1. (Exercise 4.3) [15pt] When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are starting with a datapath from Figure 4.2, where I-Mem, Add, Mux, ALU, Regs, D-Mem, and Control blocks have latencies of 400 ps, 100 ps, 30 ps, 120 ps, 200 ps, 350 ps, and 100 ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively.

Consider the addition of a multiplier to the ALU. This addition will add 300 ps to the latency of the ALU and will add a cost of 600 to the ALU. The result will be 5% fewer instructions executed since we will no longer need to emulate the MUL instruction.

1.1 [5] <§4.1> What is the clock cycle time with and without this improvement?

1.2 [5] <§4.1> What is the speedup achieved by adding this improvement?

1.3 [5] <§4.1> Compare the cost/performance ratio with and without this improvement.
2. (Exercise 4.9) [20pt] 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:

<table>
<thead>
<tr>
<th></th>
<th>Without Forwarding</th>
<th>With Full Forwarding</th>
<th>With ALU-ALU Forwarding Only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250 ps</td>
<td>300 ps</td>
<td>290 ps</td>
</tr>
</tbody>
</table>


2.2 [5] <§4.5> Assume there is no forwarding in this pipelined processor. Indicate hazards and add **nop** instructions to eliminate them.

2.3 [5] <§4.5> Assume there is full forwarding. Indicate hazards and add **nop** instructions to eliminate them.

2.4 [5] <§4.5> 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?

3. (Exercise 4.11) [25pt] Consider the following loop.

```
loop: lw r1, 0(r1)
     and r1, r1, r2
     lw r1, 0(r1)
     lw r1, 0(r1)
     beq r1, r0, loop
```
Assume that perfect branch prediction is used (no stalls due to control hazards), and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

3.1 [20] §4.6> Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction of that iteration up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration).

3.2 [5] §4.6> How often (as a percentage of all cycles) do we have a cycle in which all five pipeline stages are doing useful work?

4. (Exercise 4.12) [20pt] 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.

<table>
<thead>
<tr>
<th>EX to 1&lt;sup&gt;st&lt;/sup&gt; Only</th>
<th>MEM to 1&lt;sup&gt;st&lt;/sup&gt; Only</th>
<th>EX to 2&lt;sup&gt;nd&lt;/sup&gt; Only</th>
<th>MEM to 2&lt;sup&gt;nd&lt;/sup&gt; Only</th>
<th>EX to 1&lt;sup&gt;st&lt;/sup&gt; and MEM to 2&lt;sup&gt;nd&lt;/sup&gt;</th>
<th>Other RAW Dependences</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>20%</td>
<td>5%</td>
<td>10%</td>
<td>10%</td>
<td>10%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX (no FW)</th>
<th>EX (Full FW)</th>
<th>EX (FW from EX/MEM only)</th>
<th>EX (FW from MEM/WB only)</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>150 ps</td>
<td>100 ps</td>
<td>120 ps</td>
<td>150 ps</td>
<td>140 ps</td>
<td>130 ps</td>
<td>120 ps</td>
<td>100 ps</td>
</tr>
</tbody>
</table>

4.1 [§4.7] If we use no forwarding, what fraction of cycles are we stalling due to data hazards?

4.2 [§4.7] If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we stalling due to data hazards?

4.3 [§4.7] 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?

4.4 [§4.7] 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?

5. (Exercise 4.14) [20pts] This exercise is intended to help you understand the relationship between control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```mips
lw r2,0(r1)

label1: beq r2,r0,label2  # not taken once, then taken
       lw r3,0(r2)

       beq r3,r0,label1  # taken

       add r1,r3,r1

label2: sw r1,0(r2)
```
5.1 (4.14.1) [§4.8] Draw the pipeline execution diagram for this code, assuming that branches execute in the EX stage.

5.2 (4.14.4) [§4.8] Using the first branch instruction in the given code as an example, describe the hazard detection logic needed to support branch execution in the ID stage as in Figure 4.62. Which type of hazard is this new logic supposed to detect?

5.3 (4.14.5) [§4.8] For the given code, what is the speedup achieved by moving branch execution into the ID stage? Explain your answer. In your speedup calculation, assume that the additional comparison in the ID stage does not affect clock cycle time.

5.4 (4.14.6) [§4.8] Using the first branch instruction in the given code as an example, describe the forwarding support that must be added to support branch execution in the ID stage. Compare the complexity of this new forwarding unit to the complexity of the existing forwarding unit in Figure 4.62.