University of Missouri-Columbia

Lab 1: RISC-V Simulator

Group 1:
Chandler Tabor
Aleesa Hill
Cameron Ellis
ECE 4270

Date: February 19, 2023

Objective and Lab Description

The point of this lab was to write a simulator for RISC-V assembly language. We were given starter code, and had to implement two functions–one to print instructions, and one to actually handle them. Both of these functions needed to decode the instructions from memory.

Implementation

For both print() and handle() functions, we chose to write long switch statements. They started by checking the opcode of each instruction, which then calls separate functions for each instruction type. These functions also have switch statements that check the relevant func3 and func7 codes to determine the final instruction. Most of the required instructions were fairly easy to implement the instruction execution. For instructions that needed to access memory, we decided to initialize the stack pointer at the beginning of the program so we can access memory with ease.

Results

The first major hurdle we faced was simply understanding the starter code we had been given. This took the majority of the two lab sessions we originally had. There are many different parts to the starter code, with many of them having little description of their purpose to aid us in understanding them. Once we were able to understand the base code, we continued on to the full implementation.

After we accomplished that, we had a series of smaller milestones. The first of which was making the decision to break down instructions into each individual part, such as the opcode, registers, and immediate values. We broke down the code into each type of instruction, and from there it was fairly straightforward to write the print and handle instructions for R-type, and several I-type instructions.

Our next major problem was in figuring out how to handle the instructions that required making use of memory. In the end, we decided to just modify the initialize() function to set the initial value of the stack pointer and store it in register x2. This makes it easy to access memory later on by using the predefined stack pointer in our assembly code.

Work Distribution

Chandler:

Implemented most R-type instructions for both handle() and print() functions

Aleesa:

Figured out how a lot of the starter code works. Wrote code to decode the instructions.

Implemented some immediate instructions()

Cameron:

Implemented some immediate instructions for both handle() and print() functions, and load and store instructions for handle() and print().