LAB 3. Single Cycle CPU

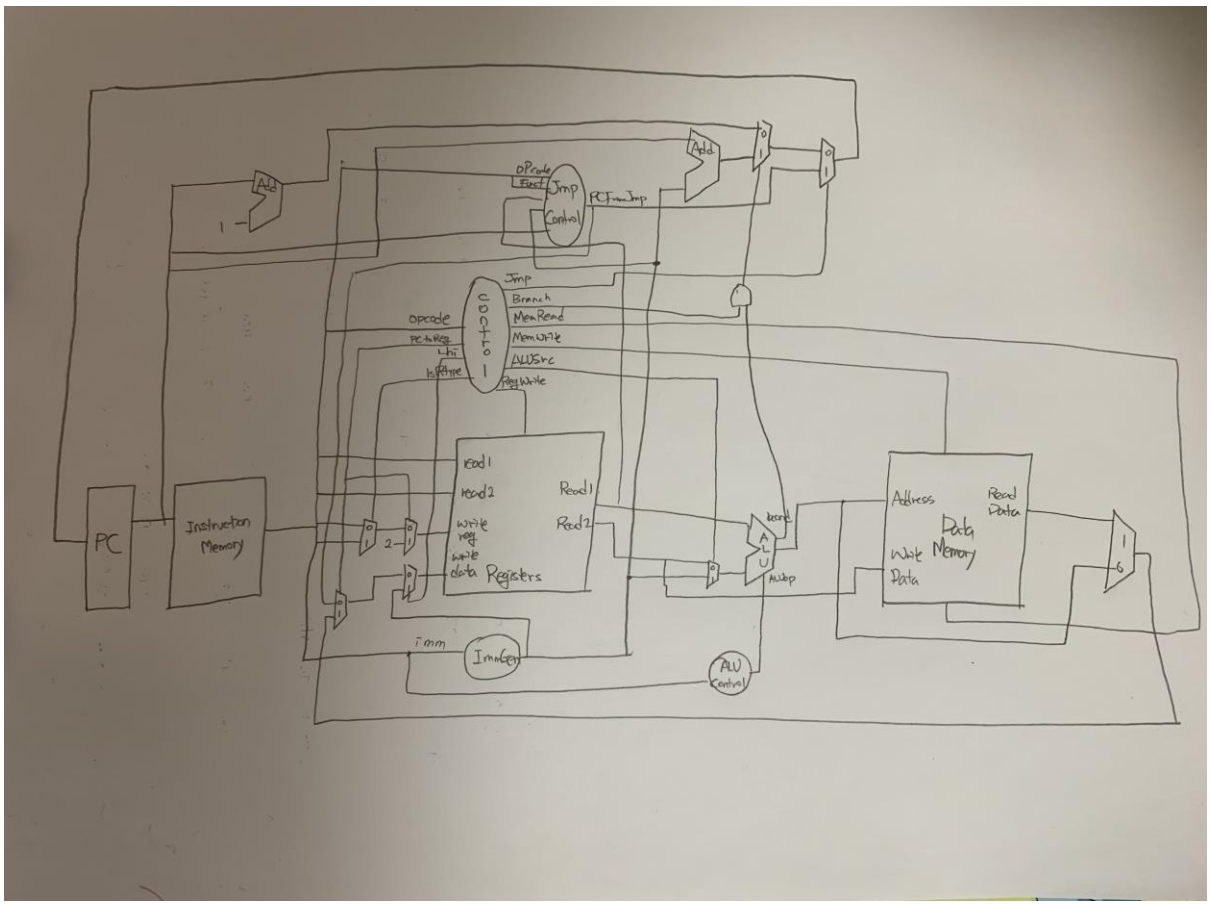
201800038 박형규, 20180480 성창환

1. Introduction

이번 Lab에서는 single-cycle TSC CPU를 구현하는 것을 목적으로 하였다. TSC_manual.pdf에 나와있는 내용에 따라 각각 R type, I type, J type에서 올바른 action이 수행되도록 하였다. Single Cycle CPU를 구현하면서 수업에서 배운 Single Cycle CPU의 구조와 작동하는 원리를 익히고, 이후 Lab에서 구현해 볼 Multi Cycle CPU의 기초가 되는 CPU를 구현하는 것이 목표이다.

2. Design

먼저 TSC_manual과 RISC-V의 Single Cyle CPU의 설계도를 이용하여 single cycle TSC CPU의 설계도를 그려보았다. 설계도는 아래 그림과 같다.



설계도에 의거하여 여러 모듈들을 만들었다. 모듈은 ALU, ALUControl,. InstructionDecoder, Registers, ImmGen, MUX16, MUX2, JmpControl, Control, cpu이다. 다음은 각 모듈에 대한 대략

적인 설명이다.

ALU : 16비트의 input들을 이용하여 기본적인 연산을 수행해 주는 모듈이다.

ALUControl : opcode와 InstFunc에 따라서 수행되어야 할 ALU연산을 알려준다.

InstructionDecoder : instruction을 받아와서 각 bit에 해당되는 부분에 맞도록 파싱해준다.

Registers : 레지스터들을 관리하는 모듈로 레지스터를 읽기도 하고 RegWrite에 따라 쓰기도 한다.

ImmGen : Immediate value를 사용하는 경우에는 opcode에 따라서 8bit의 imm를 16bit의 imm으로 전환하는 방법이 다르기 때문에 각 opcode에 맞게 8bit의 imm을 16bit로 확장시켜 준다.

MUX16, MUX2 : 각각 16비트와 2비트의 input과 output을 가지는 MUX 역할을 한다.

JmpControl : J type의 opcode가 실행될 때 이동하게 될 instruction의 위치를 알려준다.

Control : opcode와 funct에 따라 control bit를 설정해준다.

cpu : 메인 모듈의 역할을 하고, instruction memory와 data memory의 역할을 수행한다.

3. Implementation

4. Discussion

Single Cycle CPU를 구현하는 과정에서 이론적으로 배웠던 내용을 직접 코드로 구현하고자 하니 생각보다 세세한 부분에서 헷갈리는 내용들이 많았다. 대표적으로 load나 store을 실행할 때 어느 때에 readM과 writeM을 수정해야 하는지, regwrite을 언제 활성화시켜야 하는지 등이 대표적인 문제였다. 처음에 이런 것들을 제대로 고려하지 않고 구현하였더니 여러 값들이 제대로 된 값을 가지지 못하는 것을 보고 이런 세세한 부분이 컴퓨터의 구현에 있어서 매우 중요함을 다시금 깨닫게 되었다.

5. Conclusion

이번에 Single Cycle TSC Cpu를 구현해 보면서 이론상으로 배운 Single Cycle CPU의 작동원리와 구조에 대해 좀 더 자세히 익힐 수 있었다. 구체적으로 IF, IM, EX, Mem, WB에서 각각 어떤

역할이 수행되는지, 어떠한 체계를 가지고 각 모듈이 작동되어 CPU를 구성하는지에 대해 심도있게 공부 할 수 있는 시간이 되었던 것 같다.