***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:

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

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

## There is a one-to-one correspondence between assembly language and machine language.

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

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

## Dynamic RAM is slower than static RAM but is denser and cheaper.

## 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)

## 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 4.17 ms.

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

## addi $at, $s2, -1

slti $at, $at, 10

bne $at,$0, Next

OR

ori $at, $0, 10

slt $at, $at, $s2

beq $at, $0, Next

## 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 minimum MIPS instructions:

sll $at, $s0, 28

srl $s0, $s0, 4

or $s0, $s0, $at

## Assuming that $a0 contains an Alphabetic character, the instruction *xori $a0, $a0, 0x20* 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.

## 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 (0x00400014-0x00400034)/4=0xFFF8.

## 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)

## 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.

## [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

## [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

## [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:**













