Computer Architecture and Mobile Processor

Project 2 – Single Cycle MIPS emulator



Student Name : Jason Melvin Tedjokusumo

Student ID : 32205061

Major : SW 융합모바일시스템공학과

Professor : 유시환교수님

Submission Date : April 29, 2022

Free Days Left : 0 days

* Introduction

This is a project of making a MIPS emulator that can run a program within a single cycle (excluding floating point operations), with the 2 important features of Von Neumann Architecture, which is stored program, and sequential execution, meaning that assuming we have memory, and the program is loaded in memory before execution. MIPS is a simple microprocessor that swept last decades, it has simple Instruction Set Architecture (ISA), and optimized for concurrent parallel execution for performance. I hope by understanding the concept of ISA of MIPS, I can understand the ISA of others such as INTEL. And hopefully, I can make my own in the future.

* Important Concept

The important concept in building this project is: understanding the ISA of MIPS and how some assembly code works. Also, the 2 important features of Von Neumann Architecture would also be very helpful. Single cycle machines take only a single clock cycle which mean it has a long clock cycle time, and all state updates made at the end of an instruction’s execution.

MIPS instructions are categorized into I, R, and J types, and MIPS ISA defines 32 general-purpose registers. MIPS will take executable binary (input.bin) and emulates the operation in a single cycle. Fundamentally, MIPS are divided into 5 phases, which are instruction fetch (IF), Instruction decode and register operand fetch (ID/RF), execute/evaluate memory address (EX/AG), memory operand fetch (MEM), store/writeback result (WB). Additionally, in every instruction processing engine, consists of 2 components, which is data path, and control logic. Data path is the hardware structures (functional units that operate on data) / storage units that store data. While control logic is the one who control the signal given for the hardware or engine to operate; in other words, signals that specify what the data path elements should do to the data.

* Unique Considerations for Implementation

Because MIPS is a 32-bit microprocessor, so I have to consider all the inputs and outputs and all the operations inside works or calculated in 32-bits, this is kind of hard because I encounter a lot of overflows from converting data types. Also, when retrieving instructions from binary input file, the way to read the instructions are swapped, so I have to reconstruct the instructions first to be able to assign every part to the variables according to the instruction format.

* 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.

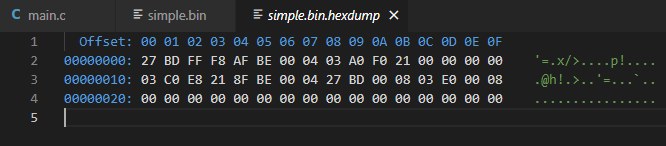
Sometimes when you want to use scanf() function, you need to insert something to the terminal, and we can use run in terminal to give the output in the terminal. Click on File > Preference > Settings > Extensions > Run Code Configuration > scroll down to Run in Terminal and check it.

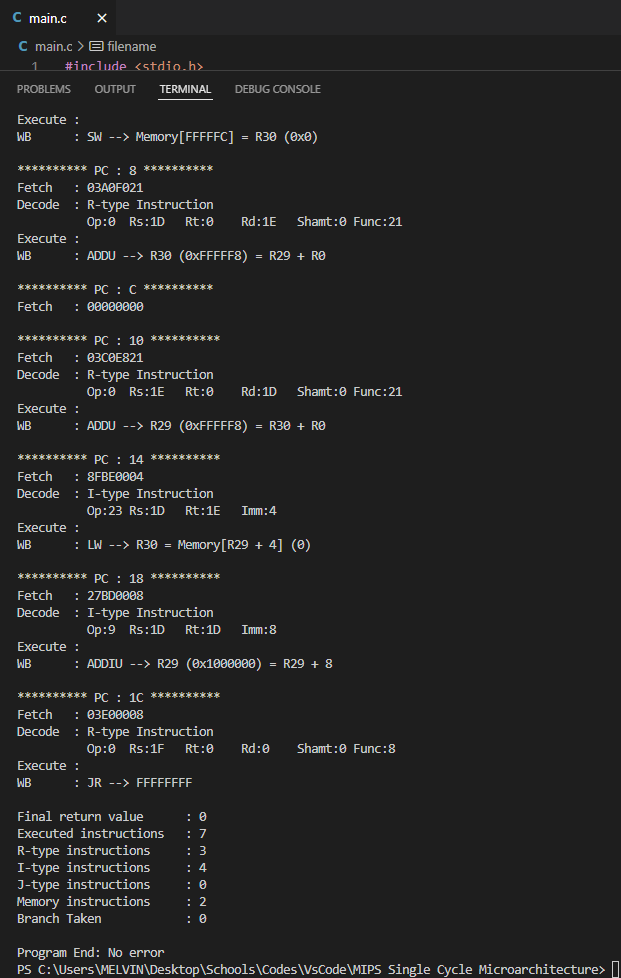
Because we will handle binary input file, so we have to install hexdump for Vscode in the extensions. This extension could show the binary input (which is impossible for human to read) in a unique hexadecimal format.

