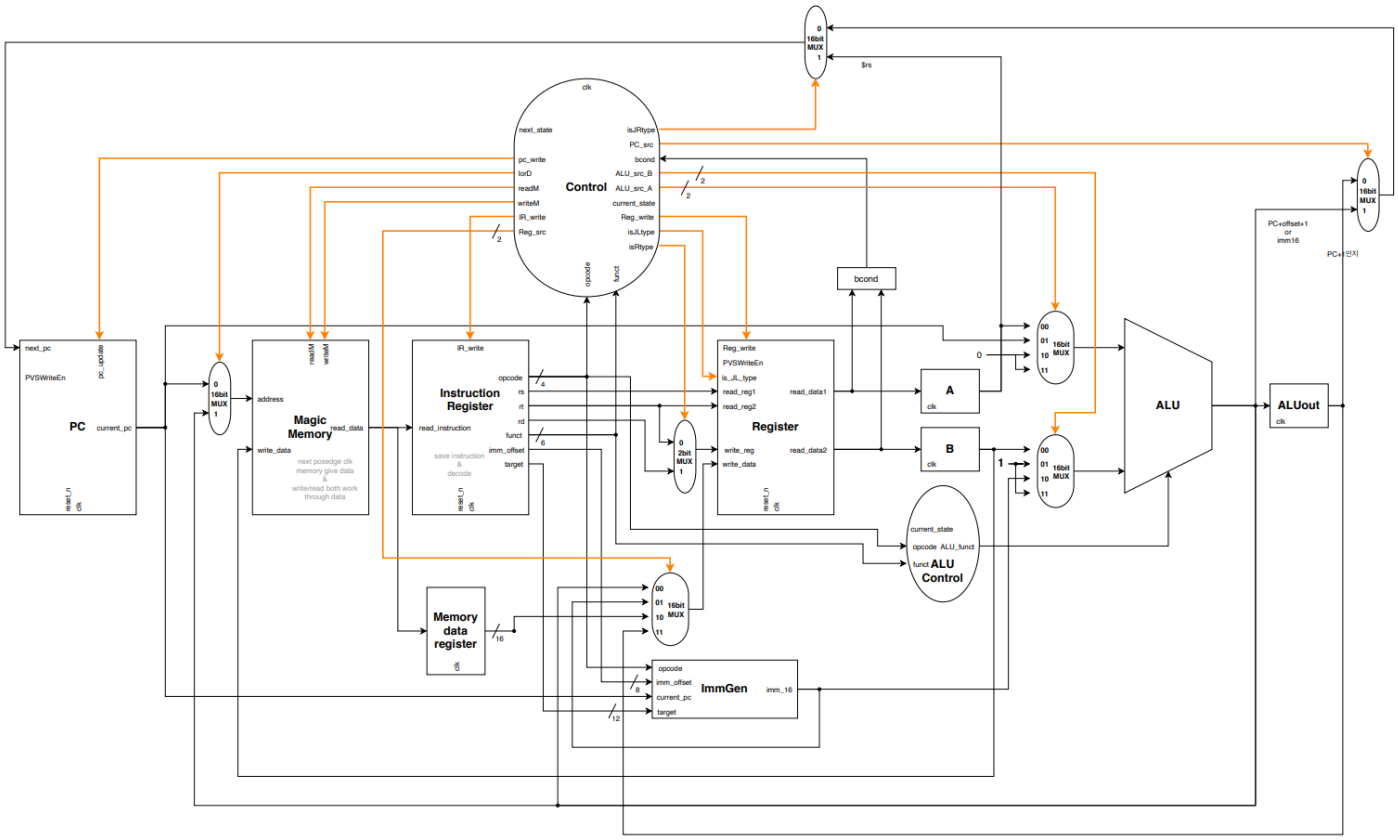
LAB 4. Multi Cycle CPU

201800038 박형규, 20180480 성창환

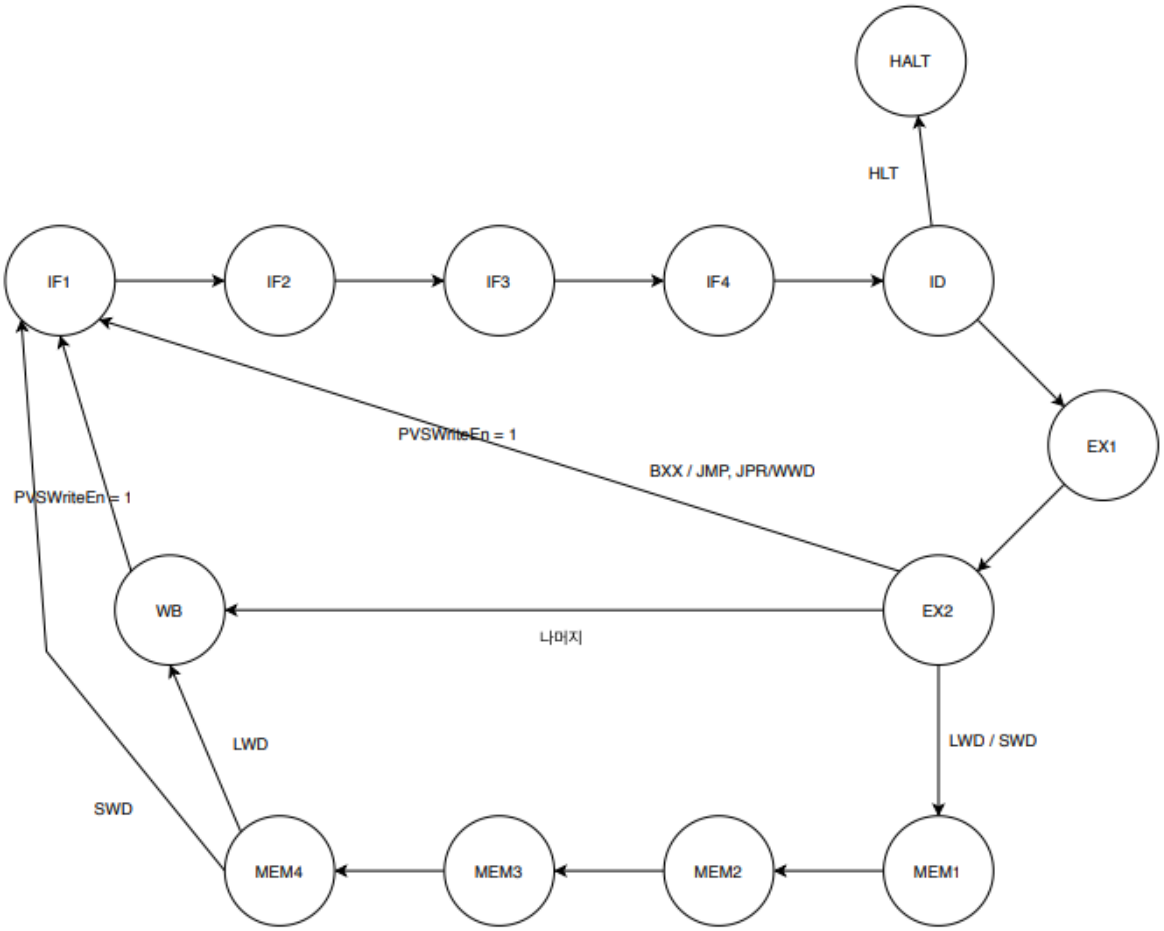
1. Introduction

이번 Lab에서는 multi-cycle TSC CPU를 구현하는 것을 목적으로 하였다. TSC_manual.pdf에 나와있는 내용에 따라 각각 R type, I type, J type에서 올바른 action이 수행되도록 하였다. Multi Cycle CPU를 구현하면서 수업에서 배운 Multi Cycle CPU의 구조와 작동하는 원리를 익힌다.

2. Design



이전에 구현했던 single cycle cpu에서 수업시간에 배웠던 몇가지 회로들을 추가하고, nextPC계산을 ALU를 통해 진행하도록 구현했다.



