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 171 (Fall 2017-2018)
Major Exam 1
Saturday Oct. 21, 2017

Time: 120 minutes, Total Pages: 10

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
- No calculators are allowed to be used in the exam

Question	Max Points	Score
Q1	28	
Q2	11	
Q3	17	
Total	56	

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

[28 Points]

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

(1) Assuming 12-bit unsigned number representation, the binary number 1111 1111 0000 is equal to the decimal number 4080.

(2) Assuming 16-bit signed 2`s complement representation, the hexadecimal number FEA0 is equal to the decimal number -352.

(3) Two advantages of programming in assembly language are space and time efficiency and accessibility to system hardware.

(4) Two advantages of programming in high-level language are programs are portable and program development and maintenance are faster.

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

(6) With a 24-bit address bus and 32-bit data bus, the maximum memory size (assuming byte addressable memory) that can be accessed by a processor is 2^{24}=16 MB and the maximum number of bytes that can be read or written in a single cycle is 32/8=4.

(7) The advantage of static RAM over dynamic RAM is that it is faster but the disadvantage is that it is less dense and more expensive.

(8) 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 100 consecutive sectors is 8 ms + 0.5*8 ms + 100/200 * 8 ms = 16 ms.

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

```
Array: .word  10
       .half   11, 12
       .byte  13, 14, 15, 16
```

The content of register $t1 (in hexadecimal) after executing the following sequence of instructions is <u>0x000c000b</u>.

```
la $t0, Array
lw $t1, 4($t0)
```

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

<u>slti $at, $s2, 11</u>
<u>bne $at, $0, Next</u>

(11) The pseudo instruction *li $t0, 0x12345678* is implemented by the following <u>minimum</u> MIPS instructions:

<u>lui $t0, 0x1234</u>

<u>ori $t0, $t0, 0x5678</u>

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

<u>srl $at, $s0, 28</u>

<u>sll $s0, $s0, 4</u>

<u>or $s0, $s0, $at</u>

(13) Assuming that $a0 contains an Alphabetic character, the instruction <u>andi $a0, $a0, 0xDF</u> will make the character stored in $a0 always upper case. 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 0x00400040 in the text segment, and the label NEXT is at address 0x00400028. Then, the value stored in the assembled instruction for the label NEXT is <u>(0x00400028-0x00400044)/4=FFF9</u>.

-

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

Array2: .half -2,-3, 4, 5

After executing the following sequence of instructions, the content of the two registers (in hexadecimal) is $t1=<u>000000FF</u> and $t2=<u>FFFFFFFD</u>.

```
la   $t0, Array2
lbu $t1, 1($t0)
lh   $t2, 2($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 <u>0x10010006</u> and <u>0x10010010</u>.

```
.data
X:      .byte  10, 11, 12, 13, 14
Y:      .half   15, 16, 17, 18
Z:      .word 19, 20
```

(17) To multiply the **signed** content of register $t0 by 112 without using multiplication instructions, we use the following minimum MIPS instructions (HINT: 112=16*7):

<u>sll $t1, $t0, 4</u>

<u>sll $t0, $t1, 3</u>

<u>sub $t0, $t0, $t1</u>

[11 Points]

(Q2) Answer each of the following questions. <u>Show how you obtained your answer</u>:

(i) Given that **TABLE** is defined as: **TABLE: .asciiz "Aiman El-Maleh"**

Determine the content of register **$t0** after executing the following code:

```
        xor $t0, $t0, $t0
        la $t1, TABLE
        li $t2, 'a'
Next:   lbu $t3, ($t1)
        beq $t3, $zero, ENL
        ori $t3, $t3, 0x20
        addi $t1, $t1, 1
        bne $t2, $t3, Next
        addi $t0, $t0, 1
        j Next
ENL:
```

The content of register $t0=3 as the program counts the number of characters equal to ‘A’ or ‘a’ in TABLE.

(ii) Determine the content of register $t1 after executing the following code:

```
            li $t0, 0x1234
            xor $t1, $t1, $t1
AGAIN:      andi $t2, $t0, 0xf
            add $t1, $t1, $t2
            srl $t0, $t0, 4
            bne $t0, $zero, AGAIN
```

The content of register $t1=0xA as the program computes the sum of the hexadecimal digits in register $t0.

(iii) Given that **TABLE** is defined as: **TABLE: .word 90, 70, 80, 60, 100**

Determine the content of register **$v0** after executing the following code:

