This quiz covers one of the problems from homework #5.
Good Luck!

<table>
<thead>
<tr>
<th>Your Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SID Number:</td>
<td></td>
</tr>
<tr>
<td>Discussion Section:</td>
<td></td>
</tr>
</tbody>
</table>

Total:
In problem 6.19 and 6.20, you explored hazards involving memory instructions. For your reference, Figure 6.51 from the book is reproduced on the previous page.

1. Consider the following instruction sequence (RAW hazard through memory):
   
   \[
   \begin{align*}
   &\text{sw } \$2, 20(\$4) \\
   &\text{lw } \$5, 20(\$4)
   \end{align*}
   \]

   Does this require forwarding hardware for maximum performance? If yes, draw/describe the forwarding hardware and describe the control circuitry. If no, explain why not.

   \textit{No. Both of these operations occur in the memory stage. Hence, the value is stored out on one cycle and grabbed on the next cycle. This is a forward motion of information that happens automatically through the memory system.}

   \[
   \begin{array}{ccccccc}
   & & & F & D & E & M & W \\
   \text{sw} & & & F & D & E & M & W \\
   \text{lw} & & & F & D & E & M & W
   \end{array}
   \]

2. Consider the following instruction sequence (RAW hazard through registers):
   
   \[
   \begin{align*}
   &\text{lw } \$2, 20(\$4) \\
   &\text{sw } \$2, 100(\$5)
   \end{align*}
   \]

   Does this require forwarding hardware for maximum performance? If yes, draw/describe the forwarding hardware and describe the control circuitry. If no, explain why not.

   \textit{Yes. This requires forwarding for maximum performance. Without forwarding, we would have to wait for register \$2 to get written back to the register file before accessing it (2 instructions or stalls between). With normal forwarding, we would still have to stall for 1 instruction since the output value is not available until the end of the memory stage, and normal forwarding forwards into the end of the decode stage. However, noting that the value for a store is only necessary at the beginning of the memory stage, we could use special forwarding hardware:}

   \[
   \begin{array}{ccccccc}
   & & & & & & \\
   & & & & & & \\
   \text{Special Forwarding MUX}
   \end{array}
   \]

   \textit{This forwarding logic consists of a mux that connects the output of the memory to the store value for the next instruction. The control logic is simple:}

   \textit{In the execute stage of a store instruction, check to see if:}
   
   1. The instruction in the memory stage is a load.
   2. The destination register for that load matches the source register for the store in the execute stage

   \textit{If these two conditions are met, flip the mux to latch the value from the memory rather than from register rt (normal
mode).