

CMSC 411 – Spring 2014

Homework #2 – Unit 3 - Pipelining

1. This question is the same as the first 3 parts of question 1 from Appendix C in the book.

Use the following code fragment:

```
Loop: LD      R1,0(R2)    ; load R1 from address 0+R2
      DADDI   R1,R1,#1    ; R1=R1+1
      SD      R1, 0(R2)   ; store R1 at address 0+R2
      DADDI   R2,R2,#4    ; R2=R2+4
      DSUB    R4,R3,R2    ; R4=R3-R2
      BNEZ    R4,Loop     ; branch to Loop if R4!=0
```

Assume that the initial value of R3 is $R2 + 396$.

Use the classic MIPS five-stage integer pipeline (see Figure C.1 in Hennessy and Patterson) and assume all memory accesses take 1 clock cycle.

- Data hazards are caused by data dependences in the code. Whether a dependency causes a hazard depends on the machine implementation (i.e., number of pipeline stages). List all of the data dependences in the code above. Record the register, source instruction, and destination instruction; for example, there is a data dependency for register R1 from the LD to the DADDI.
 - Show the timing of this instruction sequence for the MIPS pipeline *without* any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle “forwards” through the register file, as in Figure C.6. Use a pipeline timing chart like Figure C.5. Assume that the branch is handled by flushing the pipeline. If all memory references take 1 cycle, how many cycles does this loop take to execute?
 - Show the timing of this instruction sequence for the MIPS pipeline with normal forwarding and bypassing hardware. Use a pipeline timing chart like Figure C.5. Assume that the branch is handled by predicting it as not taken. If all memory references take 1 cycle, how many cycles does this loop take to execute?
2. Scheduling branch delay slots (see the three ways in Figure C.14) can improve performance. Assume a single branch delay slot and an instruction pipeline that determines branch outcome in the second stage.

- a. For a delayed branch instruction, what is the penalty for each branch delay slot scheduling scheme if the branch is taken and if it is not taken, and what condition, if any, must be satisfied to ensure correct execution?
 - b. A cancel-if-not-taken branch instruction (also called *branch likely* and implemented in MIPS) does not execute the instruction in the delay slot if the branch is not taken. Thus, a compiler need not be as conservative when filling the delay slot. For each branch delay slot scheduling scheme, what is the penalty if the branch is taken and if it is not taken, and what condition, if any, must be satisfied to ensure correct execution?
 - c. Assume that an instruction set has a delayed branch and a cancel-if-not-taken branch. When should a compiler use each branch instruction and from where should the slot be filled?
3. This question is the same as question 11 from Appendix C in the book. Construct a table like Figure C.25 to check for WAW stalls in the MIPS FP pipeline of Figure C.35. Do not consider FP divides.
4. Suppose that in the original MIPS implementation (no pipeline), the stages to execute an instruction could run this fast:

IF	15ns
ID	8ns
EX	7ns
MEM	15ns
WB	5ns

- a. How long, in nanoseconds, would it take to complete 100 instructions, consisting of 20 loads, 20 stores, 10 branches, and 50 register-register ALU instructions? Assume only loads and stores use the MEM stage to execute.
 - b. Suppose MIPS was implemented on a 5-stage pipeline with the same stages as above. If there are no stalls in the pipeline, how many nanoseconds would it take to complete the same 100 instructions?
 - c. Suppose that half of the loads cause a 3 cycle delay in the pipeline. How many nanoseconds would it take to complete the 100 instructions?