* Working Proofs

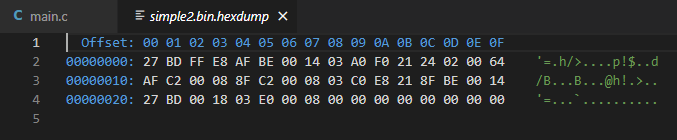
Below are the binary input files showed in hexadecimal dump file and the results of each binary input file. Because the results and some input binary files are too long, I will crop some of it and present the final results and print outputs.

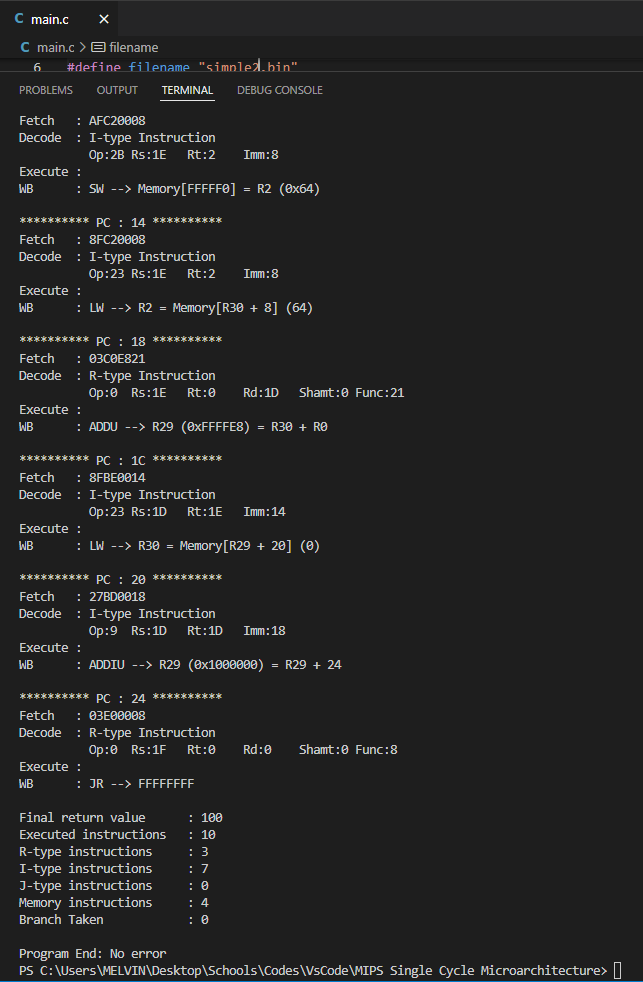
*Simple.bin*

**

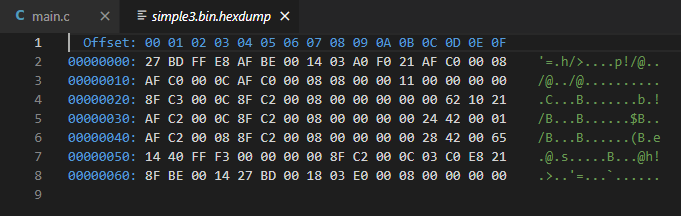


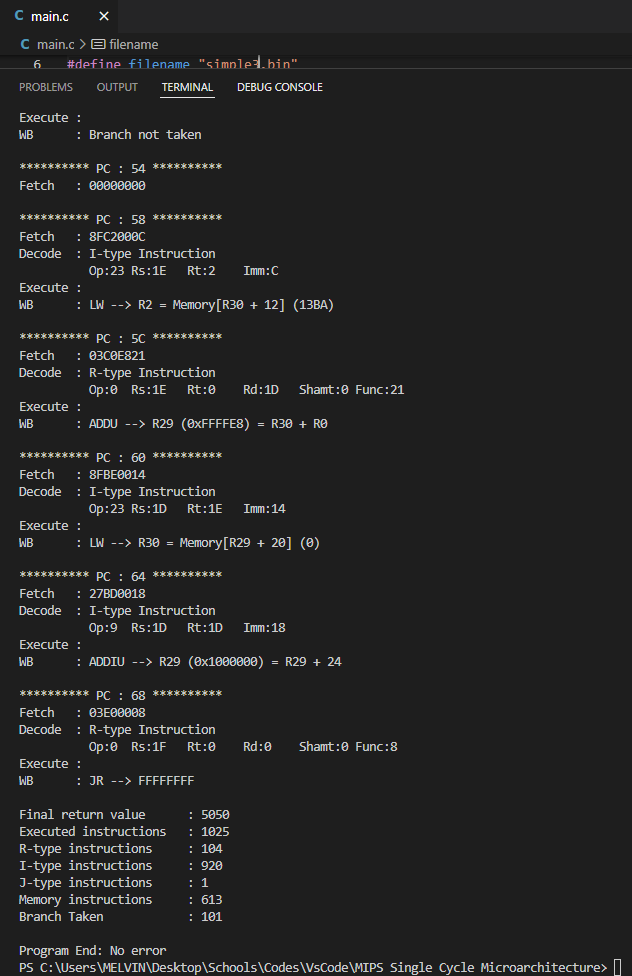
*Simple2.bin*



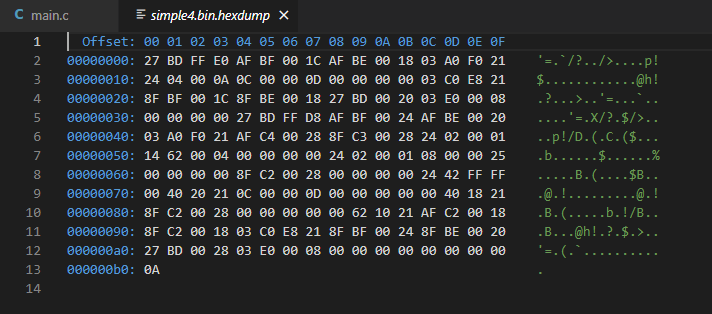


