COMP 740 Homework 2

Spring 2009

Due Thursday, Mar 3, 2009

The book has sets of problems that are useful for practice. The solutions are on the web at:
http://www.elsevierdirect.com/companions/9780123704900/casestudies/Case%20Study%20Solutions.pdf

or at: http://tinyurl.com/ad8eta

Please be sure to explain your solutions.

1. (50 points) *MIPS Integer Pipeline: Hazards, Stalls and Forwarding.* Consider the following program fragment to compute values for variable D. All variables are memory-resident, integer-valued, and located at distinct memory addresses.

```
D = (A*X+B)*X+C;
```

A compiler for a MIPS machine generates the following code for this program fragment: (The memory addresses are symbolic names.)

```
1:  LW R1, X
2:  LW R2, A
3:  MULT R10, R1, R2
4:  LW R3, B
5:  ADD R10, R10, R3
6:  MULT R10, R10, R1
7:  LW R4, C
8:  ADD R10, R10, R4
9:  SW D, R10
```

Note: Make sure that the code fragment actually does compute `D` correctly; if not, fix it.

(a) Enumerate all of the data hazards that exist among the instructions. Classify them as RAW, WAR, and WAW hazards.

(b) Assume a five-stage linear pipelined implementation of MIPS with the two additional assumptions (both admittedly unrealistic): (i) all operations spend a single cycle in the EX stage, and (ii) there are no forwarding paths whatsoever.[1] Represent the execution of the program above with the help of a space-time chart (i.e., the spreadsheets we have been using in class), and determine the number of clock cycles it will take to complete execution. Clearly indicate all stalls.

(c) Still assuming no forwarding, *rearrange* the order of instructions in this program to reduce execution time without violating any dependencies. Draw a space-time chart to represent the execution of the rearranged program.

(d) Now assume that all reasonable forwarding paths are available. Draw a space-time diagram for the execution of the original program.

(e) Still assuming that all reasonable forwarding paths are available, draw a space-time diagram for the execution of the rearranged program.

(f) Summarize the execution times of the four program executions in a two-dimensional table. Label the y-axis with the lack (or availability) of forwarding, and the x-axis with the lack (or availability) of compile-time scheduling.

(over)

[1] Note, however, that a combination of a register write and a register read in the same clock cycle "forwards" through the register file, because the latter occurs on a later clock edge.

2. (50 points) *Scoreboarding.* We will look at a dynamically scheduled version of the DAXPY loop, the double precision a * x + y. You may have heard of this as SAXPY (single precision). It is a common loop in computations such as Gaussian elimination. Specifically, we'll compute y[i] = a * x[i] + y[i]. The MIPS code for DAXPY is:

```
loop: L.D F2, 0(R1)           ; load x[i]
      MULT.D F4, F2, F0       ; a * x[i]
      L.D F6, 0(R2)           ; load y[i]
      ADD.D F6, F4, F6        ; add y[i]
      S.D 0(R2), F6           ; store result into y[i]
      DADDUI R1, R1, #8       ; increment x index
      DADDUI R2, R2, #8       ; increment y index
      DSGTUI R3, R1, done     ; test if at end (R1 == done?)
      BEQZ R3, loop           ; branch if not done
```

For this problem you'll use a pipeline like that described in Section A.5 of the textbook. Assume that integer operations issue and execute in one clock, including loads, and their results are fully forwarded. For the functional units, use the latencies and initiation intervals shown in Figure A.30 (page A-49). State explicitly any other assumptions you make. For example, "as soon as an instruction enters WB, one that is waiting for the functional unit can issue."

Note: Scheduling will be of only one basic block; do not schedule the branch instruction or subsequent iterations.

(a) Show the number of stall cycles and at which clock cycle each instruction begins execution.

(b) How many clock cycles does an execution of the loop take?

(c) Show the state of the scoreboard tables (as in Figure A.52) when the SGTI instruction reaches WR. Assume that issue and read operands each take a cycle, and that there is a single integer functional unit that takes only a single execution cycle. The branch instruction should not go on the scoreboard.