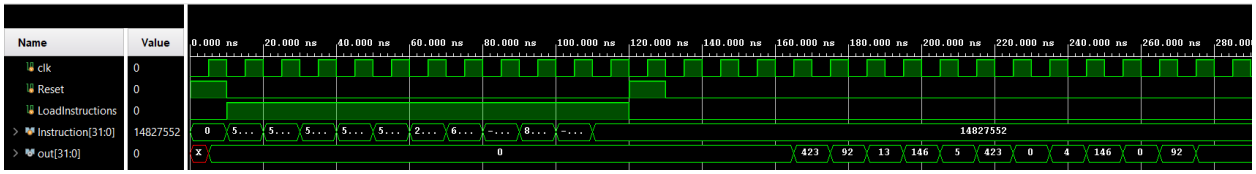
Leanorine Lorenzana-Garcia
Lab 7 Report

Task 1: Generating Output

For this task, we decided to run the testbench of the CPU and noticed that the outputs for some instructions were wrong (5, 6, and 8). However, we were able to synthesize and generate the outputs for the given instruction sequence.



The image above shows the outputs for the instructions where instructions 5, 6, and 8 are incorrect.

Task 2: "1 ahead" Forwarding

For "1 ahead" forwarding, we checked if (EXMEM_RegWrite && EX_MEM_RegisterRd != 0) && (EX_MEM_RegisterRd == EX_rs) or if (MEMWB_RegWrite && MEM_WB_RegisterRd != 0) && (MEM_WB_RegisterRd == EX_rs). If either of these conditions are true, then forwardA returns 2'b01.

Task 3: "2 ahead" Forwarding

For "2 ahead forwarding, aside from checking for "1 ahead" forwarding, we also checked if (EXMEM_RegWrite && EX_MEM_RegisterRd != 0) && (EX_MEM_RegisterRd == EX_rt) or if (MEMWB_RegWrite && MEM_WB_RegisterRd != 0) && (MEM_WB_RegisterRd == EX_rt). If either of these conditions are true then forwardB returns 2'b10. We also implemented it using only if statements since both cases of forwardA and forwardB can be true at the same time.

Task 4: Arbitration Logic

To check for arbitration logic, we decided to check if (EX_MEM_RegisterRd != EX_rs) and (EX_MEM_RegisterRd != EX_rt) for forwardA and forwardB respectively. If the EX_MEM_RegisterRd is equal to either EX_rs or EX_rt it will not run.

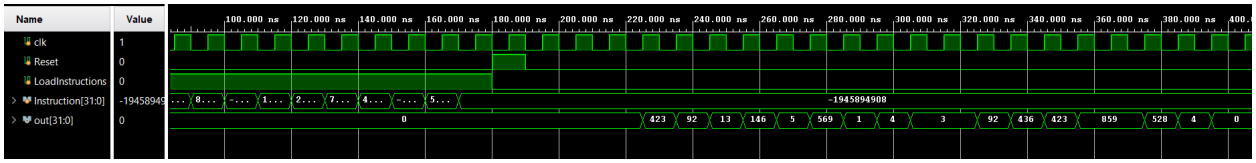
Task 5: $0 Write

To check for $0 write, in the code above, we include logic to check if EX_MEM_RegisterRd is equal to 0. If it is, it would output forward as 0.

Task 6: No Write

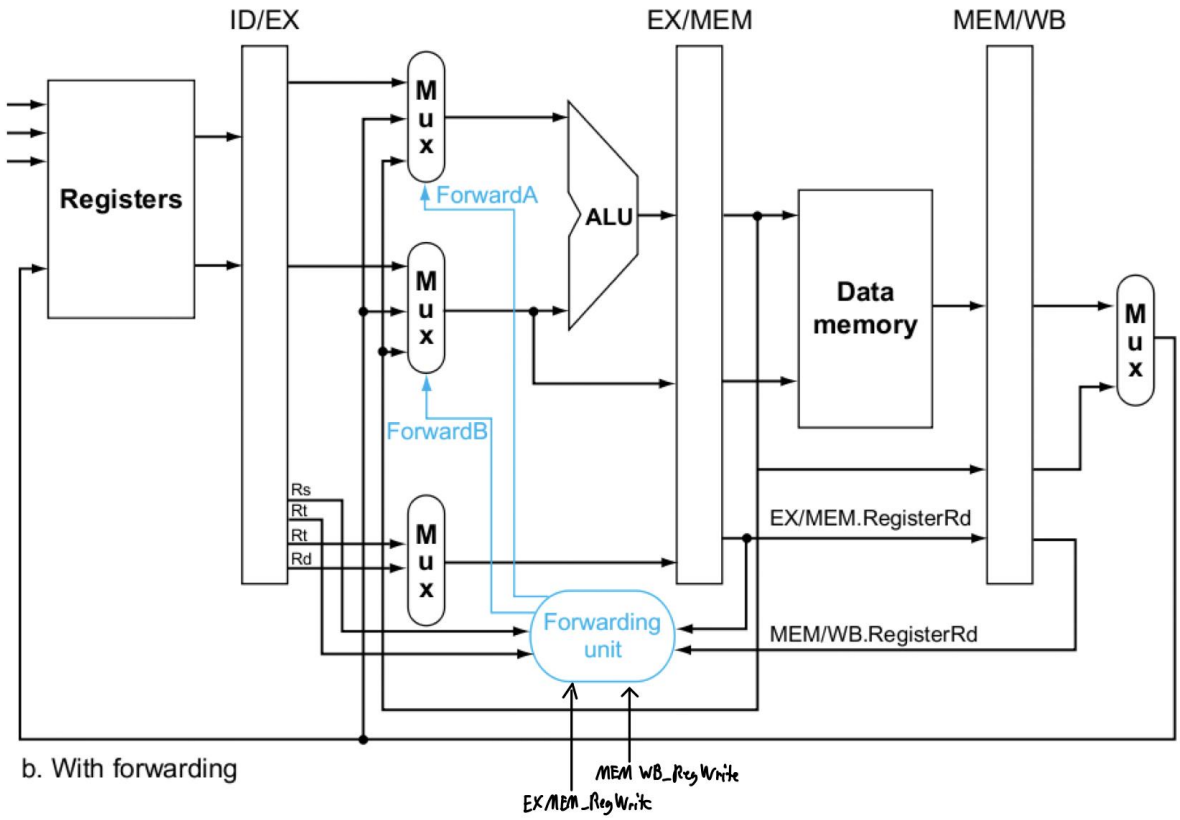
To check for no write, we initialize both forwards to 0 and if none of the conditions pass, it stays as 0.

Task 7: Register Bypass



The image above shows the final output with extra test cases we added. It displays the correct output of 5, 6, and 8, which are 569, 1, and 3 respectively. We also repeated the same test cases to check if it is passing through correctly.

Updated Diagram:



The image above shows the implementation of Forwarding Unit where it takes input Rt, Rd, EX/MEM.RegisterRd, MEM/WB.RegisterRd, EX/MEM.RegWrite, and MEM/WB.RegWrite. It then outputs ForwardA and ForwardB to one MUX each where the mux for A has inputs IDEX_A, WriteBackData, and MemAluOut. ForwardB MUX has inputs AluIB, WriteBackData, and MemAluOut.