ECE649 Final Project Report

5/11/2024

Tannyr Singleterry, Zane Zollers

Table of Contents

Team Member Contributions……………………………………………………………………...3

Introduction………………………………………………………………………………………..4

CPU Hardware…………………………………………………………………………………….5

Describing Hardware Layout………………………………………………………….......5

Logisim Schematics……………………………………………………………………….5

CPU Software…………………………………………………………………………………….X

Intent Behind Code……………………………………………………………………….X

LCD Task…………………………………………………………………………………X

Conclusion………………………………………………………………………………………..X

Team Member Contributions

Both: We both decided on what different extra credit challenges we wanted to do based on how much knowledge we had of design beforehand. Decode module, ALU module.

Tannyr: Program Memory module, some of the control lines, and report.

Zane: Registers module, Data Memory module, most of the control lines, and test code.

Introduction

For this project, we were tasked to build a CPU that supports a subset of RISC-V instructions. For this project we used Logisim-Evolution to handle the design work, and Ripes to write test code to port over to Logisim. The base requirement tasks for this project are as follows: 32-bit instruction set architecture (ISA), at least 8 registers, an Arithmetic Logic Unit (ALU), instruction support for RISC-V instructions (ADD, SUB, ADDI, LW, SW, BEQ), LED output, and testing.

We were also given the opportunity to do advanced tasks for our CPU. These advanced tasks in most cases for us were just a basic expansion of the basic tasks and we felt they could be implemented without much difficulty. The advanced tasks we decided to implement are: support for 32 registers for addi/add/sub/lw/sw, support for R-Type instructions (and, or, xor), 3 stage pipeline support, extending the LED matrix to 16 \* 16, Pong, and data hazard enhancement support. We felt like between the basic tasks and these advanced tasks, there will be enough of a challenge to be fun for our group but not too complicated that it felt unrealistic for us to complete the project.

CPU Hardware

For this section, we will be presenting detailed diagrams of our Logisim hardware designs and discussing key features and points of interest.

*Describing Hardware Layout*

The program will take in an instruction set and load it into the Program\_Memory module. Which will store all the instructions and load them accordingly. It will take these instructions and put them into the Decode module, which will fetch the instructions and decode them into their bits for the registers. The Registers module will take in addresses from the Decode module and will store and manipulate the instructions provided based on what we need. The ALU module performs any arithmetic provided via the instructions. The Data\_Memory module acts as RAM for our CPU.

A diagram of a circuit

Description automatically generated*Logisim Schematics*

Figure 1: Main Module Diagram

A diagram of a circuit board

Description automatically generatedA diagram of a building

Description automatically generated

Figure 3: Decode Module Diagram

Figure 2: Program Memory Module Diagram

A computer screen shot of a diagram

Description automatically generated A diagram of a computer

Description automatically generated

Figure 5: ALU Module Diagram.

Figure 4: Registers Module Diagram. This is only the required 8 of the 32 implemented.

A diagram of a computer

Description automatically generated

Figure 6: Data Memory Module Diagram

CPU Software

For this section, we will outline the functions and purposes of our ASM programs. We will have four testing codes: the provided testing code, our own testing code for the advanced codes, code for Pong, and code to display the LED.

*Provided Testing Code*

Temp

*Advanced Tasks Test Code*

Temp

*Pong Code*

Temp

*LED Code*

Temp

Conclusion

Intro to section.