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 161 (Fall 2016-2017)
Major Exam 1
Saturday Oct. 22, 2016

Time: 90 minutes, Total Pages:

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	22	
Q2	14	
Total	36	

Dr. Aiman El-Maleh
Dr. Marwan Abu-Amara

[22 Points]

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

(1) Assuming 12-bit signed 2's complement representation, the binary number 1100 0000 0011 is equal to the decimal number -1021.

(2) Assuming 16-bit signed 2`s complement representation, the hexadecimal number FF00 is equal to the decimal number -256.

(3) There is a one-to-one correspondence between assembly language and machine language.

(4) One main advantage of programming in high-level language is that programs are portable.

(5) Accessing data from random access memory is slower than accessing it from cache memory but faster than accessing it from hard disk memory.

(6) Dynamic RAM is slower than static RAM but is denser and cheaper.

(7) Assuming variable Array is defined as shown below:

```
Array: .word 10, 11, 12, 13, 14
```

The content of register $t0 (in hexadecimal) after executing the following sequence of instructions is 0x0000000B.

```
la $t0, Array
lw $t0, 4($t0)
```

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

- Rotation speed = 7200 RPM (rotations per minute)
- Average seek = 8 ms, Sector = 512 bytes, Track = 200 sectors

The average rotational latency is <u>4.17</u> ms.

(9) The pseudo instruction *ble $s2, 10, Next* is implemented by the following <u>minimum</u> MIPS instructions:

<u>addi $at, $s2, -1</u>
<u>slti $at, $at, 10</u>
<u>bne $at,$0, Next</u>

<u>OR</u>

<u>ori $at, $0, 10</u>
<u>slt $at, $at, $s2</u>
<u>beq $at, $0, Next</u>

(10) The pseudo instruction *ror $s0, $s0, 4* ($s0 is rotated to the right by 4 bits and stored in $s0) is implemented by the following <u>minimum</u> MIPS instructions:

<u>sll $at, $s0, 28</u>
<u>srl $s0, $s0, 4</u>
<u>or $s0, $s0, $at</u>

(11) Assuming that $a0 contains an Alphabetic character, the instruction <u>*xori $a0, $a0, 0x20*</u> will convert the character in $a0 from upper case to lower case and from lower case to upper case. Note that the ASCII code of character 'A' is 0x41 while that of character 'a' is 0x61.

(12) Assume that the instruction *beq $t0, $t1, NEXT* is at address 0x00400030 in the text segment, and the label NEXT is at address 0x00400014. Then, the value stored in the assembled instruction for the label NEXT is <u>(0x00400014-0x00400034)/4=0xFFF8</u>.

(13) 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 (in hexadecimal) is $t1=0x04FDFE01, $t2=0xFFFFFFFE, and $t3=0x000004FD.

```
la   $t0, Array
lw  $t1, 0($t0)
lb   $t2, 1($t0)
lh   $t3, 2($t0)
```

(14) 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 **0x10010004** and **0x1001000C.**

```
.data
X:      .byte  1, 2, 3
Y:      .half   3, 4, 5
Z:      .word 6, 7, 8
```

[14 Points]

(Q2) Write **separate** MIPS assembly code fragments with **minimum** instructions to implement each of the given requirements. You can use pseudo instructions in your solution.

(i) [5 points] Write a MIPS code fragment that computes the number of 0→1 and 1→0 transitions in the content of register $s0 and stores the result in register $s1.The content of register $s0 should be preserved. For example, if $s0=0x75 (=01110101 in binary), then $s1=5.

```
li $s1, 0                #initialize transition counter to 0
move $t0, $s0            # preserve $s0
Loop:
andi $t1, $t0, 3         # check least significant 2 bits
beq $t1, 1, Next         # 1→0 transition from LSB
bne $t1, 2, Skip         # 0→1 transition from LSB
Next:
addi $s1, $s1, 1         # increment transition counter
Skip:
srl $t0, $t0, 1          # examine next 2-bit pair
bne $t0, $0, Loop
```

(ii) [4 points] Write a MIPS code fragment that computes the equation $s0 = $s0*105 without the use of multiplication instructions with the minimum number of instructions. HINT: 105=15*7.

```
sll $t0, $s0, 3
sub $t1, $t0, $s0
sll $t2, $t1, 4
sub $s0, $t2, $t1
```

(iii) [5 points] Given an array of words A with its base address stored in registers $s0, array size n stored in $s1, write the smallest MIPS assembly fragment for the following computation:

Count=0;

for (i=0; i<n-1; i++)

if (A[i]==A[i+1]) then Count++;

```
addi $s1, $s1, -1        #s1=n-1
li $s2, 0                #Count=0
Loop:
lw $t0, 0($s0)
lw $t1, 4($s0)
bne $t0, $t1, Skip       # if ( A[i]==A[i+1])
addi $s2, $s2, 1         # Count++
Skip:
addi $s0, $s0, 4
addi $s1, $s1, -1
bne  $s1, $0, Loop
```

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 ^ $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x26
nor $s1, $s2, $s3	$s1 = ~($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 << 10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 0
srl $s1,$s2,10	$s1 = $s2>>>10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 2
sra $s1, $s2, 10	$s1 = $s2 >> 10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 3
sllv $s1,$s2,$s3	$s1 = $s2 << $s3	op = 0	rs = $s3	rt = $s2	rd = $s1	sa = 0	f = 4
srlv $s1,$s2,$s3	$s1 = $s2>>>$s3	op = 0	rs = $s3	rt = $s2	rd = $s1	sa = 0	f = 6
srav $s1,$s2,$s3	$s1 = $s2 >> $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 ^ 10	op = 0xe	rs = $s2	rt = $s1	imm^{16} = 10
lui $s1, 10	$s1 = 10 << 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}