Eugene Ngo

EE 469

May 3, 2022

Lab 3 Report

Procedure:

This lab was comprised of one task:

1. Implementing the pipelined arm CPU module by changing the original single-cycle CPU from lab 2.

After the pipelined CPU was implemented, it was thoroughly tested in Modelsim, ensuring to test forwarding from memory to execute, forwarding from writeback stage to execute, an example of stalling for a memory instruction and flushing for a branch instruction.

Task #1:

This was the main task, transforming the original single cycle CPU into a pipelined CPU using multiple cycles per instruction and skipping and forwarding between instructions as required. This was done by following the mapped-out diagram of the pipelined CPU from class and creating more logic signals as required, to forward and skip as required. At each stage of the cycle a flip-flop was added, to ensure that each cycle executed once per clock cycle (technically, each stage is executed near instantaneously, just the input and output of each stage is on a clock cycle to ensure proper pipelining). The schematic I implemented is shown in the diagram below.

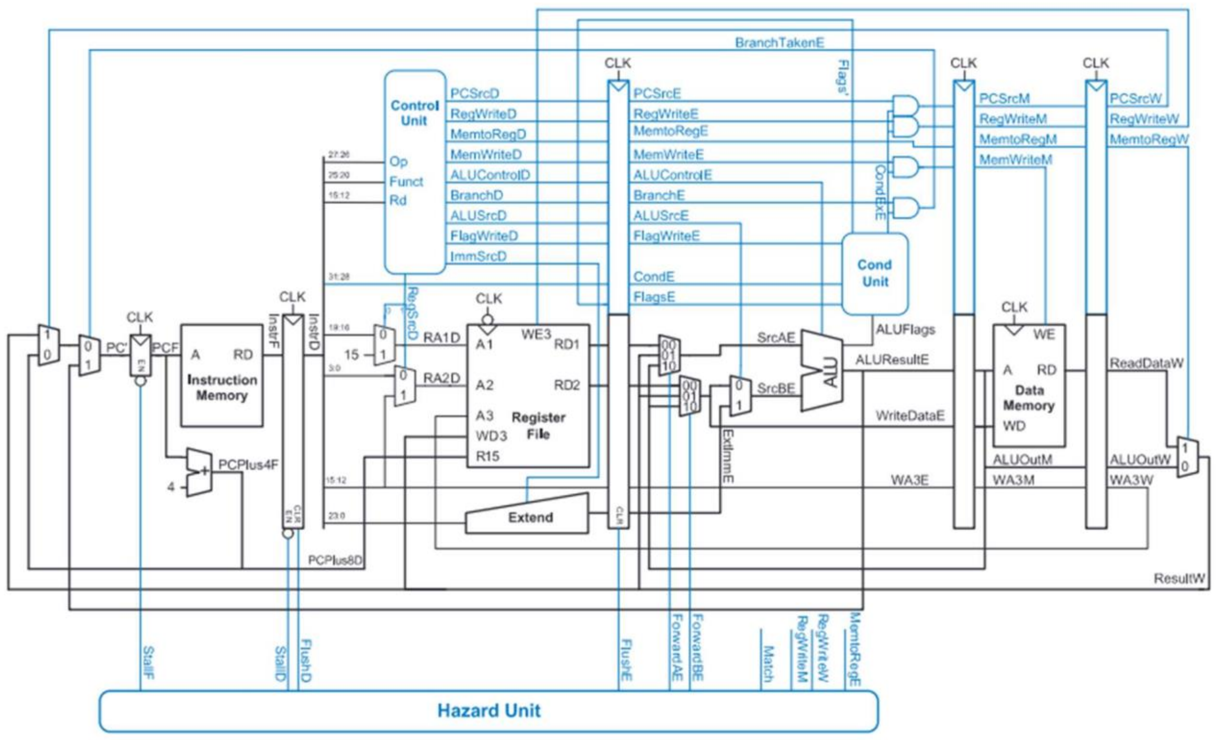

Figure 1: Schematic for a pipelined ARM CPU

This schematic was then written and compiled in Quartus and then tested in Modelsim by varying the Instruction input based on the memfile.dat files.

The values of instructions and the resulting signals and outputs were tabulated to determine the proper expected values from the CPU as it goes through memfile.dat, memfiled2.dat and memfile3.dat.

Results

Task #1:

After implementing the ARM module, I ran Modelsim to test it.

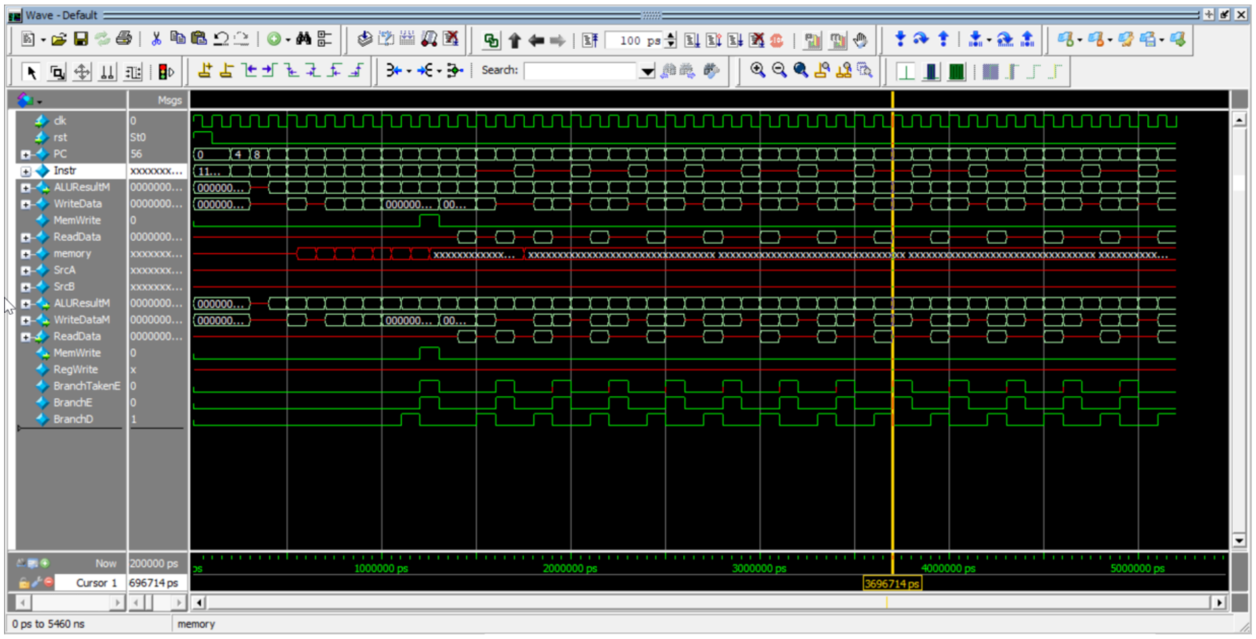

Figure 2: The waveform generated for task1, memfile.dat

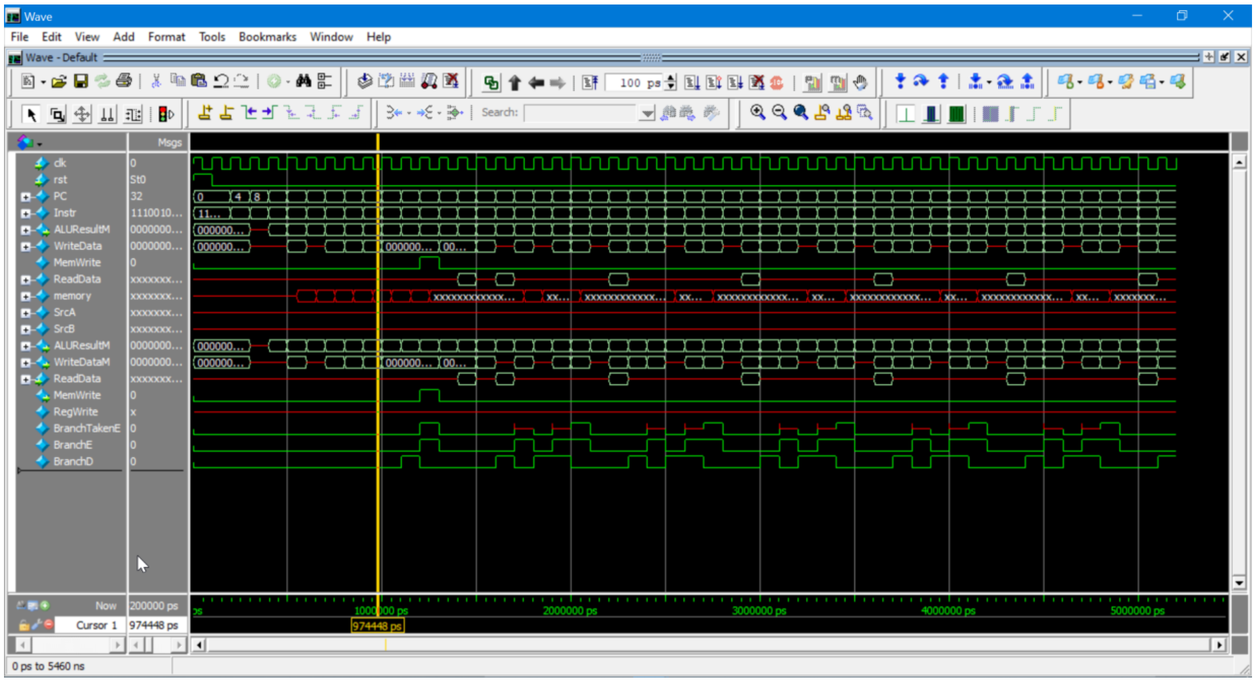

Figure 3: The waveform generated for task1, memfile2.dat

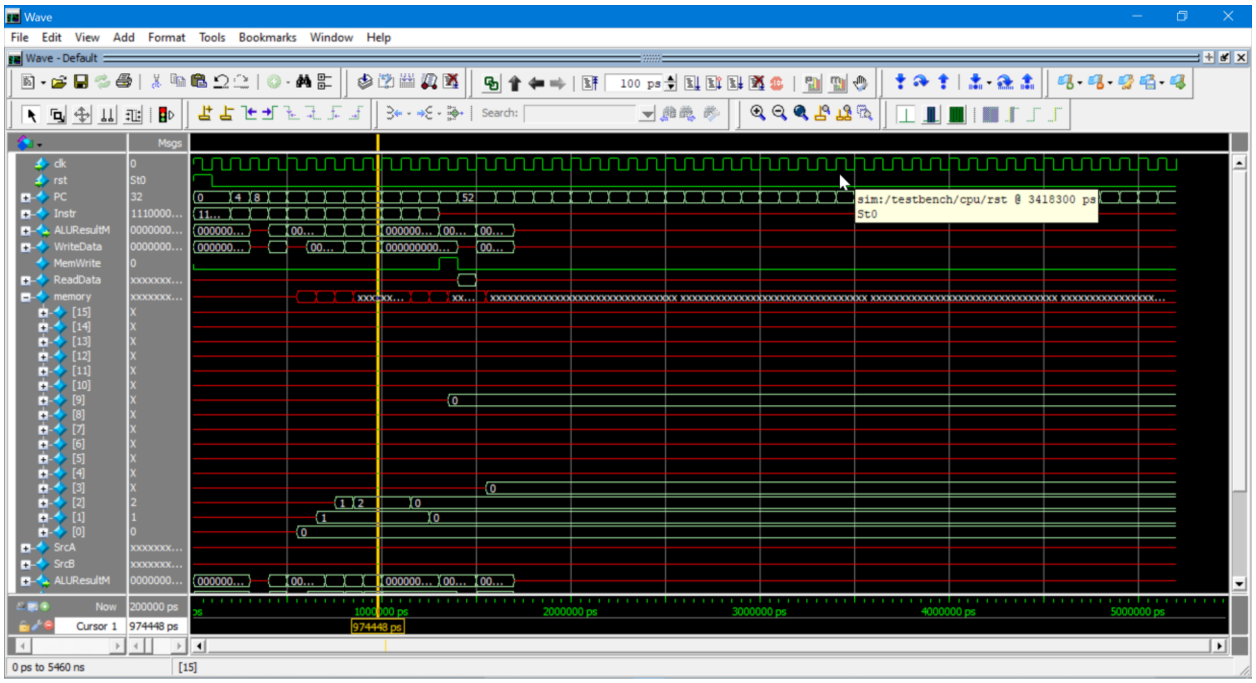

Figure 4: The waveform generated for task1, memfile3.dat

As seen in the waveforms above, the pipelined ARM CPU varies the values of the regfile based on the instructions executed, as a CPU should.

