CSE 30321 – Computer Architecture I – Fall 2011
Homework 03 – The MIPS ISA – 75 points
Assigned: September 13, 2011 **– Due:** September 20, 2011

Problem 1: (25 points)

The C-code shown below will loop through an array of grades (integer values between 0 and 100) and calculate counts for a histogram.

```
int A, B, C, D, F;          // declare grade bins
int i;                      // declare loop counter
int N;                      // # of grades to analyze

A=0; B=0; C=0; D=0; F=0;    // initialize grade counters

…                           // read in test scores into grades array
                            // N will be equal to the number of grades
                            // assume N = 60

for (i=0; i<N; i++) {
      if(grades[i] >= 90)
            A = A + 1;
      else if(grades[i] >= 80)
            B = B + 1;
      else if(grades[i] >= 70)
            C = C + 1;
      else if(grades[i] >= 60)
            D = D + 1;
      else
            F = F + 1;
}
```

Write the MIPS assembly for the C-code shown above.

You can / should assume the following:

- The number of grades to be analyzed (N) is 60.
- The address of the first element in the *grades* array is contained in $s0.

Additionally, to make it easier for you to keep track of register assignments, you may use register names that have the same name as the variable names used in the code above. For example, to initialize the register that holds the *A* counter, you might write:

```
add     $A, $0, $0
```

However, please note that for your lab, you will need to use actual register names in order to use the XSPIM tool (which will allow you to dynamically step through MIPS instructions). Thus, you should consider this when you choose to answer this question. If you do chose to use actual MIPS register names, you should write your code out as follows (i.e. with appropriate comments):

```
add     $t0, $0, $0             # $t0 maps to variable A; we need to initialize it to 0
```

Problem 2: (35 points)

Part A: (25 points)

Write an assembly language program using MIPS instructions that performs the same function as the following piece of C code:

```
int i, j, M, N, count;
int a[];
int b[];

for(i=0; i<M; i++) {
      for(j=0; j<N; j++) {
            if((a[i] > 0) && (b[j] > 0)) {
                  count = count + 1;
            }
      }
}
```

Again, if you wish, you may use register names that are the same as the variable names used in the code above. For example, to initialize the register that holds the *i* counter, you might write:

```
add     $I, $0, $0
```

If you do chose to use actual MIPS register names, you should write your code out as follows:

```
add     $t0, $0, $0             # $t0 maps to variable A; we need to initialize it to 0
```

(i.e. be sure to add appropriate comments)

Assume that the starting addresses of the a[] and b[] arrays are in registers $s1 and $s2 respectively.

Part B (10 points):
How many instructions will be executed for your nested loops?

- Hint 1: Don't forget about the instruction fetches!
- Hint 2: Write a parameterized expression as a function of M, N, etc.
- Hint 3: Answers will vary based on the code that you right

Bonus (5 points):
The assembly code that I wrote for Part A required 21 MIPS instructions … including instructions to initialize different variables, etc. 5 extra credit points will be awarded if your answer to Part A can beat that (in terms of code size).

Problem 3: (15 points)

Given the instructions below, describe what you think this code does – i.e. what programming construct and/or operation does this code most closely match.

```
        addi    $10, $0, 0
L1:     beq     $8, $0, L2
        addi    $10, $10, 1
        lw      $8, 4($8)
        j       L1
L2:
```