***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:

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

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

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

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

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

## 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 224=16 MB and the maximum number of bytes that can be read or written in a single cycle is 32/8=4.

## 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.

## 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.

## 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 0x000c000b.

la $t0, Array

lw $t1, 4($t0)

## The pseudo instruction *bgt $s2, 10, Next* is implemented by the following minimum MIPS instructions:

slti $at, $s2, 11

beq $at, $0, Next

## The pseudo instruction *li $t0, 0x12345678* is implemented by the following minimum MIPS instructions:

## lui $t0, 0x1234

## ori $t0, $t0, 0x5678

## 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 minimum MIPS instructions:

## srl $at, $s0, 28

## sll $s0, $s0, 4

## or $s0, $s0, $at

## Assuming that $a0 contains an Alphabetic character, the instruction andi $a0, $a0, 0xDF 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.

## 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 (0x00400028-0x00400044)/4=FFF9.

-

## 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=000000FF and $t2=FFFFFFFD.

la $t0, Array2

lbu $t1, 1($t0)

lh $t2, 2($t0)

## 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 0x10010006and 0x10010010.

## .data

## X: .byte 10, 11, 12, 13, 14

## Y: .half 15, 16, 17, 18

## Z: .word 19, 20

## 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):

## sll $t1, $t0, 4

## sll $t0, $t1, 3

## sub $t0, $t0, $t1

**[11 Points]**

# **(Q2) Answer each of the following questions. Show how you obtained your answer:**

## **(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.

## **[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**

## **[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:**

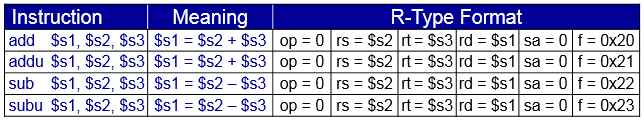
**addiu $s2,$s2,4 # point to A[i+1]**

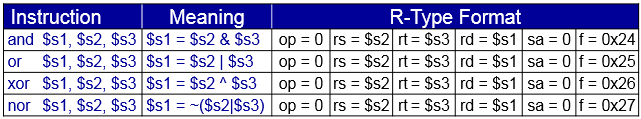
**addiu $s3,$s3,4 # point to B[i+1]**

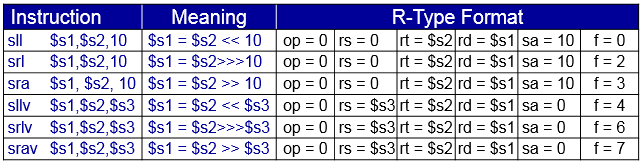
**addiu $s1,$s1,-1 # decrement loop index**

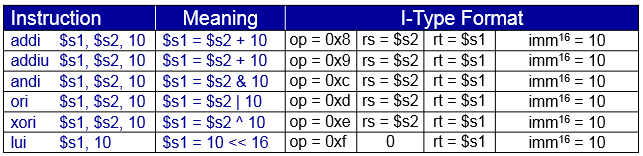
**bne $s1,$0,loop**

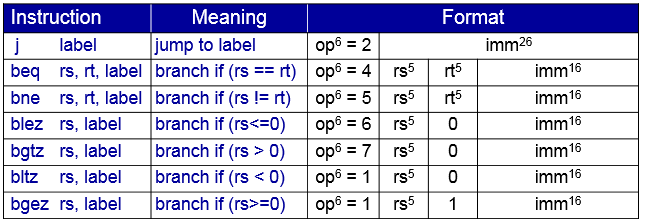
**MIPS Instructions:**

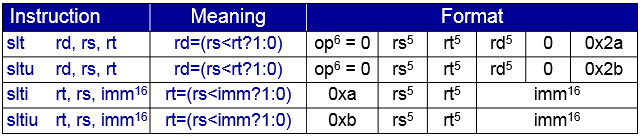


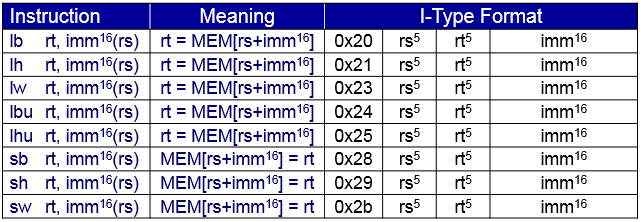












**Syscall Services:**