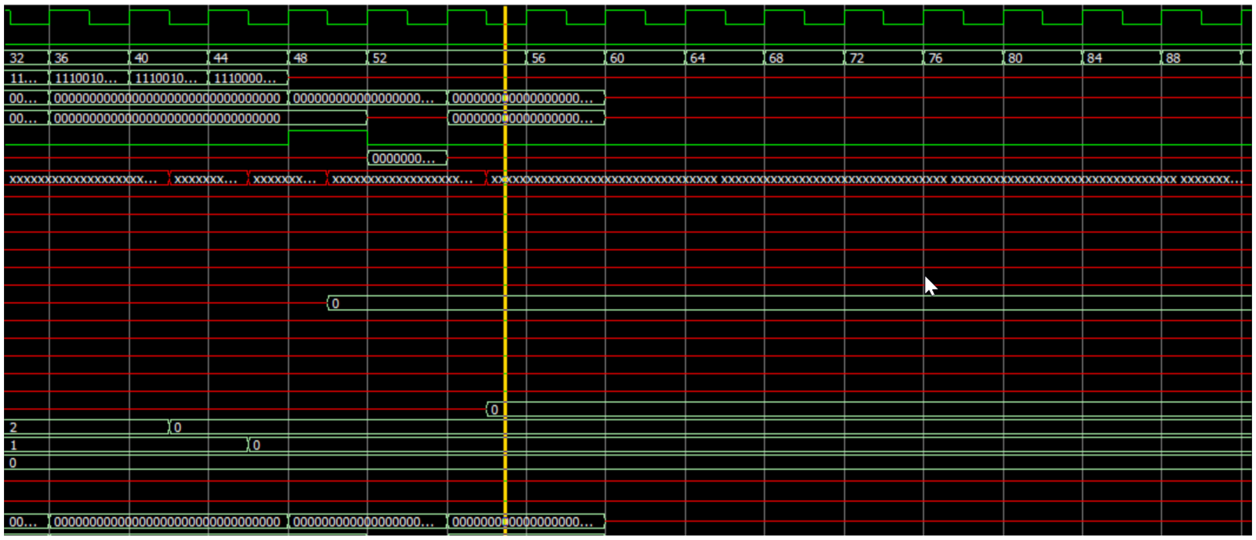

Figure 5: Memfile3.dat

As shown in Figure 5, Memfile3.dat stalls at PC = 52 as the previous two instructions (44 and 48) are store and load respectively, with consecutive memory accesses to the same location. To ensure correct data being read in, the CPU stalls for memory instruction store being completed.

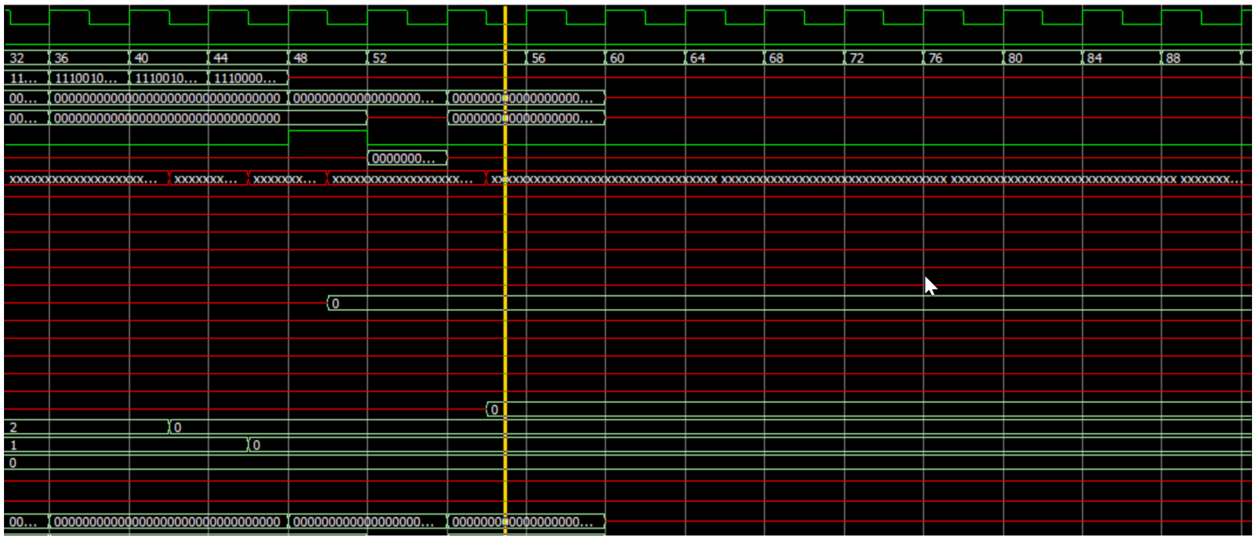

Figure 6: Memfile.dat

As shown in Figure 6, Memfile.dat forwards data at PC = 52 as the instruction before loads a value into R3 which is used in PC = 52, thus the computed value is forwarded to PC = 52's execute stage, ensuring the integrity of the CPU architecture.

Appendix

See the following list for the order:

top.sv
testbench.sv
reg_file.sv
reg_file_testbench.sv
dmem.sv
imem.sv
fullAdder.sv
arm.sv
alu.sv
alu_testbench.sv

See the attached documents for the code: