CSC 211.02 **Assignment 10: Pipelining and Hazards** **Fall 2015**

Assigned: Tuesday 17 November

Due: Monday 23 November 10:30 pm

Collaboration: This assignment must be completed independently. Answers may only be discussed with the instructor.

Submission: Upload a PDF with your name, box number, and all solutions to PioneerWeb.

Problem 1

Consider the pipelined MIPS processor developed in chapter five of the textbook (e.g. Figure 4.65). Assume register `$1` holds the value `5` and register `$2` has the value `3` when executing the instruction

```
add $10, $1, $2
```

a) Which pipeline registers hold the value `8`, the result of the addition, at some point during the execution of this instruction?

b) Which pipeline registers hold the value of the `MemWrite` control line, which enables writes to data memory?

c) If this processor does not support forwarding, how many bubbles must be added to the pipeline before this processor can execute the next instruction:

```
add $1, $10, $3,
```

d) If the processor does support forwarding, how many bubbles must be added to the pipeline before this processor can execute the next instruction ?

Problem 2

Each pipeline stage has some latency, and passing a value through each pipeline register adds some additional latency. Answer the following questions using the latency values below, assuming the five-stage MIPS pipeline developed in chapter 5 of the textbook.

IF	ID	EX	MEM	WB	Pipeline Registers
250ps	200ps	150ps	180ps	180ps	20ps

a) What is the latency of a `lw` instruction in a single cycle (non-pipelined) processor with these latencies?

b) What is the minimum clock cycle time in a single cycle processor with these latencies?

c) What is the latency of a `lw` instruction in a pipelined processor with these latencies?

d) What is the minimum clock cycle time for a pipelined processor with these latencies?

e) Ignoring hazards, what is the speed-up achieved by introducing pipelining?

f) Suppose you could split one stage into two new stages, each with half the latency of the original stage. Explain which stage would you split; what is the new minimum clock cycle time of the processor?

Problem 3

Consider the following sequence of instructions executed on the five-stage pipelined datapath of chapter 5:

```
lw  $1, 40($6)
add $2, $3, $1
add $1, $2, $6
sw  $2, 20($4)
and $1, $1, $4
```

a) Assuming there is no forwarding, insert the minium number of `nop` instructions in this sequence to eliminate all data hazards.

b) Draw a diagram of the execution of this instruction sequence *with forwarding*, as in Figure 4.59 in the text. Using a monospaced font you may type your "diagram" with the form

```
IF -> ID -> EX -> MEM -> WB
```

ensuring all clock cycles line up in the same columns.

Problem 4

Consider the following sequences of branch outcomes. T means the branch is taken; N means the branch is not taken.

Sequence 1: T, T, T, N

Sequence 2: T, N, T, T, N

a) What is the accuracy of the always-taken branch predictor for each sequence?

b) What is the accuracy of the always-not-taken branch predictor for each sequence?

c) What is the accuracy of the 2-bit branch predictor (Figure 4.63) for each sequence? Assume the predictor starts in the "weak" not taken state (lower-right).

d) What is the accuracy of the 2-bit branch predictor for each sequence, if the sequence repeats thousands of times?

e) Write a new sequence of six branch outcomes that a 2-bit branch predictor will predict with 100% accuracy, assuming the predictor starts in the "weak" not taken state (lower-right).

f) Write a new sequence of six branch outcomes that a 2-bit branch predictor will predict with 0% accuracy, assuming the predictor starts in the "weak" not taken state (lower-right).

Problem 5

In this exercise we compare the performance of 1-issue and 2-issue processors, taking into account program transformations that can be made to optimize for 2-issue execution on the following loop.

```
loop:  lw   $t0, 0($s0)
       addi $s0, $s0, 4
       sw   $t0, 0($s0)
       addi $s0, $s0, 4
       bne  $s0, $s1, loop
       nop
```

a) Assuming the processor has perfect branch prediction, give the schedule for two iterations of this loop on the static two-issue datapath (Figure 4.69).

b) Unroll this loop so each iteration performs four iterations of the original loop and rearrange instructions to improve the performance on the same static two-issue datapath.

c) Draw a pipelined execution diagram (as in Problem 3B) for your revised code on a static two-issue datapath. Show two iterations of the unrolled loop, assuming the processor has perfect branch prediction.

Acknowledgments This derivative work of Janet Davis, used under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License was developed in collaboration with Charlie Curtsinger. Problems 4–5 are adapted from Patterson & Hennessy (2008), Exercises 4.24 and 4.28 (pp. 432, 436).[1]

[1] Patterson, D. A., & Hennessy, J. L. (2008). *Computer organization and design: The hardware/software interface* (Fourth Edition). Morgan Kaufmann: Burlington, MA.