CSE 30321 – Computer Architecture I – Fall 2010
Homework 03 – The MIPS ISA – 100 points
Assigned: September 14, 2010 **– Due:** September 21, 2010

Problem 1: (35 points)

Consider the following fragment of C code:

```
for (i = 0; i <= 100; i = i + 1)
        a[i] = b[i] + c[i];
```

Part A (25 points):

Assume that *a*, *b*, and *c* are arrays of 32-bit MIPS integer data words and:

- The base address of *a* is in $a0
- The base address of *b* is in $a1
- The base address of *c* is in $a2.
- Register $t0 is associated with variable *i*.

Write MIPS code that can be used to implement this loop.

Part B (10 points):

How many memory references will be made during execution?

- Hint: Don't forget about the instruction fetches!

Problem 2: (50 points)

Part A: (35 points)

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

```
int a[n][m];
acc = 0;
for (i = 0;  i < n; i++) {
        for (j = 0; j < m; j++) {
                acc += a[i][j];
        }
}
```

Assume that the first element of the array is stored at memory location A_{start}, which is stored in $s4. The array is stored in the "row major" form, i.e., all the elements are stored in successive memory locations in the following order:

```
a[0][0],        ..., a[0][m-1],
a[1][0],        ..., a[1][m-1],
                ...
a[n-1][0],      ..., a[n-1][m-1]
```

(Thus, a[0][0] is in the first memory location, a[0][1] is in the second, etc.)

Also, assume that *n* and *m* are stored in registers $a0 and $a1, respectively.

Part B: (15 points)

The different classes of MIPS instructions can take different numbers of clock cycles to execute (refer to the table below). Using your answer to part A, calculate the execution time for your code if the clock rate is 1 GHz. You should assume that *n* and *m* are both 7.

(Note that extra credit will be given for a correct solution that offers the lowest overall execution time.)

R type (add, sll, etc.)	4 CCs
I type (addi, subi, etc.)	4 CCs
I type (load)	5 CCs
I type (store)	4 CCs
I type (branch)	3 CCs
J type (jump)	3 CCs

<u>Problem</u> <u>3</u>: (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:
```