Eugene Ngo
CSE 469
April 20, 2023
Lab 2 Report

Procedure:

This lab was comprised of two tasks:

1. Implementing the given arm CPU module with the ALU and reg_file that was created in Lab 1.

2. Add extra control path logic to implement CMP, B EQ, B NE, B GE, B GT, B LE, B LT instructions

Once the designs for both tasks were implemented, they were thoroughly tested in ModelSim using the given testbenches and the DAT files, to demonstrate their functionality.

Task #1:

The first task was to implement the given ARM CPU with the ALU and reg_file that were created in Lab 1. This was done by looking through the mapped-out diagram for the single-cycle CPU in the Lab 2 document and then aligning the proper signals that were already implemented in the ARM module with the inputs and outputs for the reg_file. The outputs from the reg_file were then properly aligned with the inputs of the ALU and then processed by the ALU and outputted for the memory or writeback sections. The schematic I implemented is shown in the diagram below.

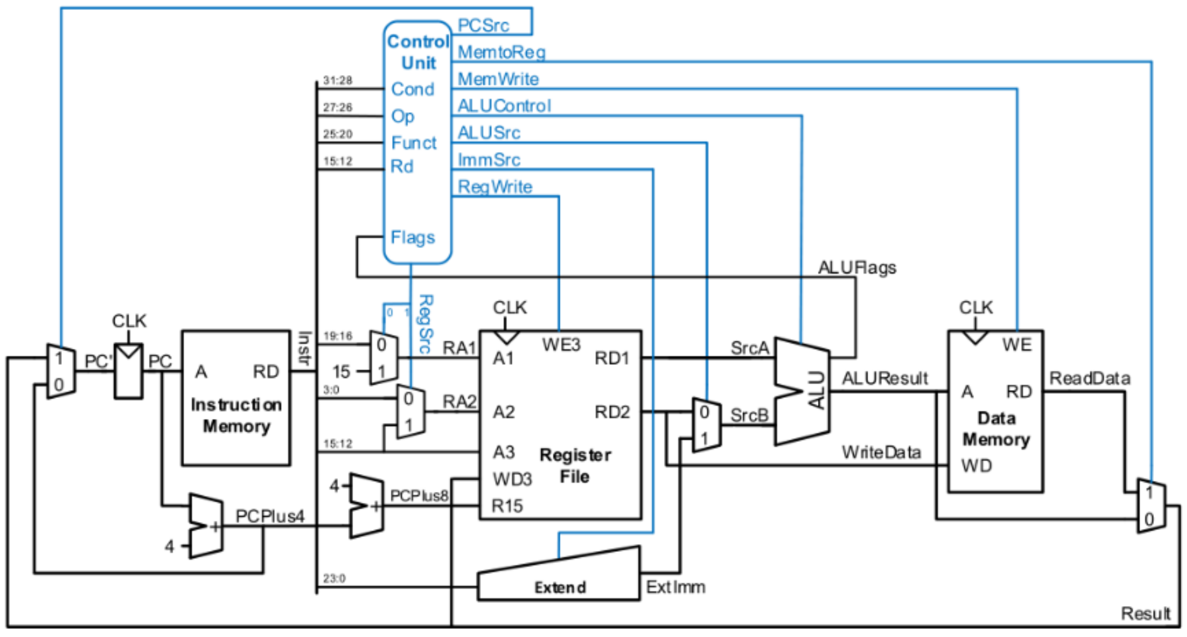
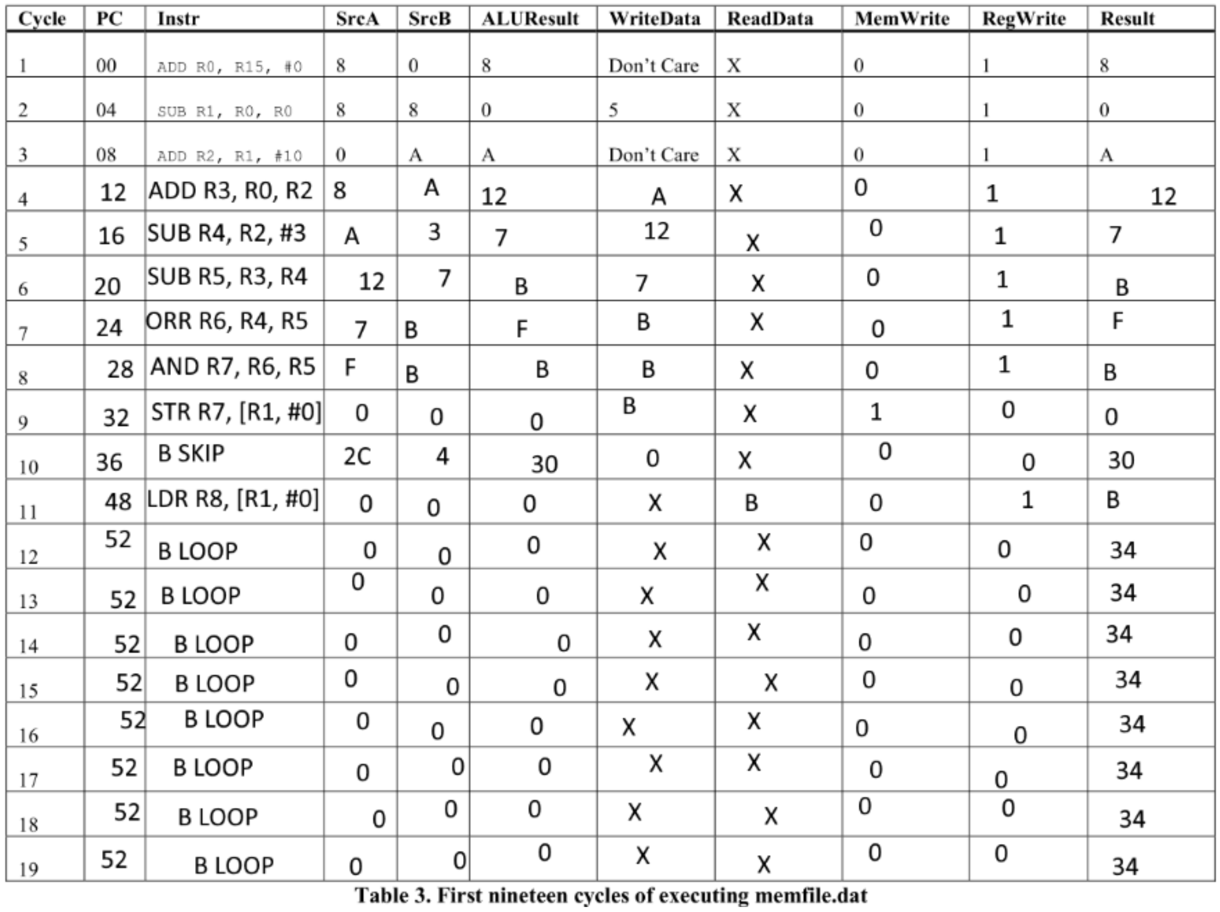


