CSCE 212: Exam 1 **Spring 2010**

Name (please print):_____ **Total points: ___/110**

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. (10 points) Translate the following Java statement into MIPS assembly code. Assume that x, y, z, q are stored in registers $s1-$s4. You may use the other registers to hold intermediate results.

```
x = x + y + z - q;
```

```
add $t0,$s1,$s2
add $t0,$t0,$s3
sub $s1,$t0,$s4
```

2. The MIPS instruction set includes several shift instructions. They include logical shift left, logical shift right, and arithmetic shift right.

   a. (10 points) Why doesn't MIPS offer an "arithmetic shift left" instruction?

   there's no sign bit on the right-hand side of a register to extend

   b. (10 points) How would you implement in assembly a shift-left- logical (SLL) pseudo-instruction for a machine that didn't have this particular instruction? Be sure your SLL instruction can shift up to ***W*** bits where ***W*** is the machine word size in bits. Assume the pseudo-instruction you're implementing for this particular answer is **SLL $s0,$s1,23**.

   Hint: To answer this question, you need to consider which arithmetic operation the left shift performs.

   *There are several ways to answer this question, but the simplest solution computes the output using a **loop** whose body includes a single **add** instruction which doubles the value in a register.*

```
        move $s0,$s1
        li $1,23
loop:   add $s0,$s0,$s0
        addi $1,$1,-1
        bgtz $1,loop
```

3. Use the register and memory values in the tables below for the next questions. Assume a 32-bit machine. Assume each of the following questions starts from the table values; that is, DO NOT use value changes from one question as propagating into future parts of the question.

Register	Value
$1	12
$2	16
$3	20
$4	24

Memory Address	Value
12	16
16	20
20	24
24	28

a. (10 points) Give the values of registers $1, $2, and $3 after this instruction is executed:

```
add $3, $2, $1
```

$3=28

b. (10 points) Give the values of registers $1 and $3 after this instruction is executed:

```
lw $3, 12($1)
```

$3=28

c. (10 points) Give the values of registers $2 and $3 after this instruction is executed:

```
addi $2, $3, 16
```

$2=36

4. (10 points) Suppose that there is a MIPS pseudo-instruction called **bcp rd, rs, rt**, that copies a block of words from one address to another. The starting address of the source block is in register **rs**, the starting address of the destination block is in **rd**, and the number of words to copy is in **rt**.

Assume that the values of these registers as well as register $1 can be destroyed in executing this instruction (so that the registers can be used as temporaries to execute the instruction). Write the MIPS assembly code to implement the instruction:

```
bcp $s0, $s1, $s2
```

I have provided the skeleton for the answer below. Fill it in.

```
loop:   lw $1,0($s1)        # load a word from the source area

        sw $1,0($s0)        # store it to the destination area
        addi $s1,$s1,4      # increment base address for source area

        addi $s0,$s0,4      # increment base address for destination area
        addi $s2,$s2,-1     # decrement count

        bgtz $s2,loop       # repeat loop body if count > 0
```

5. (10 points) In MIPS programs, what is the purpose of the stack?

   preserve register values across subroutine calls, so the callee doesn't destroy intermediate values being used by the caller

6. (10 points) Write the sequence of instructions needed to push registers **$ra** and **$s0** onto the stack.

```
addi $sp,$sp,-8
sw $ra,0($sp)
sw $s0,4($sp)
```

7. (10 points) Assemble the following assembly instruction using hexadecimal representation:

```
sb $s0,10($s1)
```

opcode	rs	rt	imm
28 hex=101000	s1=17=10001	s0=16=10000	10=000A hex

1010 0010 0011 0000 binary + 000A hex
A230000A

8. (10 points) Translate the following loop from Java to MIPS assembly language:

```
for (i=0;i<10;i++) vals[i]=i;
            li $s0,0 # i
    loop:   bge $s0,10,exit
            sll $s1,$s0,1
            sw $s0,vals($s1)
            addi $s0,$s0,1
            j loop
    exit:
```