Computer Architecture and Mobile Processor

Project 1 – Simple Calculator



Student Name : Jason Melvin Tedjokusumo

Student ID : 32205061

Major : SW 융합모바일시스템공학과

Professor : 유시환교수님

Submission Date : April 1, 2022

Free Days Left : 5 days

* Introduction

This is a project of making a simple calculator based on the important features of Von Neumann Architecture, which has memory, registers (working CPU), and program counter/instruction pointer. Basically, the calculator woks just like an ordinary calculator; however, following the principles of Von Neumann Architecture, the calculator is built with 10 registers to store and hold calculation values, program counter to point what instruction to execute, and memory to store the instructions provided by the input text file. I hope by doing this, I can understand the implementation of Von Neumann Architecture deeper and make a great Instruction Set Architecture (ISA).

* Important Concept

The important concept in building this project is: support 10 registers to hold some values and store the result in register 0, memory to store instructions or inputs which is a string type instruction from a text file, and also a program counter to know which instruction to execute and what next instruction to execute.

It has to support basic arithmetic binary operations such as +, - , \*, /, move, and halt operation. Additional operation can also be included, for instance, jump, compare, branch, and GCD calculation.

The input text file has to follow the instruction format which has the ‘opcode’ ‘operand1’ ‘operand2’ , where operand1 and operand2 can be hexadecimal or register number and executing one instruction in one line.

* Unique Considerations for Implementation

Because I use C programming language, for implementing this program I need fopen() and fclose() to open and close a text file. To read the line I use fgets() to get the string in one line instead of getline(). To split the string into opcode, operand1, and operand2, I use strtok().

* Unique Features

I use strcpy() to copy the memory instruction into another one, so that I can restore back the original string, because when using strtok(), the function change the original string, so whenever I want to Jump or Branch (making a continuous loop), the original file won’t change. Also, atoi() to turn a string into integer value after using strtok() to get the string of number.

* Build Configuration / Environment

I am using C programming language in VScode. First of all you need to install Vscode and mingw. Mingw is a C/C++ toolset, because Vscode doesn’t have a C compiler. After installing VScode, you have to download several extensions, which is C/C++ provided by Microsoft and its extensions and Code runner to run codes.

Sometimes when you want to use scanf() function, you need to insert something to the terminal, and we can use run in terminal to give the output in the terminal. Click on File > Preference > Settings > Extensions > Run Code Configuration > scroll down to Run in Terminal and check it. That’s it!

* Working Proofs

Below is the input.txt file, containing the inputs and instructions to do.

A screenshot of a computer

Description automatically generated with medium confidence

Below is the results of each instructions pointed by the program counter from the beginning to the end of file.

A screenshot of a computer

Description automatically generated with medium confidence

* Trial & Errors

While doing this project, I have several trials and errors, mostly because I don’t understand the syntax of the functions, such as strtok() and fgets(). I also couldn’t use getline() function which is in the c <stdio.h> library, because of different POSIX or platform (I don’t really understand about this), so I stick to fgets() function. However, when I try again and again and again, I begin to understand fgets() and strtok() functions and it becomes so easy to write the codes.

* Final Thoughts

I personally think that this is a good project to understand the Instruction Set Architecture as well as the Von Neumann Architecture. It’s also a good practice of research to know better of the programming language that I use. I first thought that this will be easy, but it wasn’t that easy, and it wasn’t that hard either. Still, this is a good experience to improve my programming skills.