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.

Diagram, schematic

Description automatically generated

*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

Table

Description automatically generated

*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.

Text, letter

Description automatically generated

*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.

Timeline

Description automatically generated with medium confidence

*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.

A screenshot of a computer

Description automatically generated with medium confidence

*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: