Aaron Hong (ahong02)

Stephen Macris (smacris)

EE 469

9 May 2023

**Lab 3 Report**

**Procedure**

In this lab, the single cycle processor was upgraded to a pipelined processor. Stalling, flushing, and forwarding logic were added to handle data and control hazards.

**Results**

Text, table

Description automatically generated

Figure 1: Test assembly instructions with expected register outputs

A picture containing schematic

Description automatically generated

Figure 2: Simulation waveform

As shown in Figure 1 and 2, the pipelined processor is executing the instructions correctly. Stalling, flushing, and forwarding processes are also demonstrated. Figure 2 will be investigated more thoroughly in the following sections. Figures 3, 4, and 5 are magnified portions of Figure 2.

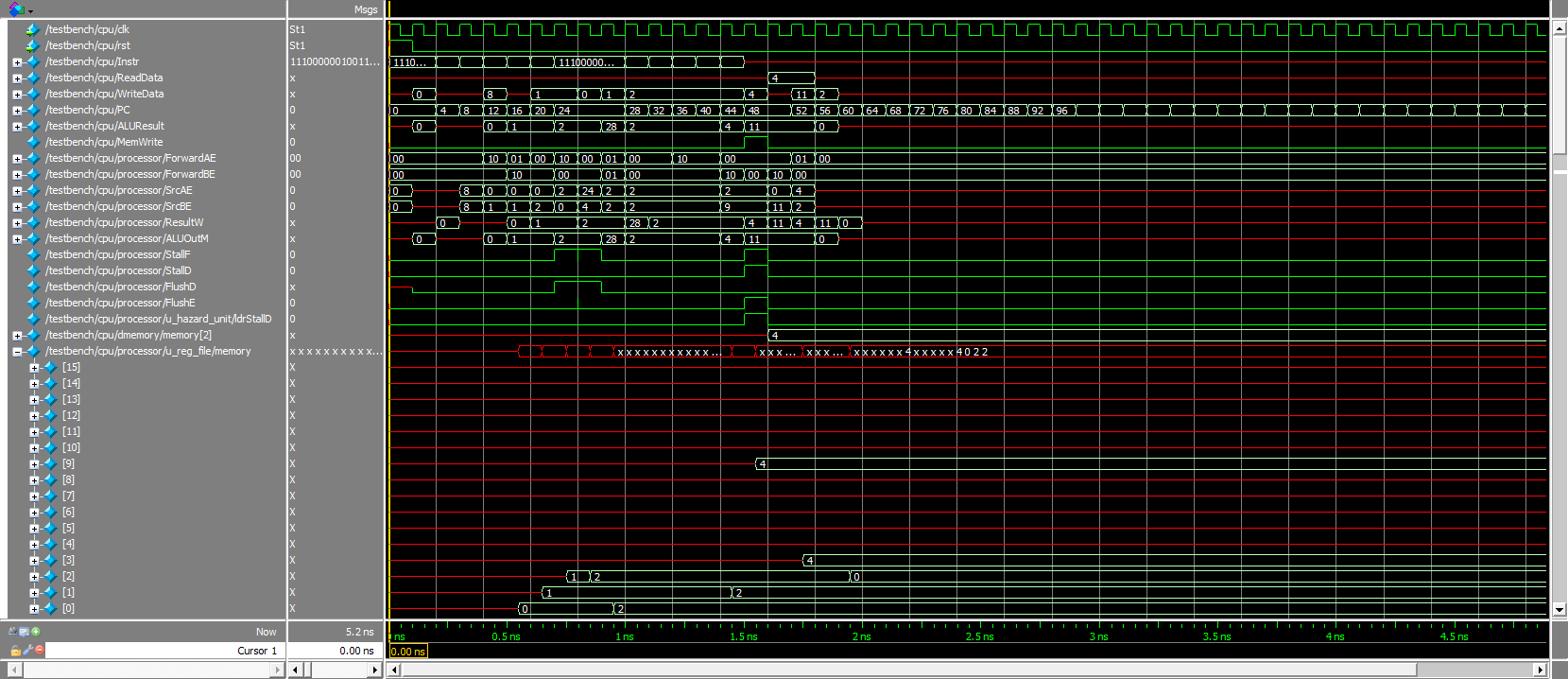
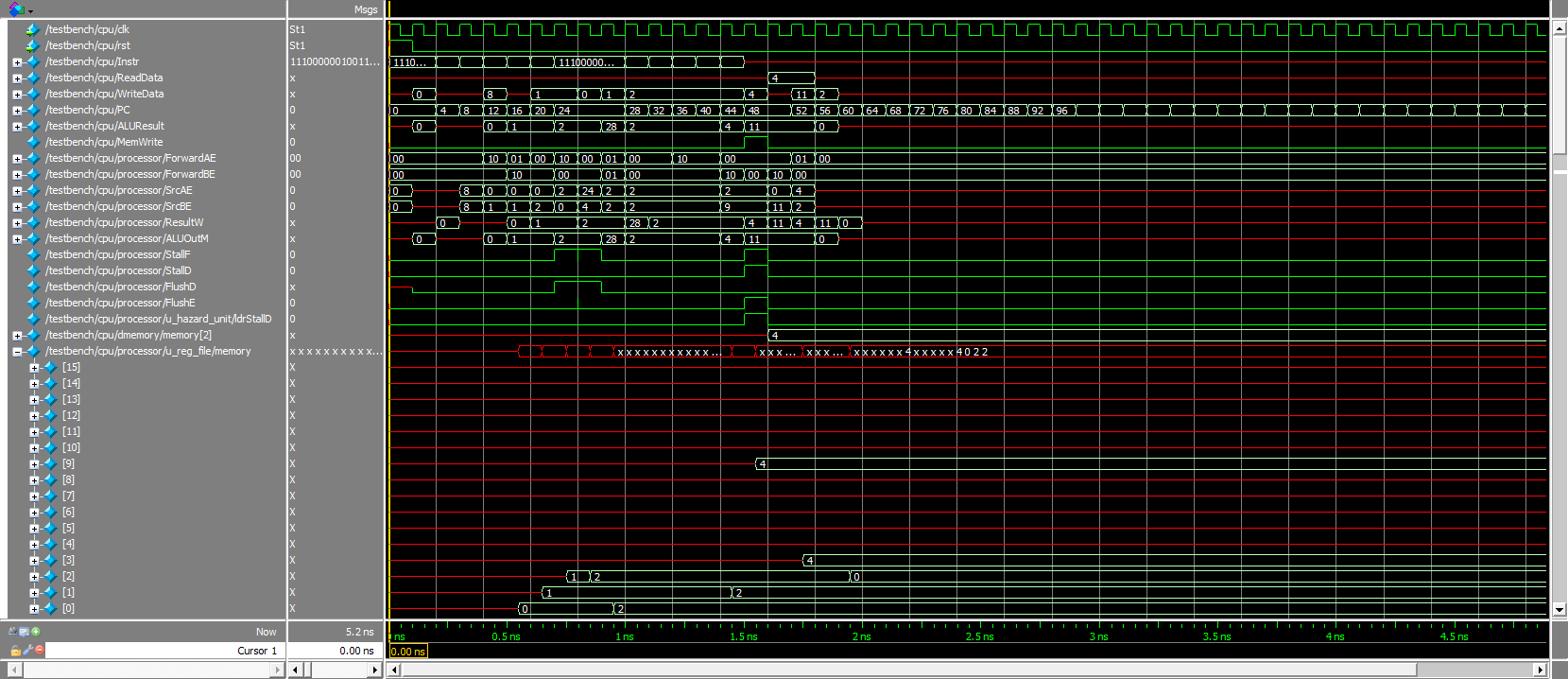


