Problems

1. Pipeline control
   Consider the instruction
   \[
   \text{add } $5, $5, $5
   \]
   As this instruction executes, what is kept in each register located between two pipeline stages?

2. Pipeline performance
   Each pipeline stage has some latency. Additionally, pipelining introduces registers between stages, and each register adds further latency.

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
<th>Pipeline register</th>
</tr>
</thead>
<tbody>
<tr>
<td>250ps</td>
<td>200ps</td>
<td>150ps</td>
<td>180ps</td>
<td>180ps</td>
<td>20ps</td>
</tr>
</tbody>
</table>

   (a) What is the latency of a \text{lw} instruction in a nonpipelined vs a pipelined processor?
   (b) What is the clock cycle time in a nonpipelined vs a pipelined processor?
   (c) What is the speed-up achieved by introducing pipelining?
   (d) If we could split one stage 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?

3. Hazards
   Consider the following sequence of instructions executed on a five-stage pipelined datapath:

   \[
   \text{lw } $1, 40($6) \\
   \text{add } $2, $3, $1 \\
   \text{add } $1, $2, $6 \\
   \text{sw } $2, 20($4) \\
   \text{and } $1, $1, $4
   \]

   (a) Assuming there is no forwarding, indicate data hazards and show how to insert \text{nop} instructions to eliminate the hazards.
   (b) Assuming there is forwarding, draw a multiple-clock-cycle pipeline diagram, as we drew in class, to show where and when data will be forwarded in the execution of these instructions. Show any stalls (“pipeline bubbles”) that are required.

4. Error correction
   (a) Suppose an 8-bit data word is 11000010. Using the Hamming code algorithm, show how the data will be stored in memory along with its check bits.
   (b) For the 8-bit data word 00111001, the check bits stored would be 0111. Suppose that when the word is read from memory, the check bits are calculated as 1101. What is the data word that was read from memory?
   (c) How many check bits are needed if the Hamming error correction code is used to detect single bit errors in a 1024-bit data word?
   (d) Explain in your own words how adding an extra parity bit to a Hamming code lets us detect, but not correct, two errors in the same data word.