COMPSCI 542 - Spring 2009
Assignment 1
Detecting Pipeline Hazards

1. PROJECT DESCRIPTION

Your assignment is to write a program that reads in a trace of MIPS instructions to be executed and determines for each instruction the exact cycle(s) that each pipeline stage is initiated. The instructions will be read in from standard input. The report you generate should be printed to standard output.

You can assume the following distinct MIPS pipeline stages:

IF - instruction fetch
ID - instruction decode
EX - execute (integer instructions, data movement and conversion, loads and stores)
MEM - memory access
WB - integer write back
FADD - floating-point adder (FP adds and subtracts)
MUL - floating-point multiplies
DIV - floating-point divides
FWB - floating-point write back

Note that the type of write-back stage used depends upon the type of data register being assigned a value. You should read in the cycles required in the execute stage for the floating-point operations from a file called *config.txt* in the current directory. You can assume the cycles required in the integer execute (EX) is one. Note that there is NO pipelining of the individual FP operations, as depicted in Figure A.31 of your text.

The MIPS instructions you have to recognize are a subset of those listed on the inside of the back-cover of your text ("subset of the instructions in MIPS64") The instructions you need to recognize are:

- loads and stores (LW, SW, L.S, and S.S)
- single cycle arithmetic operations (DADD, DSUB, AND, OR, XOR)
- branches (BEQ, BNE)
- floating-point arithmetic operations (ADD.S, SUB.S, MUL.S, DIV.S)
- data movement (MOV.S, MFC1, MTC1)
- data conversion (CVT.S.W, CVT.W.S)

In addition, a ":T" or ":F" will follow a branch instruction to indicate if it was taken or if it fell through, respectively.

WAW data hazards should be handled by squashing the writeback of the previously executed instruction (i.e. the instruction never enters the WB stage). Stores leave the pipeline after the MEM stage. Multicycle instructions (floating-point arithmetic operations) do not enter the MEM stage. You may assume there is a maximum of 100 instructions in the trace. You may also assume that the instructions in the input are valid.

A sample report is given at the end of this assignment. You should first generate a numbered list of instructions. You should next produce a table that indicates for each cycle the number of the instruction each pipeline stage is processing (if any). The detection of all hazards and resolution by stalls should occur in the ID stage, even when the hazard may occur in a later stage of the pipeline. You should indicate an instruction that is stalled by placing the word stall in the appropriate entry of the table. After the table, you should print associated with each type of hazard the number of stalled cycles attributed to that hazard, the percentage of totalstalls, and the percentage of total cycles caused by the stalls. If an instruction could stall for more than one reason, then give priority to attributing the stall to first structural hazards, next load hazards, and then finally data hazards. In addition, you should have a summary line that indicates the total stalls and percentage of total cycles that were stalls. Finally, you should print the number of times that a result was squashed due to a WAW hazard and the number of times that an instruction was flushed due to a taken branch.

2. PROJECT SUBMISSION

You should submit the source code for your program before class starts on Feb 17th, 2009 with "submit uh cs542 pipe" utility on onyx. The header comments should identify you, the assignment, and the command you used to compile your program. You should also use readable and consistent indentation and comments throughoutthe program. Your code should be able to compile and execute on *program*.

3. EXAMPLE

The following pages contain an example of input to your program and the corresponding output. The trace of instructions that were used as input is in the file `~uh/cs542exec/trace.dat`. The instructions should match the form described in Appendix A. The configuration file used is in `~uh/cs542exec/config.txt`. The portion of the configuration file that can change are the cycles, which must be positive integer values. You can check your execution with mine using the executable uh/cs542exec/pipe.exe. You should make your output exactly match mine so I can check for correctness using the *diff* command.

3.1 configuration ("config.dat")

```
fp_add: 4
fp_mul: 7
fp_div: 11
```

3.2 trace data ("trace.dat")

```
LW      R1,32(R6)
DADD    R4,R1,R7
SW      R5,16(R6)
DADD    R6,R1,R7
BEQ     R6,R2,L5:F
ADD.S   F12,F14,F16
ADD.S   F0,F2,F4
DIV.S   F6,F0,F8
CVT.S.W F2,F0
MFC1    R6,F2
SUB.S   F6,F14,F16
```

3.3 output

```
Configuration:
   fp adds and subs cycles:  4
      fp multiplies cycles:  7
         fp divides cycles: 11


Instructions:
  1. LW      R1,32(R6)
  2. DADD    R4,R1,R7
  3. SW      R5,16(R6)
  4. DADD    R6,R1,R7
  5. BEQ     R6,R2,L5:F
  6. ADD.S   F12,F14,F16
  7. ADD.S   F0,F2,F4
  8. DIV.S   F6,F0,F8
  9. CVT.S.W F2,F0
 10. MFC1    R6,F2
 11. SUB.S   F6,F14,F16


cycle    IF    ID    EX   MEM    WB  FADD  FMUL  FDIV   FWB
----- ----- ----- ----- ----- ----- ----- ----- ----- -----
    1     1
    2     2     1
    3     3     2     1
    4 stall stall           1
    5     4     3     2           1
    6     5     4     3     2
    7     6     5     4     3     2
    8 stall stall           4
    9     7     6                 4
   10     8     7                       6
   11 stall stall                       6
   12 stall stall                       6
   13 stall stall                       6
   14     9     8                       7                 6
   15 stall stall                       7
   16 stall stall                       7
   17 stall stall                       7
   18    10     9                                   8     7
   19    11    10     9                             8
   20          11    10     9                       8
   21                      10          11           8     9
   22                            10    11           8
   23                                  11           8
   24                                  11           8
   25                                               8    11
   26                                               8
   27                                               8
   28                                               8

hazard type  cycles  % of stalls  % of total
-----------  ------  -----------  ----------
load-delay        1        12.50        3.57
structural        3        37.50       10.71
data              4        50.00       14.29
-----------  ------  -----------  ----------
```

```
total             8       100.00       28.57

WAW squashes: 1
branch flushes: 0
```