CSCE 212: Exam 1 **Spring 2009**

Name (please print):_____ **Total points: ___/60**

Instructions
This is a CLOSED BOOK and CLOSED NOTES quiz. However, you may use calculators, scratch paper, and the green MIPS reference card from your textbook. Ask the instructor if you have any questions. Good luck!

1. (20 points) Write a short MIPS program that performs the following.

   The program reads 10 integers from the console and stores each value into an array called **vals**. After all the integers have been entered, the program prints them out in reverse order. Recall that system call 1 is **print_int** and system call 5 is **read_int**.

```
        .data
vals:   .space 4000
        .text
main:   li $s0,0
loop:   li $v0,5
        syscall
        sw $v0,vals($s0)
        addi $s0,$s0,4
        blt $s0,40,loop

        li $s0,36
loop2:  lw $a0,vals($s0)
        li $v0,1
        syscall
        addi $s0,$s0,-4
        bgez $s0,loop2
        jr $31
```

2. (10 points) Assume a new MIPS processor is released that includes a new load word instruction called **LW2** that allows **base-index** addressing instead of the usual base-offset addressing. As such, this instruction performs the following behavior: R[rd] <= memory[R[rs] + R[rt]]

   a. What encoding format should be used for this instruction and why?

      R-type, since it requires 3 registers (rs, rt, rd)

   b. To achieve backwards compatibility with older MIPS processors that didn't have this instruction, let's make **LW2** as a pseudo instruction. How would the following be translated into traditional MIPS hardware instructions, where $s1 is the base register and $s2 is the index register:

      LW2 $s0, 0($s1,$s2)

```
add $1,$s1,$s2
lw $s0,0($1)
```

3. (20 points) Write a MIPS subroutine that fulfills the following requirements:

Arguments: $a0 contains the address of an array of words
$a1 contains the size of the array

Return values: $v0 contains the sum of all the values in the array

The subroutine must not change any of the caller's registers (except for $v0), meaning that it must use the stack to save the original values of any registers that it uses.

```
sum:        addi $sp,$sp,-12      # save $s0, $s1, $s2
            sw $s0,0($sp)
            sw $s1,4($sp)
            sw $s2,8($sp)

            li $s0,0              # init index to 0
            li $v0,0              # init sum to 0
loop:       add $s1,$a0,$s0       # add index to address
            lw $s2,0($s1)         # load
            add $v0,$v0,$s2       # accumulate
            add $s0,$s0,4         # increment index
            blt $s0,$a1,loop

            lw $s0,0($sp)         # restore $s0, $s1, $s2
            lw $s1,4($sp)
            lw $s2,8($sp)
            addi $sp,$sp,12
            jr $31                # return to caller
```

4. (10 points) Convert the following sequence of machine instructions, represented as hexadecimal values, into assembly language.

```
8c900008
2210fff4
00108900
```

```
10001100100100000000000000001000
opcode=100011 => lw
rs=4, rt=16, imm=8
lw $16,8($4)

00100010000100001111111111110100
opcode=001000 => addi
rs=16, rt=16, imm=-0100 => 1011+1 => 1100 => 12
addi $16,$16,-12

00000000000100001000100100000000
opcode=000000, func=000000 => sll
rs=0, rt=16, rd=17, shamt=4
sll $17, $16, 4
```