Figure 3. Single-cycle ARM processor

Figure 1: Schematic for a single-cycle 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 file.

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



Cycle	PC	Instr	SrcA	SrcB	ALUResult	WriteData	ReadData	MemWrite	RegWrite	Result
1	00	ADD R0, R15, #0	8	0	8	Don't Care	X	0	1	8
2	04	SUB R1, R0, R0	8	8	0	5	X	0	1	0
3	08	ADD R2, R1, #10	0	A	A	Don't Care	X	0	1	A
4	12	ADD R3, R0, R2	8	A	12	A	X	0	1	12
5	16	SUB R4, R2, #3	A	3	7	12	X	0	1	7
6	20	SUB R5, R3, R4	12	7	B	7	X	0	1	B
7	24	ORR R6, R4, R5	7	B	F	B	X	0	1	F
8	28	AND R7, R6, R5	F	B	B	B	X	0	1	B
9	32	STR R7, [R1, #0]	0	0	0	B	X	1	0	0
10	36	B SKIP	2C	4	30	0	X	0	0	30
11	48	LDR R8, [R1, #0]	0	0	0	X	B	0	1	B
12	52	B LOOP	0	0	0	X	X	0	0	34
13	52	B LOOP	0	0	0	X	X	0	0	34
14	52	B LOOP	0	0	0	X	X	0	0	34
15	52	B LOOP	0	0	0	X	X	0	0	34
16	52	B LOOP	0	0	0	X	X	0	0	34
17	52	B LOOP	0	0	0	X	X	0	0	34
18	52	B LOOP	0	0	0	X	X	0	0	34
19	52	B LOOP	0	0	0	X	X	0	0	34

Table 3. First nineteen cycles of executing memfile.dat

Figure 2: Signals and output when running memfile.dat

Task #2:

The second task was to implement CMP, B EQ, B NE, B GE, B GT, B LE, and B LT instructions. This required modifying the control logic and data path to first store the flags from CMP instructions, then to modify the control logic sections to account for the conditional bits within the instruction to implement the new instructions.

Following the implementation of the new instructions, the modules were compiled in Quartus and then tested in ModelSim by varying the Instruction input based on the memfile2.dat file.