Figure 3: Forwarding signals

When PC=12, instruction 4 is in the execution stage and instruction 0 is in the memory stage. Since source register A of instruction 4 is the same as the destination register of instruction 0, there is a match between execute and memory stages and ForwardAE is set to 10. Hence, the data from the memory stage is forwarded to the execution stage: SrcAE = ALUOutM.

When PC=16, instruction 8 is in the execution stage and instruction 0 is in the writeback stage. Since source register A of instruction 8 is the same as the destination register of instruction 0, there is a match between execute and writeback stages and ForwardAE is set to 01. Hence, the data from the writeback stage is forwarded to the execution stage: SrcAE = ResultW.

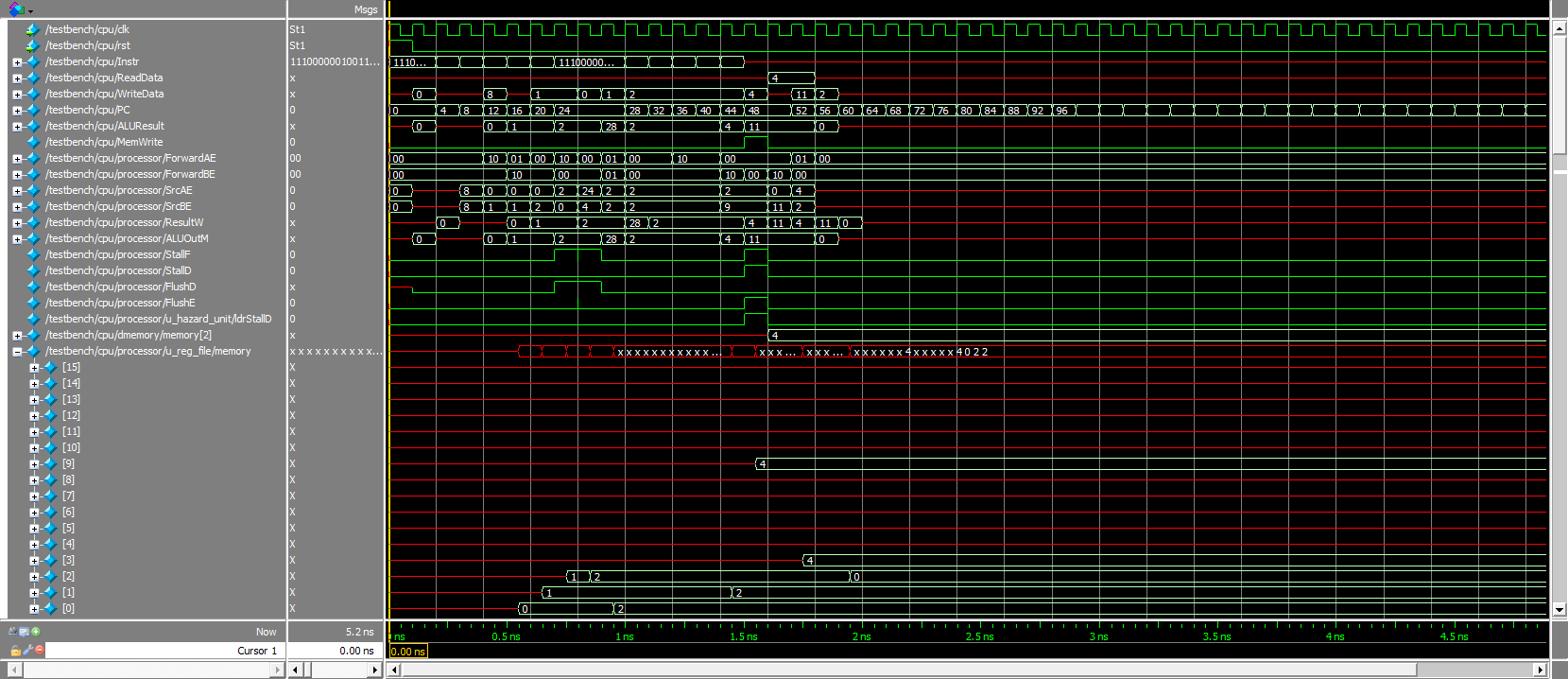
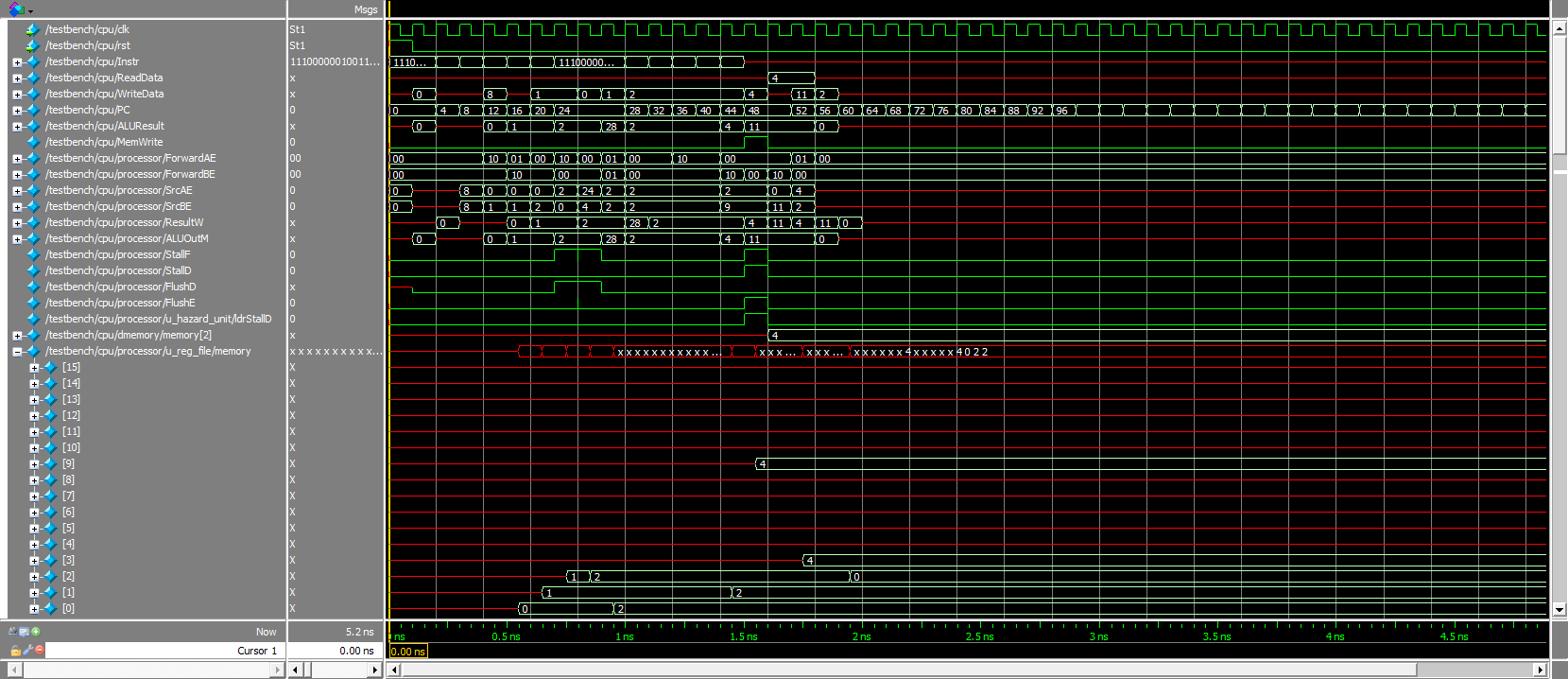


