Report for CSC3050 Assignment 1

Name: Liu Yuheng

ID: 120090263

Date: March 6, 2022

1 General Idea

MIPS uses 32 32-bit registers to store 3 types of instructions. Registers each have a unique number and corresponding usages. To simulate a MIPS assembler, we have to simulate registers and address first. When conducting one instruction, the information stored in the registers will be visited. Therefore, the dictionary data structure is needed to store the register and its information. And as 3 types of instructions have different formats, they need to be simulated separately corresponding to its format. Also, machine code is a binary number and address is usually stored as hexadecimal numbers, the transformation between different number systems is needed for simulation.

2 Breakdown of the problem

2.1 Basic Preparations

Considering that different instructions have different formats. I first use a map to store all forms of every instruction. The key is the name of specific instruction and the value is a tuple. For R type instruction， the tuple includes the instruction type, the function code, the operation code, and three more elements for specific registers.

For I type instructions, the tuple includes the instruction type, operation code, and three more elements for specific registers. For J type instructions, it includes the type and operation code. Another map called reg in phase2.py is used to store the information of different registers. The key is their specific name and the value is their corresponding number.

2.2 Process

First, a list is created to store lines from the original file. Then I conduct the first scan. I discard the information except for the text part. I then make some modifications, like delete '\t', '\n' and space. Considering that the label uses the same address as the instruction of its next line, I made a calculation to get the corresponding address of the label.

Another list is created to contain instructions instead of labels to conveniently get the index of instructions. Then the label's name and its address are added to a new dictionary for further referring. These are all operations in phase1.py.

For phase2.py, to deal with signed and unsigned numbers, I defined a function to calculate the 2's complement. Then I conduct the second scan.

This time I go through every instruction stored in the previous list and analyze the component. First, I refer to the dictionary to see the component of its first position, which is either 'rd' or 'rs' etc. therefore I can decide the corresponding number for that part. I keep doing this for all components of all instructions. I store the number for its corresponding part and then joined them together to form the machine code.

They are all added to a list. At last, by iterating the machine code list, I write them line by line to the output file. Other data type like strings are widely used in simulation to store necessary information.

3 Conclusion

This program successfully simulates the MIPS assembler, which will transform assembly language into corresponding machine code. It also passes the virtual machine test.

End of Report