Problem 0
Read book chapter 4.5 to 4.7.

Problem 1 (30 pts)
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:

\[
\begin{align*}
&\text{or } r1, r2, r3 \\
&\text{or } r2, r1, r4 \\
&\text{or } r1, r1, r2
\end{align*}
\]

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

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

Without forwarding, there is no additional forwarding paths. Note that a new value can be written into a register in the first half of a clock cycle and this value can be read from the same register in the second half of a clock cycle. Full forwarding means the pipeline has two forwarding paths: from EX/MEM to EX (also called ALU-ALU forwarding), and from MEM/WB to EX.

1.1) Indicate dependences and their type (RAW, WAR, WAW).
[Hint: For example, RAW (read-after-write) on r2 from I2 to I3.]

1.2) Assume there is no forwarding in this pipelined processor. Indicate hazards and add \texttt{nop} instructions to eliminate them.

1.3) Assume there is full forwarding. Indicate hazards and add \texttt{nop} instructions to eliminate them.

1.4) What is the total execution time (in wall clock 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?

1.5) Add \texttt{nop} instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage). Be careful. Make sure the added \texttt{nop} instructions solve all the RAW dependencies.

1.6) What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup over a no-forwarding pipeline?
Problem 2 (30 pts)
In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
sw r16,12(r6)
lw r16,8(r6)
beq r5,r4,Label # Assume r5!=r4
add r5,r1,r4
slt r5,r15,r4
```

Assume that individual pipeline stages have the following latencies:

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>200ps</td>
<td>120ps</td>
<td>150ps</td>
<td>190ps</td>
<td>100ps</td>
</tr>
</tbody>
</table>

2.1) Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

2.2) Assume that the latency ID stage increases by 50% and the latency of the EX stage decreases by 10ps when branch outcome resolution is moved from EX to ID. Given these pipeline stage latencies, repeat the above speedup calculation, taking into account the (possible) change in clock cycle time.

2.3) Assuming stall-on-branch and no delay slots, what is the new clock cycle time and execution time of this instruction sequence if beq address computation is moved to the MEM stage? What is the speedup from this change? Assume that the latency of the EX stage is reduced by 20ps and the latency of the MEM stage is unchanged when branch outcome resolution is moved from EX to MEM.

Problem 3 (20 pts)
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), that there are no delay slots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

3.1) 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) Do we have a cycle in which all five pipeline stages are doing useful work? Please explain why. For instance, the AND instruction does not do useful work in the MEM stage.

**Problem 4 (20 pts)**

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 1. 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). For instance, “EX to 1st only” means that there is a RAW dependence from the EX stage of the current instruction to the first instruction that follows the current instruction. “Other RAW Dependencies” means that the dependence is more than 2 instructions away.

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 1st Only</th>
<th>MEM to 1st Only</th>
<th>EX to 2nd Only</th>
<th>MEM to 2nd Only</th>
<th>EX to 1st and MEM to 2nd</th>
<th>Other RAW Dependencies</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) If we use no forwarding, what fraction of cycles are we stalling due to data hazards?

4.2) If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we stalling due to data hazards?
Extra credits for the following two questions (5 pts each)

4.3) 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?

4.4) 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.

![Figure 1. The single-clock-cycle diagram corresponding to clock cycle 5 of the pipeline.](image)