Problem 1 (25pts).

Problem 3.1 in PH. For this problem assume that the branch condition and the target address are computed in the EX stage and available in the MEM stage.

Also, solve the problem for the following code segment instead of the one proposed in the book:

\[
\begin{align*}
I1: & \quad \text{Loop:} \quad \text{LW} \quad r1, 0 (r2) \\
I2: & \quad \text{LW} \quad r3, 0 (r4) \\
I3: & \quad \text{ADD} \quad r3, r1, r3 \\
I4: & \quad \text{SW} \quad r3, 0 (r2) \\
I5: & \quad \text{SUBI} \quad r2, r2, #4 \\
I6: & \quad \text{SUBI} \quad r4, r4, #4 \\
I7: & \quad \text{BNEZ} \quad r4, \text{Loop}
\end{align*}
\]

The initial value of r4 is 100. The pipeline is empty at the start and time should be counted until all instructions in the loop have cleared the pipeline.

Problem 2 (25pts).

Problem 3.3 in PH. Whenever possible assume that the function of each stage is the same as in the basic DLX pipeline.

Solve first part (a) and (b).

Then, instead of solving the rest of the problem, fill up Table 1. In each row of this table circle YES if you think that a forwarding path should exists, and circle NO otherwise. Please give some thought to your answers. Some answers are obvious, others are difficult. In the right-most column give an example that justifies the forwarding path (if any).

Problem 3 (25pts)

Problem 3.9 in PH. Assume here that there are 4 stages: IF, ID, EX, WB (actually the identity of the stages is irrelevant for this problem. An instruction is always fetched. However, if the preced-
ing instruction is decoded as a branch, then the instruction following the branch waits until the end of the 3rd cycle of the branch instruction to continue. If the branch is taken the instruction must then be re-fetched, otherwise the instruction can directly enter the ID stage.

**Table 1:**

<table>
<thead>
<tr>
<th>From pipeline register</th>
<th>To pipeline register</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU1/MEM</td>
<td>RF/ALU1</td>
<td>YES/NO</td>
</tr>
<tr>
<td>MEM/ALU2</td>
<td>RF/ALU1</td>
<td>YES/NO</td>
</tr>
<tr>
<td>MEM/ALU2</td>
<td>ALU1/MEM</td>
<td>YES/NO</td>
</tr>
<tr>
<td>ALU2/WB</td>
<td>RF/ALU1</td>
<td>YES/NO</td>
</tr>
<tr>
<td>ALU2/WB</td>
<td>ALU1/MEM</td>
<td>YES/NO</td>
</tr>
<tr>
<td>ALU2/WB</td>
<td>MEM/ALU2</td>
<td>YES/NO</td>
</tr>
</tbody>
</table>

**Problem 4 (25pts)**

Problem 3.10 in PH. In (a) there is NO forwarding, including no internal forwarding in the register file. In (b) there is enough forwarding in the 4-stage pipeline so that the numbers of stalls due to data dependencies in both pipelines are equal.