```
            la      $a0, TABLE
            addi    $a1, $a0, 16
            lw      $v0, 0($a0)
      loop: addi    $a0, $a0, 4
            lw      $t1, 0($a0)
            bge     $t1, $v0, skip
            move    $v0, $t1
skip:       bne     $a0, $a1, loop
```

The content of register $v0=0x3C=60 as the program computes the minimum of the numbers stored in TABLE.

[17 points]

(Q3) 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) **[10 points]** Write a MIPS code fragment that returns the **maximum** integer value found in a user-specified row number of a **32** × **32** matrix **A** of 32-bit signed integers. The program should read the desired row number from the user and check that it is in the range between **0** and **31**. If not, the program should display the error message "**Row number is out of range.**" and terminate. Otherwise, the program should display the message "**Maximum integer in the row is** " and the value of the maximum integer found in the specified row, and then terminate. Assume that matrix **A** is already stored in memory.

```
.data
prompt:      .asciiz     "Please enter a row number between 0 and 31: "
outofrange:  .asciiz     "Row number is out of range.\n"
outmsg:      .asciiz     "Maximum integer in the row is "

.text
.globl main
main:
      la     $a0,prompt    # display prompt string
      li     $v0,4
      syscall
      li     $v0,5        # read row number into $t0
      syscall
      move   $t0,$v0
      bltz   $t0,error     # check row boundary
      addiu  $t1,$t0,-31   # If $t0 > 31, then result of ($t0-31) > 0
      bgtz   $t1,error
      la     $t1,A        # compute starting location of 1st element in desired row
      sll    $t2,$t0,5     # $t2 = i*32    (ixCOL+0)
      sll    $t2,$t2,2     # $t2 = i*32*4  (ixCOL+0)x(int size)
      addu   $t2,$t1,$t2   # $t2 = address of 1st element in desired row
      li     $t3,31        # max j = 31
      lw     $t4,0($t2)    # read 1st element of desired row & set as maximum
loop:
      addiu  $t2,$t2,4     # increment index to point to next row element
      lw     $t5,0($t2)    # read next element of desired row
      ble    $t5,$t4,next  # next element ($t5) <= current max ($t4)?
      move   $t4,$t5       # No -> set max ($t4) = next element ($t5)
next:
      addiu  $t3,$t3,-1    # prepare for next row element
      bgtz   $t3,loop
      la     $a0,outmsg    # display prompt string
      li     $v0,4
      syscall
      move   $a0,$t4      # output $t4 = maximum in desired row
      li     $v0,1
      syscall
      j      exit
error:
      la     $a0,outofrange
      li     $v0,4
      syscall
exit:
      li     $v0,10      # exit
      syscall
```

(ii) **[7 points]** Given two arrays **`A`** and **`B`**, write the smallest MIPS assembly fragment for the following computation. Assume that register **`$s0`** will be used to store **`cnt`** and assume that the following registers have the mentioned values: register **`$s1`** = number of elements, **`N`**, in each array, register **`$s2`** = base address of the array **`A`**, and register **`$s3`** = base address of the array **`B`**. Each array element is a 32-bit signed integer. Assume that **`N > 0`**. Insert comments to clarify the meaning of instructions and the use of registers.

```
int cnt = 0;
for (i=0; i != N; i++) {
   if (((A[i] - B[i]) > 5) || ((B[i] - A[i]) > 5)) cnt = cnt + 1;
}
```

```
      li    $s0,0           # $s0 = cnt = 0
loop:
      lw    $t0,0($s2)      # $t0 = A[i]
      lw    $t1,0($s3)      # $t1 = B[i]
      addiu $t2,$t0,5       # $t2 = A[i]+5
      addiu $t3,$t1,5       # $t3 = B[i]+5
      bgt   $t0,$t3,incr    # Check if (A[i]-B[i]>5)
      ble   $t1,$t2,done    # Check if (B[i]-A[i]>5)
incr:
      addiu $s0,$s0,1       # cnt++
done:
      addiu $s2,$s2,4       # point to A[i+1]
      addiu $s3,$s3,4       # point to B[i+1]
      addiu $s1,$s1,-1      # decrement loop index
      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}

Syscall Services:

Service	$v0	Arguments / Result
Print Integer	1	$a0 = integer value to print
Print Float	2	$f12 = float value to print
Print Double	3	$f12 = double value to print
Print String	4	$a0 = address of null-terminated string
Read Integer	5	Return integer value in $v0
Read Float	6	Return float value in $f0
Read Double	7	Return double value in $f0
Read String	8	$a0 = address of input buffer $a1 = maximum number of characters to read
Exit Program	10	
Print Char	11	$a0 = character to print
Read Char	12	Return character read in $v0