또한 위의 그림과 같이 multi cycle 구현을 위한 FSM을 구현했다.

3. Implementation

Multi cycle CPU를 구현하기 위해 verlilog에서는 Design에서 제시한 FSM과 회로도에 맞게 코드를 작성하였다. 먼저 .v 파일을 여러 개로 분리하였는데, cpu라는 main module을 가지고 있는 cpu.v, control 부분을 담당하는 control.v 그리고 나머지 datapath 부분을 담당하는 모듈들이 포함된 datapath.v가 있다.

먼저 cpu.v의 cpu 모듈에서는 여러 input, output, reg선언과 wire 연결이 이루어졌다. 그리고 branch condition을 check하는 bcond가 cpu에서 결정되었으며 //initial part에서는 초깃값 설정, //general part에서는 reset_n일 경우에는 초깃값 설정과 동일하게, 이외의 경우에는 state를 업데이트 해주고 next_state가 IF1일 경우에는 num_inst또한 업데이트 해주었다. //when halt part에서는 halt가 일어나는 상황이면(current_state == `ST_HLT) is_halted를 1로 켜주었고 //WWD part에서는 WWD instruction에 따라 결과값 출력이 발생하도록 하였다.

다음으로 datapath.v를 살펴보자. ALU module에서는 ALU_func에 따라 어떤 연산을 할 것인 것 결정을 해 주도록 하였다. Instruction register module에서는 instruction을 받아 와서 파싱 작업을 진행하고, IR_write이 켜진 경우에는 read_inst를 inst로 업데이트 해 주었다. Imm_gen module에서는 각 opcode에 따라 필요한 imm이 다르므로 opcode에 따른 imm값을 16비트 값으로 확장시켜주는 작업을 수행하였다. GPR module은 실제 레지스터를 관리하는 모듈로 PVSWrtieEn신호와 RegWrite이 모두 켜져 있을 경우에 JL_type일 경우 REGISTER[2]에 write_data를 적고, 이외의 경우에는 REGISTER[write_reg]에 write_data를 적었다. program_counter 모듈은 pc값을 관리하는 모듈로 PVSWriteEn신호와 pc_update가 모두 켜져 있으면 current_pc를 next_pc로 업데이트 해주었다. 다음으로 buffer module은 레지스터에서 나온 값을 버퍼링 해주는 모듈이다. 그리고 alu_out_buffer은 alu에서 나온 값을 버퍼링 해주는 모듈이다. 그 이후에 MUX2_2, MUX16_2, MUX16_4 모듈은 각각 2bit짜리 2개 선택지 MUX, 16비트짜리 2개 선택지 MUX, 16비트짜리 4개 선택지 MUX를 구현한 것이다.

마지막으로 control.v를 살펴보자. ALU_control에서는 opcode와 inst_func값에 따라 해당 instruction이 ALU에서 어떤 함수를 써야 하는지 할당해주는 module이다. Main_control module은 수 많은 control signal을 보내주고, next_state를 설정해주는 module이다. 여기서 발생하는 signal을 결정하는데에 조금 어려움이 있었으나, 우리가 만든 FSM과 simulation에서 wave를 분석하며 각 signal에 알맞은 기능을 구현할 수 있었다.

4. Discussion

Single cycle cpu와는 달리, 더 많은 wire들이 추가되었고, 추가적인 register들이 추가되면서 각 레지스터들의 값을 동기화 하는 시점등에 관련된 문제들이 코드를 디버깅 할 때 많이 일어났다. 이런 문제들을 해결하면서 multi cycle cpu에서 각 register들이 어떤 타이밍에 어떻게 동작해야 하는지 제대로 알게 되었다.

5. Conclusion

이번 랩을 통해서 multi cycle cpu를 성공적으로 구현했다. Multi cycle cpu의 각 isntruction들의 다른 실행시간을 어떻게 구현하는지 알게 되었다. 또한 이번 랩을 통해 Multi cycle cpu와 single cycle cpu의 차이점에 대해서 더 잘 알게 되었다.