ICS 233 – Spring 2009
Computer Architecture and Assembly Language
Programming Assignment 3

Matrix Multiplication and Counting Instruction Frequencies

Write and test a MIPS assembly language program to perform matrix multiplication of *N* by *N* matrices of double-precision floating-point numbers. In your program, define the space of three 10×10 matrices, where the maximum value of *N* is fixed at 10.

The matrix data should be read from a text file in row-major order. *N* × *N* signed numbers should be read from a text file. Prompt the user to enter the name of the text file, then open and read the text file. Each number should be converted from a string to a floating-point number. You need a procedure to convert the input string character-by-character to a floating-point number. The input string begins with an optional '-' sign, followed by one or more decimal digit characters from '0' to '9', followed by an optional decimal point '.' and optional fraction digit characters.

Write a procedure to do matrix multiplication of *N* × *N* matrices, where *N* is passed to the procedure as a parameter. *All matrix operations should be done using the double-precision floating-point instructions*. The output matrix should be displayed on the screen. Test and verify the matrix multiply procedure, by examining the result matrix output.

After succeeding in matrix multiplication and producing the correct result, you will analyze the MIPS code of the matrix multiply procedure, to have a better understanding of instruction frequencies. *You will count the dynamic number of instructions that are executed at runtime to determine their frequencies in the matrix multiply procedure*. You need a total of four counters to count instructions for the following classes of instructions:

- Class 1 is for ALU instructions
- Class 2 is for floating-point instructions
- Class 3 is for load and store instructions
- Class 4 is for branch and jump instructions

You will *augment the code of the matrix multiply procedure with additional instructions* to count the original number of instructions. You need four counters. At the beginning of the procedure, initialize all counters to zeros. Before each instruction, insert additional instructions to count that instruction. For example, if the original instruction is **addiu** then increment the counter of ALU instructions before the instruction itself. If the same instruction is executed 100 times (in different loop iterations), it will be counted as 100. Count the basic instructions. For pseudo-instructions, count the equivalent basic instructions. Make sure that your additional code does not interfere with the original program code. Count only the original instructions of the matrix multiply procedure, not the new ones that you have added. Display the statistics that you have produced. A sample run is show below:

```
Enter the matrix size N: 5
Enter the first matrix filename: m1.txt
Enter the second matrix filename: m2.txt
```

```
Output matrix:
<Display here the output matrix, row by row>

Counting Instructions in the Matrix Multiply Procedure:
Total               instructions = ???
ALU                 instructions = ??, Percentage = ?%
Floating-Point      instructions = ??, Percentage = ?%
Load & Store        instructions = ??, Percentage = ?%
Branch & Jump       instructions = ??, Percentage = ?%
```

Submission Guidelines:

All submissions will be done through WebCT.

Submit the source code of the program. Make sure that your program is well documented.

Grading Policy:

The grade will be divided according to the following components:

- Input is read properly from the text file. Converting an input string to a floating point number is done properly.
- Matrix multiply procedure works properly and produces correct results.
- Instruction counting is done properly and produces correct counts and percentages.
- Design and Coding: program is well designed and divided into procedures
- Documentation of code: program is well documented

Late Policy:

The programming assignment should be submitted on the due date by midnight. Late submissions are accepted, but will be penalized 5% for each late day and for a maximum of 3 late days (or 15%). Assignments submitted after 3 late days will not be accepted.