Figure 4: Branch flushing

Figure 4 shows that the two cycles of instruction are being flushed when the branch instruction (instruction 20) enters the decode stage.

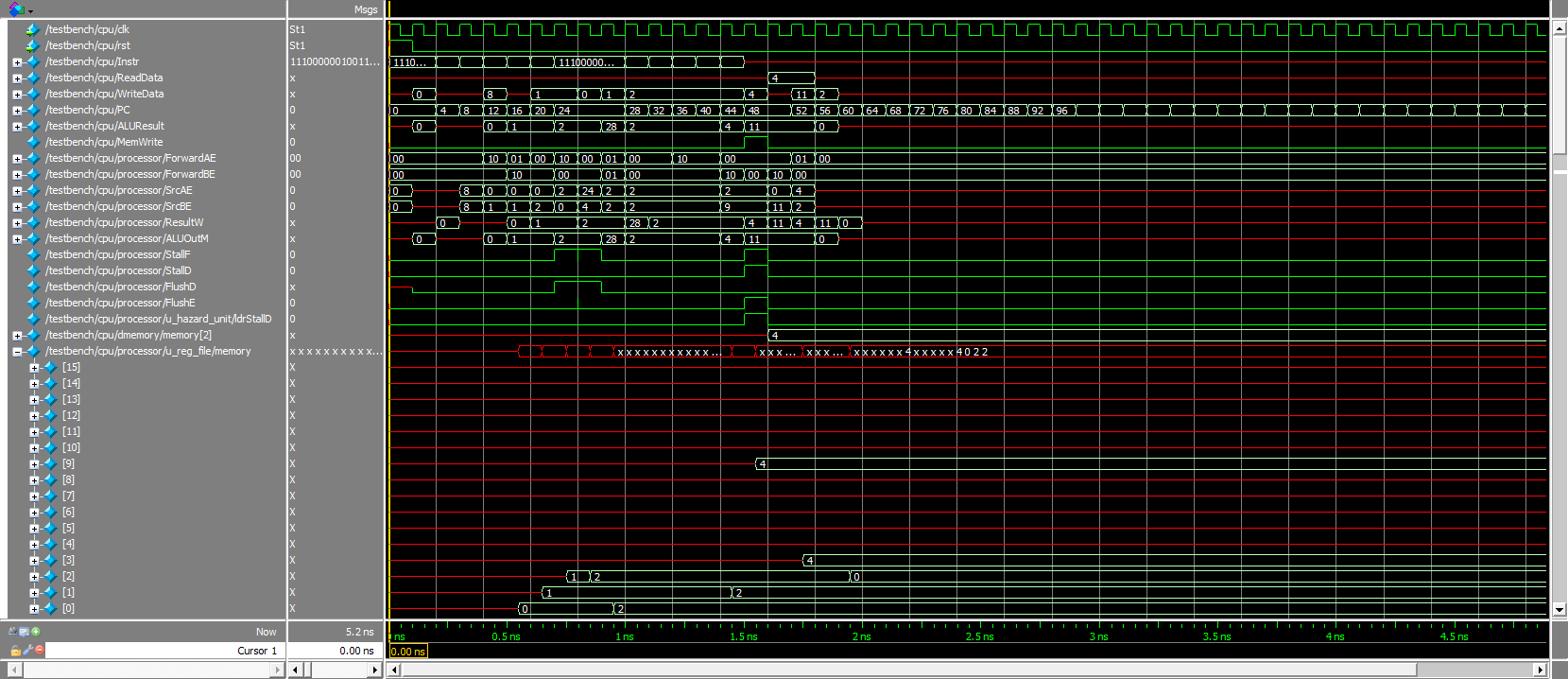
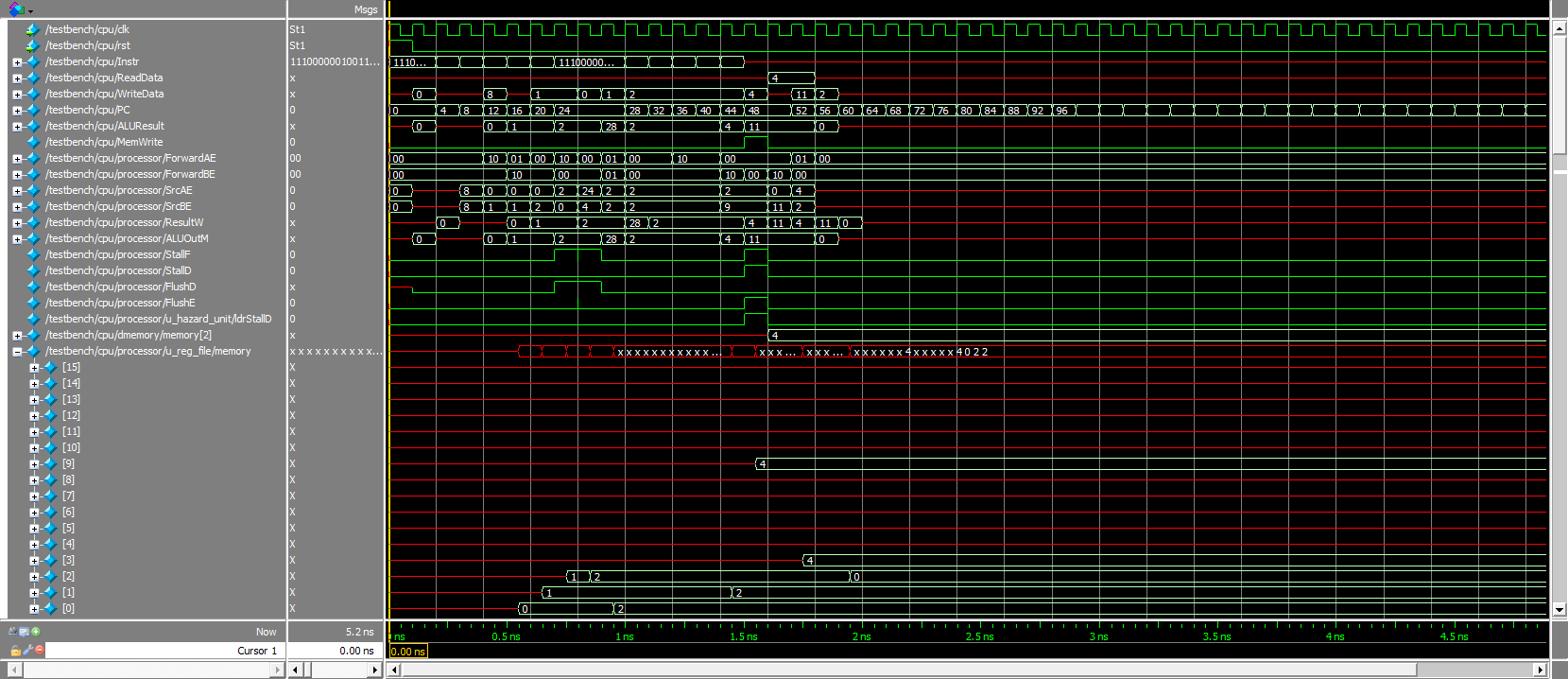


Figure 5: Memory stalling

Figure 5 shows that the processor stalled for the LDR instruction (instruction 40) to finish before executing instruction 44, which accessed the destination register of instruction 40.

**Appendix**