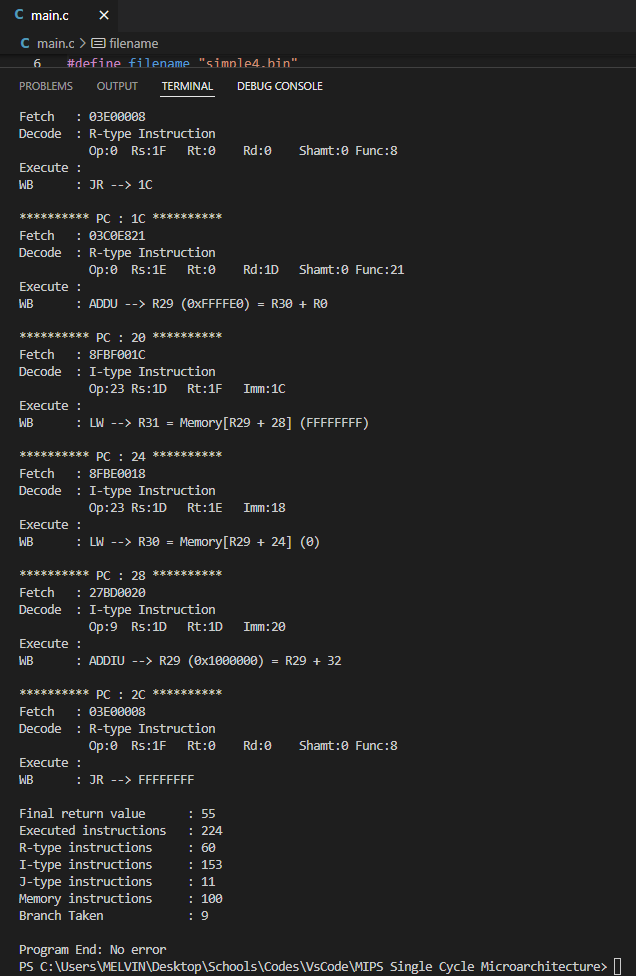
*Simple3.bin*

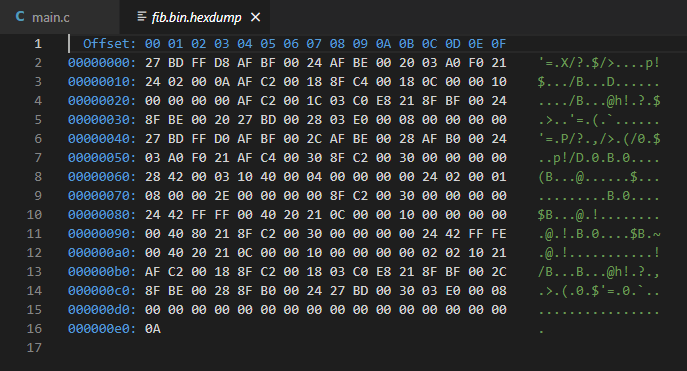


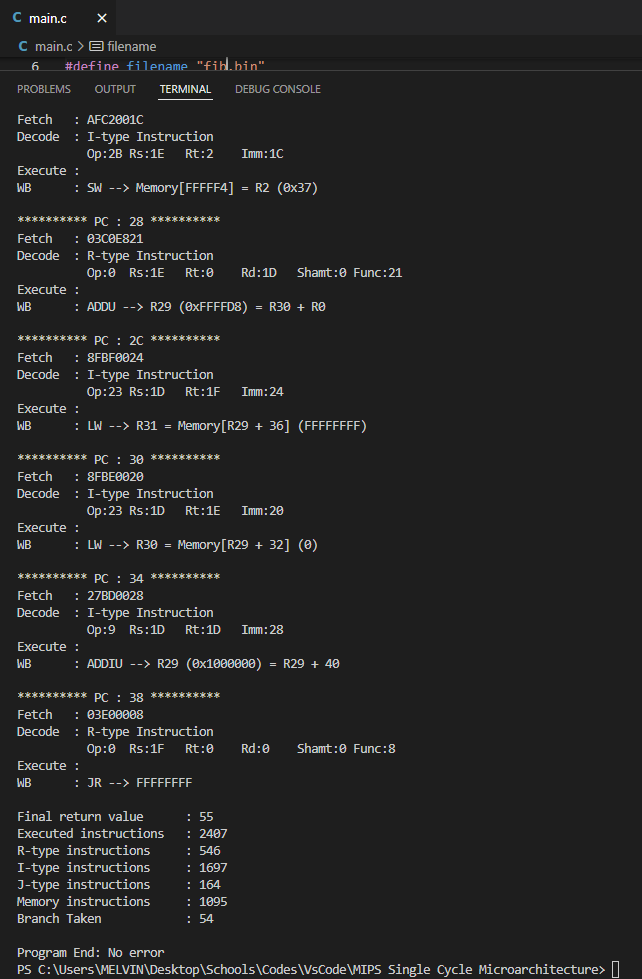


*Simple4.bin*

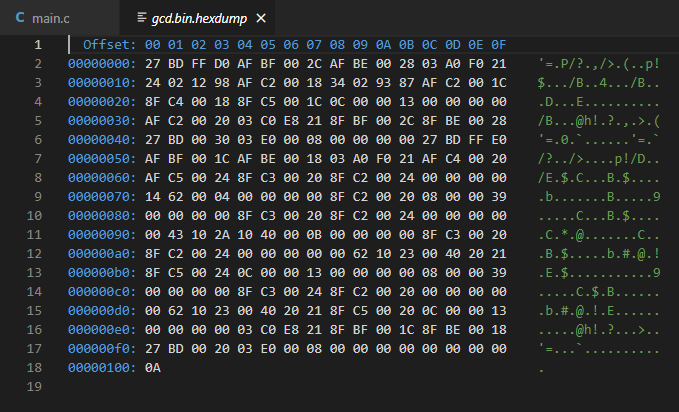
**

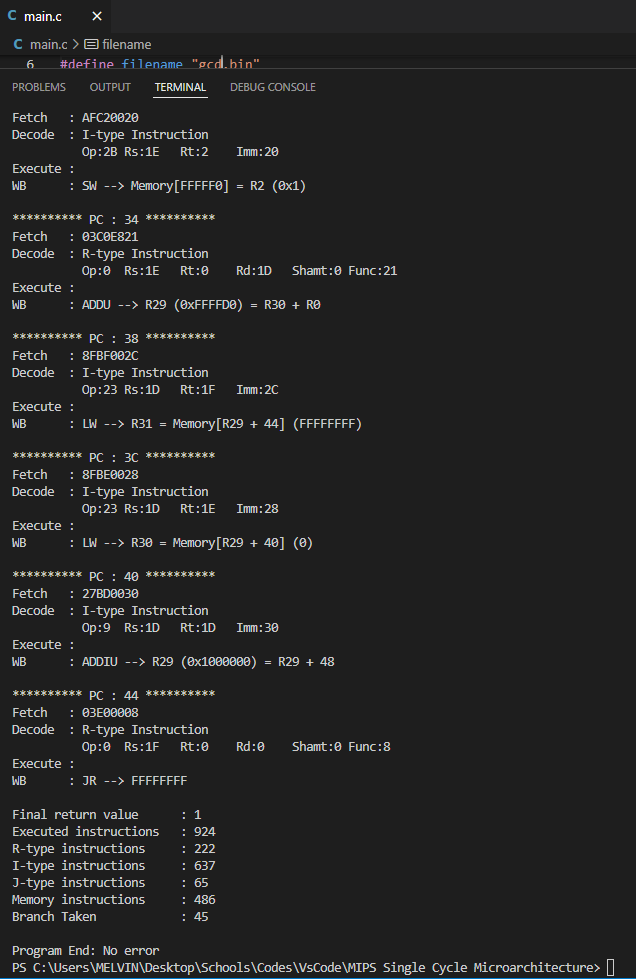


*Fib.bin*

**

*Gcd.bin*

**

**

* 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 fread() and memset(). I also got confused while constructing the ISA, what comes first and second and third, etc. Handling the swapped instruction also a hassle to do. Every time I feel that its completed and will work fine, it doesn’t. The code stops somewhere (this is where I encounter the most, and it is because of the ‘overflow’ problem), the result is absurd, or it is looped infinitely. I could finish all of the sample programs, some works fast, some results are with too many instructions, but the answer is still correct. However, I couldn’t finish and figure out gcd.bin and input4.bin, the result for gcd is wrong (its only 1) and input4.bin just looping endlessly. I don’t know where went wrong, and I have tried my best for this project.

* Final Thoughts

I personally think that this is a good project to understand the Instruction Set Architecture and the inner workings of how a computer execute or processing a program or inputs or instructions. It’s also a good practice of research to know better of the programming language that I use. I know that this is going to be a very hard project, but fortunately it works fine at the end.