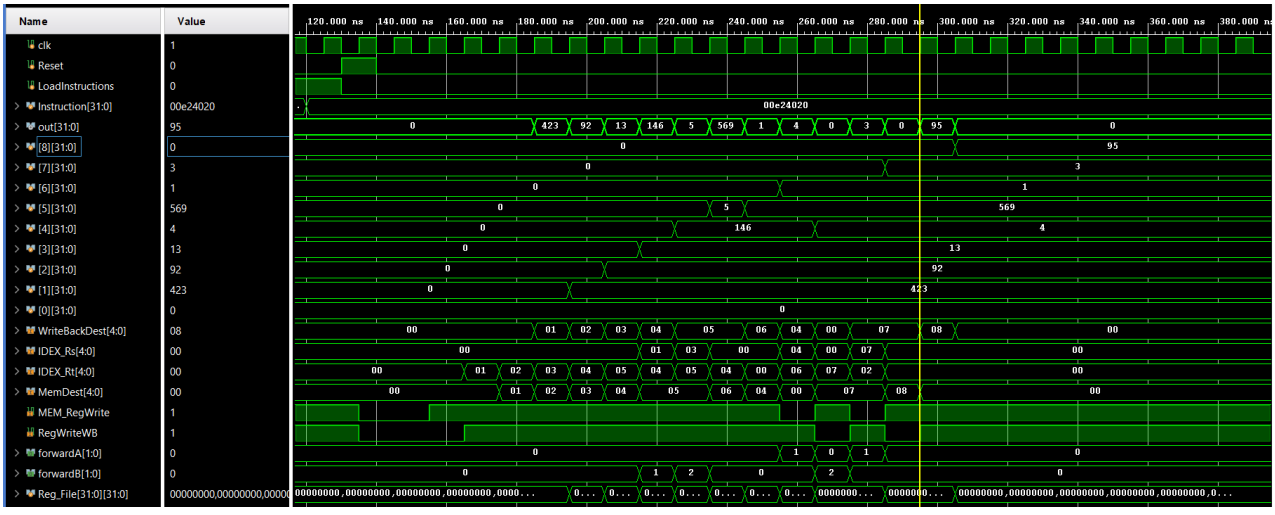
Task 1:
For this task, I just commented my 1 and 2-ahead hazard “flags”, and removed them from my ternary statements.

Tasks 2 through 5:
I implemented these tasks together using ternary statements inside my forwarding unit module. Using the logic from the data forwarding lecture slides, I first implemented one-ahead forwarding by first checking that a register in the MEM stage is being written to. Then, I check that the MEM stage Rd register is not equal to zero. Next, if the MEM stage Rd register is equal to IDEX_Rs, I set forwardA = 2’b10. Otherwise, if the MEM stage Rd register is equal to RegDestMuxOut (IDEX_Rs/Rt), I set forwardB = 2’b10.

I then implemented two-ahead forwarding by first checking that a register in the WB stage is being written to. Then, I check that the WB stage Rd register is not equal to zero. Next, if the WB stage Rd register is equal to IDEX_Rs, and the MEM stage Rd register is NOT equal to IDEX_Rs, I set forward A equal to 2’b01. Otherwise, if the WB stage Rd register is equal to RegDestMuxOut (IDEX_Rs/Rt), and the MEM stage Rd register is NOT equal to RegDestMuxOut, I set forwardB equal to 2’b01.

In order to decide between 1 and 2 ahead forwarding, I implemented two muxes in front of the ALU. Both muxes take IDEX_A/B, the WriteBackData, and MemAluOut as inputs, and use forwardA/forwardB as selects. The forwardA mux feeds its output directly into the ALU, and the forwardB mux feeds its output into the AluResultMux, which selects between the immediate and the forwardB mux output to be used as the second input for the ALU.

Waveform:



To test my design, I added my forwarding unit and register file to my waveform, and checked the wires used in my conditional logic against the register values as the instructions executed.