EECS 645
Fall 2014

Programming Project

due: December 1, 2014

Implement the following system of tracing of the MIPS-64 pipeline using any programming language you wish. **Your project should be implemented on one of the departmental computers, e.g., cycle 1, under Linux**.

Input data files should start from the list of initial contents of integer registers. This list starts from the word I-REGISTERS, then comes a sequence of lines containing the pairs of register names and integers, separated by spaces, e.g., R5 42. All possible register names are: R0, R1,..., R31. The list of the pairs of register names and integers may be empty, but the file should contain the word I-REGISTERS.

The second part of the input data file starts from the word FP-REGISTERS containing the pairs of register names and FP numbers, separated by spaces, e.g., F5 -12.56. All possible register names are: F0, F1,..., F31. The list of the pairs of register names and integers may be empty, but the file should contain the word FP-REGISTERS.

The third part of the input data file starts from the word MEMORY, then comes a sequence of lines containing the pairs of integer and FP numbers separated by spaces, e.g., 40 23.79. The first integer is the number of the memory location, the second is its contents. All possible memory location numbers are: 0, 8,..., 992. Again, the list of pairs may be empty, but the file should contain the word MEMORY.

The following part starts from the word CODE, followed by a sequence of lines, each line containing a single unlabeled instruction. We will use only five FP operations: two data transfers: FP load (e.g., `L.D F2, 0(R1)`), FP store (e.g., `S.D 8(R3), F4`); FP addition (e.g., `ADD.D F1, F2, F3`), FP subtraction (e.g., `SUB.D F8, F9, F10`) and FP multiplication (e.g., `MUL.D F4, F6, F8`). L.D and S.D are executed on a five stage pipeline: IF, ID, EXE, MEM, WB. The ADD.D and SUB.D operations are executed on a eight-stage pipeline: IF, ID, A1, A2, A3, A4, MEM, WB while MUL.D is executed on an eleven-stage pipeline: IF, ID, M1, M2, M3, M4, M5, M6, M7, MEM, WB. Instructions may write to and read from the registers at the same clock cycle (first write, then read). For stalled instructions, insert appropriate number of "stall" symbols immediately after IF. Assume forwarding, out of order completion of instructions (if it not causes any hazard) and only one port to access FP registers.

Spaces should be understood not only as ordinary spaces but also as white space characters, such as the end of line, tab, etc. Any line of the input data file may start from one or more spaces and one or more spaces may end it. For simplicity, initial contents of all registers, not listed after the word I-REGISTERS or F-REGISTERS, and initial contents of all memory locations, not listed after the word MEMORY, are all equal to zero.

First your program should ask the user for the name of the input data file. The expected response of the user is the name followed by pressing the <RETURN> key. Your program should provide the timing of the instruction sequence and final contents of all registers appearing in the CODE.

Then your program should ask the user for the name of the *timing* file, in which information about timing will be stored. Such timing should show cycle numbers in rows, while columns should be labeled with instruction numbers, e.g.,

```
         I#1   I#2   I#3 ...
c#1       IF
c#2       ID    IF
c#3       EXE   s
```

You may assume that the input data file does not contain any errors.

Finally, the program should ask the user for the name of the *register* file, in which information about contents of the FP registers will be stored, e.g.,

```
F1     F2     F3
71.2   17.8   4
```

General Remarks. Include all comments, including instructions about compiling and linking, in a single file called read.me. Do not forget to include your name and KUID#. When you are ready to submit the project, send ALL necessary source files, makefile (if any), and the read.me file by e-mail to the TA**. Do not send object files, executable files, and test data files**. Late projects will be accepted with 10% penalty per day up to five days.