1.) For the following code, explain what is happening in each stage of the pipelined processor during cycle 5.

   ADD X0, X31, #102
   LDUR X1, [X0, #10]
   STUR X0, [X0, #10]
   CBZ X1, LOOP
   EOR X6, X4, X1

2.) Our processor has a load delay slot, which means code cannot use the Rd register of a load in the instruction right after a load. What happens if we ignore this restriction? Specifically, assume the following code is given to the CPU. Assume MEM[32] = -4. What value will end up in register X5? Think carefully – this one is tricky!

   ADDI X3, X31, 24
   LDUR X3, [X3, #8]
   ADD X5, X3, X3

3.) The following code contains a “read after write” data hazard that is resolved by forwarding:

   ADD X2, X3, X4
   ADD X5, X2, X6

   Consider the following code where a memory read occurs after a memory write:

   STUR X7, [X2, #100]
   LDUR X8, [X2, #100]

   Does the code work correctly? Why/why not? Will the forwarding unit need to be altered to handle this code?

4.) In this question, we examine how pipelining affects the clock cycle time of the processor. Assume that individual stages of the datapath 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>170ps</td>
<td>220ps</td>
<td>210ps</td>
<td>150ps</td>
</tr>
</tbody>
</table>

   a.) What is the clock cycle time of the pipelined and single-cycle CPU?

   b.) 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? You can ignore hazards for answering this question.

5.) For the following code, explain what the register file and forwarding unit are doing during the fifth cycle of execution. If any comparisons are being made, mention them. Remember, use the forwarding unit from class, not the book.
ADD    X1, X2, X3
STUR   X2, [X1, #0]
LDUR   X1, [X2, #4]
ADD    X2, X2, X3
EOR    X5, X4, X1