Before testing, the PC sequence was written out to determine what set of instructions should have been taken based on the conditional branching.

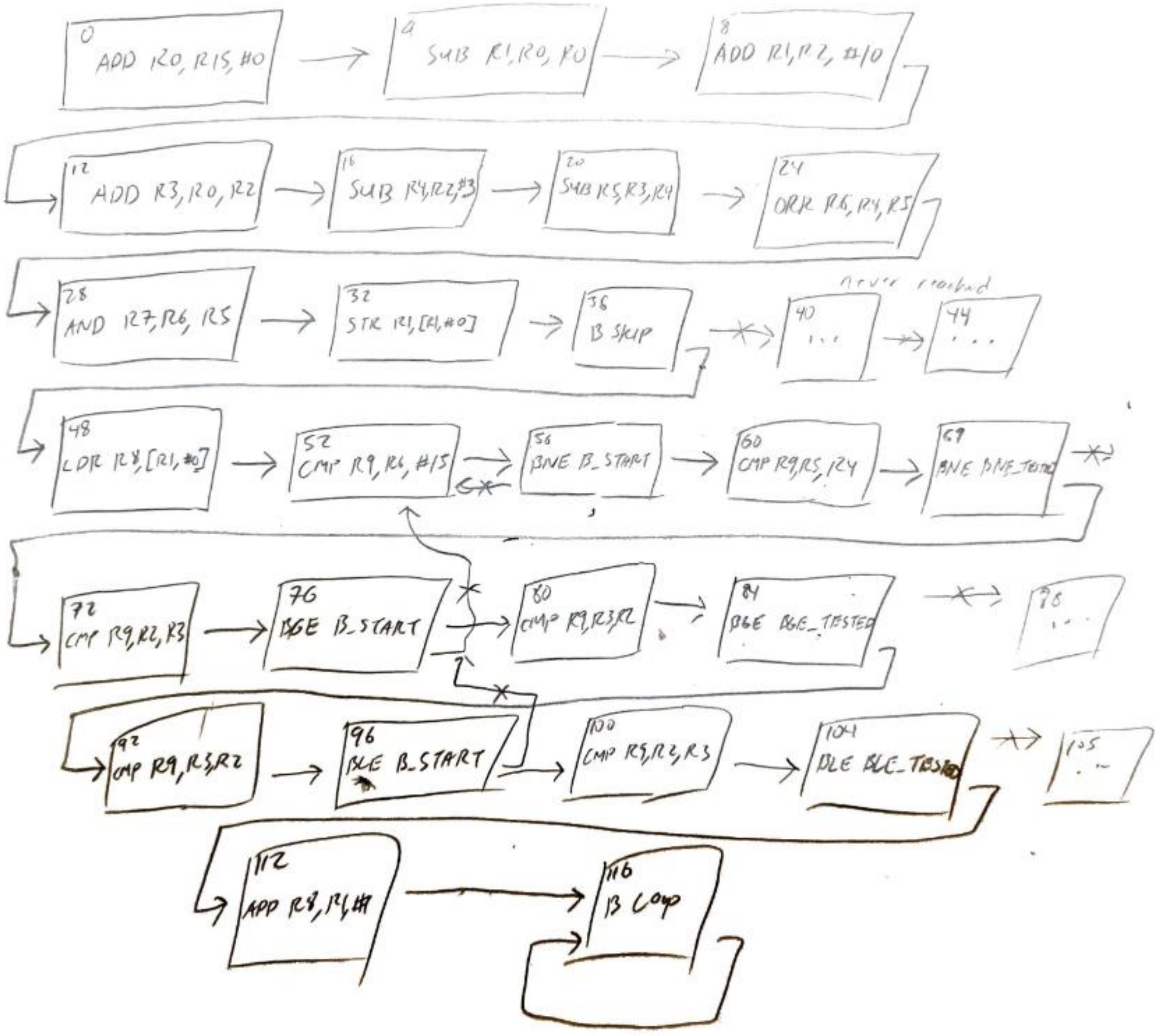


Figure 3: This is the PC sequence for memfile2.dat

The PC sequence displayed above in Figure 3 was compared with the simulation results to determine if the instructions were implemented correctly.

