, rt, label	branch if (rs != rt)	op ⁶ = 5	rs ⁵	rt ⁵	imm ¹⁶	
blez rs, label	branch if (rs <= 0)	op ⁶ = 6	rs ⁵	0	imm ¹⁶	
bgtz rs, label	branch if (rs > 0)	op ⁶ = 7	rs ⁵	0	imm ¹⁶	
bltz rs, label	branch if (rs < 0)	op ⁶ = 1	rs ⁵	0	imm ¹⁶	
bgez rs, label	branch if (rs >= 0)	op ⁶ = 1	rs ⁵	1	imm ¹⁶	

Instruction	Meaning	Format					
slt rd, rs, rt	rd=(rs<rt?1:0)	op ⁶ = 0	rs ⁵	rt ⁵	rd ⁵	0	0x2a
sltu rd, rs, rt	rd=(rs<rt?1:0)	op ⁶ = 0	rs ⁵	rt ⁵	rd ⁵	0	0x2b
slti rt, rs, imm ¹⁶	rt=(rs<imm?1:0)	0xa	rs ⁵	rt ⁵	imm ¹⁶		
sltiu rt, rs, imm ¹⁶	rt=(rs<imm?1:0)	0xb	rs ⁵	rt ⁵	imm ¹⁶		

Instruction		Meaning	I-Type Format			
lb	rt, imm ¹⁶ (rs)	rt = MEM[rs+imm ¹⁶]	0x20	rs ⁵	rt ⁵	imm ¹⁶
lh	rt, imm ¹⁶ (rs)	rt = MEM[rs+imm ¹⁶]	0x21	rs ⁵	rt ⁵	imm ¹⁶
lw	rt, imm ¹⁶ (rs)	rt = MEM[rs+imm ¹⁶]	0x23	rs ⁵	rt ⁵	imm ¹⁶
lbu	rt, imm ¹⁶ (rs)	rt = MEM[rs+imm ¹⁶]	0x24	rs ⁵	rt ⁵	imm ¹⁶
lhu	rt, imm ¹⁶ (rs)	rt = MEM[rs+imm ¹⁶]	0x25	rs ⁵	rt ⁵	imm ¹⁶
sb	rt, imm ¹⁶ (rs)	MEM[rs+imm ¹⁶] = rt	0x28	rs ⁵	rt ⁵	imm ¹⁶
sh	rt, imm ¹⁶ (rs)	MEM[rs+imm ¹⁶] = rt	0x29	rs ⁵	rt ⁵	imm ¹⁶
sw	rt, imm ¹⁶ (rs)	MEM[rs+imm ¹⁶] = rt	0x2b	rs ⁵	rt ⁵	imm ¹⁶