EECS 361

HW 3

1 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**
300 ps	450 ps	200 ps	400 ps	200 ps

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

Alu	Beq	Lw	Sw
60%	15%	15%	10%

1a. What is the clock cycle time in a pipelined and non-pipelined processor?
1b. What is the total latency of an LW instruction in a pipelined and nonpipelined processor?
1c. 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?
1d. Assuming there are no stalls or hazards, what is the utilization of the data memory?
1e. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the “Registers” unit?
1f. Instead of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction finishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g., ST only

takes 4 cycles because it does not need the WB stage). Compare clock cycle times and execution times with single-cycle, multi-cycle, and pipelined organization.

2. In this exercise, we examine how data dependences affect execution in the basic 5-stage pipeline described in Section 4.5. Problems in this exercise refer to the following sequence of instructions:

```
or r1, r2, r3
or r2, r1, r4
or r1, r1, r2
```

Also, assume the following cycle times for each of the options related to forwarding:

Without Forwarding	**With Full Forwarding**	**With ALU-ALU Forwarding Only**
300 ps	350 ps	340 ps

2a. Indicate dependences and their type.
2b. Assume there is no forwarding in this pipelined processor. Indicate hazards and add `nop` instructions to eliminate them.
2c. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them.
2d. What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?
2e. Add `nop` instructions to this code to eliminate hazards if there is ALUALU forwarding only (no forwarding from the MEM to the EX stage).
2f. What is the total execution time of this instruction sequence with only

ALU-ALU forwarding? What is the speedup over a no-forwarding pipeline?

3. This exercise is intended to help you understand the cost/complexity/performance trade-offs of forwarding in a pipelined processor. Problems in this exercise refer to pipelined datapaths from Figure 4.45. These problems assume that, of all the instructions executed in a processor, the following fraction of these instructions have a particular type of RAW data dependence. The type of RAW data dependence is identified by the stage that produces the result (EX or MEM) and the instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). We assume that the register write is done in the first half of the clock cycle and that register reads are done in the second half of the cycle, so "EX to 3rd" and "MEM to 3rd" dependences are not counted because they cannot result in data hazards. Also, assume that the CPI of the processor is 1 if there are no data hazards.

EX to 1st Only	**MEM to 1st Only**	**EX to 2nd Only**	**MEM to 2nd Only**	**EX to 1st and MEM to 2nd**	**Other RAW Dependences**
5%	15%	5%	10%	10%	15%

Assume the following latencies for individual pipeline stages. For the EX stage, latencies are given separately for a processor without forwarding and for a processor with different kinds of forwarding.

IF	**ID**	**EX (no FW)**	**EX (full FW)**	**EX (FW from EX/MEM only)**	**EX (FW from MEM/WB only)**	**MEM**	**WB**

170 ps	120 ps	140 ps	170 ps	160 ps	150 ps	140 ps	120 ps

3a. If we use no forwarding, what fraction of cycles are we stalling due to data hazards?
3b. If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we staling due to data hazards?
3c. Let us assume that we cannot afford to have three-input Muxes that are needed for full forwarding. We have to decide if it is better to forward only from the EX/MEM pipeline register (next-cycle forwarding) or only from the MEM/WB pipeline register (two-cycle forwarding). Which of the two options results in fewer data stall cycles?
3d. For the given hazard probabilities and pipeline stage latencies, what is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?
3e. What would be the additional speedup (relative to a processor with forwarding) if we added time-travel forwarding that eliminates all data hazards?
Assume that the yet-to-be-invented time-travel circuitry adds 100 ps to the latency of the full-forwarding EX stage.
3f. Repeat 3c. but this time determine which of the two options results in shorter time per instruction.

4. 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	BEW	JMP	LW	SW
50%	20%	5%	20%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
40%	60%	85%

4a. 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 EX stage, that there are no data hazards, and that no delay slots are used.
4b. Repeat 4a. for the “always-not-taken” predictor.
4c. Repeat 4a. for the 2-bit predictor.
4d. With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaces a branch instruction with an ALU instruction? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.
4e. With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaced each branch instruction with two ALU instructions? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.
4f. Some branch instructions are much more predictable than others. If we know that 80% of all executed branch instructions are easy-to-predict loop-back branches that are always predicted correctly, what is the accuracy of the 2-bit predictor on the remaining 20% of the branch instructions?

5. This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes:
`NT, T, NT, NT, T`
5a. What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?
5b. What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.63 (predict not taken)?
5c. What is the accuracy of the two-bit predictor if this pattern is repeated forever?
5d. Design a predictor that would achieve a perfect accuracy if this pattern is repeated forever. You predictor should be a sequential circuit with one output that provides a prediction (1 for taken, 0 for not taken) and no inputs other than the clock and the control signal that indicates that the instruction is a conditional branch.
5e. What is the accuracy of your predictor from 5d. if it is given a repeating pattern that is the exact opposite of this one?
5f. Repeat 5d., but now your predictor should be able to eventually (after a warm-up period during which it can make wrong predictions) start perfectly predicting both this pattern and its opposite. Your predictor should have an input that tells it what the real outcome was. Hint: this input lets your predictor determine which of the two repeating patterns it is given.