Name: KEY **Id#**

ICS 233, Term 142

Computer Architecture & Assembly Language

Quiz# 2

Date: Tuesday, Feb. 24, 2015

Q1. Fill in the blank in each of the following questions:

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

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

After executing the following sequence of instructions, the content of the three registers is $t1=<u>00000003</u>, $t2=<u>fffffd03</u>, and $t3=<u>fc04fd03</u>.

```
la $t0, Array
lbu $t1, 4($t0)
lh $t2, 4($t0)
lw $t3, 4($t0)
```

(2) Assume that the instruction j NEXT is at address 0x0040002c in the text segment, and the label NEXT is at address 0x00400018. Then, the address stored in the assembled instruction for the label NEXT is <u>0x00400018/4=0x100006</u>.

(3) Assume that the instruction bne $t0, $t1, NEXT is at address 0x0040002c in the text segment, and the label NEXT is at address 0x00400018. Then, the address stored in the assembled instruction for the label NEXT is <u>(0x00400018-(0x0040002c+4))/4=(0x00400018-0x00400030)/4=0xffffffe8/4=0xfffa</u>.

(4) Assuming that \$a0 contains an Alphabetic character, the instruction *ori \$a0, \$a0, 0x20* will guarantee that the character in \$a0 is always a lower case character. Note that the ASCII code of character 'A' is 0x41 while that of character 'a' is 0x61.

(5) The pseudo instruction bge \$s2, \$s1, Next is implemented by the following minimum native MIPS instructions:

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

(6) To multiply the signed content of register \$t0 by 48.25 without using multiplications and division instructions, we use the following instructions:

```
sll $t1, $t0, 5
sll $t2, $t0, 4
addu $t1, $t1, $t2
sra $t2, $t0, 2
addu $t0, $t1, $t2
```

Q2. Write a MIPS assembly code fragment with minimum instructions to implement the following high level language code structure:

```
i = 0;
size = 10;
while (i < size && A[i] !=0){
     A[i] = A[i + 1] ;
     i = i + 1;
 }
```

Assume that the assembler has assigned `i` to register `$s0`, `size` to register `$s1`, and has stored the address of array `A` in register `$s2`. Assume that `A` is an array of integers.

```
            li $s0, 0
            li $s1, 10
While:      bge $s0, $s1, EndWhile
            lw $t0, 0($s2)
            beq $t0, $0, EndWhile
            lw $t1, 4($s2)
            sw $t1, 0($s2)
            addiu $s0, $s0, 1
            addiu $s2, $s2, 4
            j While
EndWhile:
```