Chapter 4

Exercises

4.16 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250 ps	350 ps	150 ps	300 ps	200 ps

Also, assume that instructions executed by the processor are broken down as follows:

ALU/Logic	Jump/Branch	Load	Store
45%	20%	20%	15%

4.16.1 [5] <§4.6> What is the clock cycle time in a pipelined and non-pipelined processor?

4.16.2 [10] <§4.6> What is the total latency of an `lw` instruction in a pipelined and non-pipelined processor?

4.16.3 [10] <§4.6> If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

4.16.4 [10] <§4.6> Assuming there are no stalls or hazards, what is the utilization of the data memory?

4.16.5 [10] <§4.6> Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

4.28 The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-type	beqz/bnez	jal	lw	sw
40%	25%	5%	25%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

4.28.1 [10] <§4.9> Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the ID stage and applied in the EX stage that there are no data hazards, and that no delay slots are used.

4.28.2 [10] <§4.9> Repeat 4.28.1 for the "always-not-taken" predictor.

4.28.3 [10] <§4.9> Repeat 4.28.1 for the 2-bit predictor.

4.29 This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT.

4.29.1 [5] <§4.9> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

4.29.2 [5] <§4.9> What is the accuracy of the 2-bit predictor for the first four branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.78 (predict not taken)?