Results

Task #1:

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

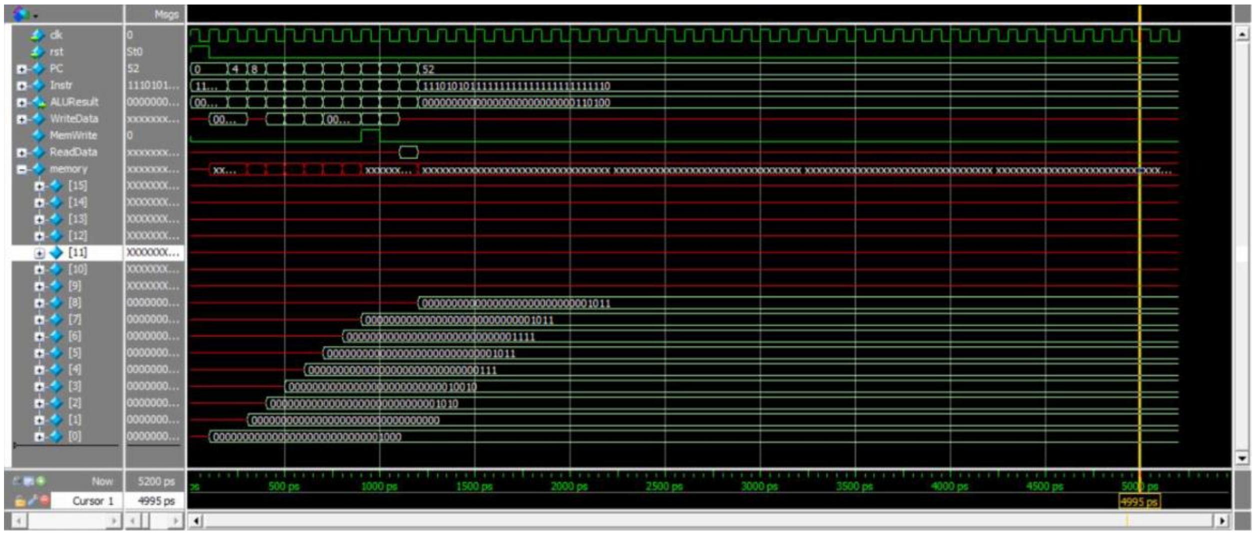


Figure 4: The waveform generated for the task 1 ARM module testbench (memfile.dat)

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

Task #2:

After implementing the new instructions in Quartus, I ran Modelsim to test it.

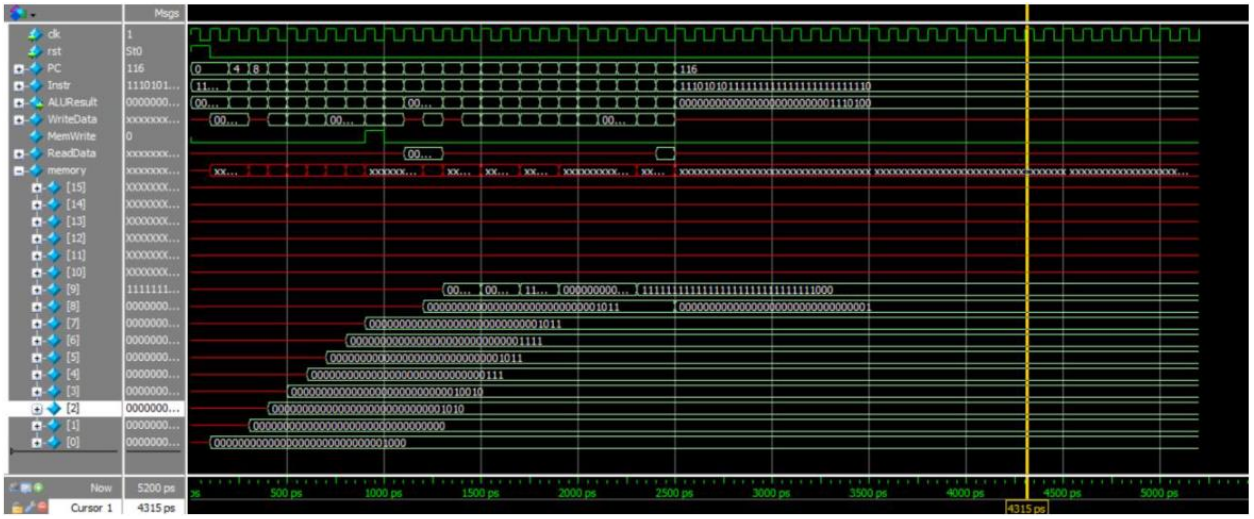


Figure 5: Modelsim waves for testing CMP, B EQ, B NE, B GE, B GT, B LE, B LT when implemented (memefile2.dat)

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

Appendix

See the following list for the order:

top.sv

testbench.sv

dmem.sv

imem.sv

alu.sv

alu_testbench.sv

arm.sv

fullAdder.sv

reg_file.sv

reg_file_testbench.sv

singleALU.sv

See the attached documents for the code: