EE312 Lab3 Report

20180155 김준범
20160791 권용빈

Ⅰ. Introduction

Lab3는 RISC- V의 uArchecture을 공부하여 37개의 instruction을 수행할 수 있는 Single Cycle Cpu를 만드는 것이 목적이었다. CPU를 구성하는 Register, Memory, Clock , ALU, PC, Control Signal에 대한 상호작용을 이해해야 하며 ALU, PC, Control Signal에 대한 부분을 implement해야 했다.

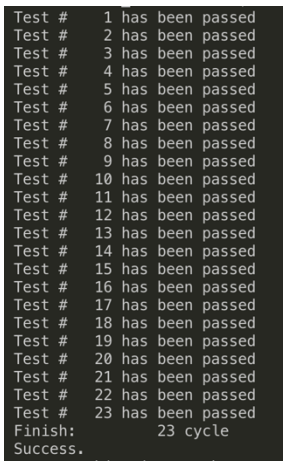
Ⅱ. Design

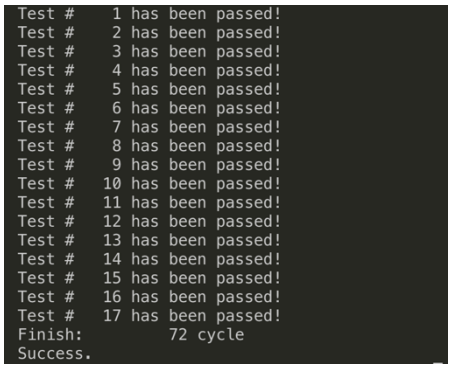
Single-cyle CPU를 module화를 하지 않고 구현하였다. 그러므로 자세한 설명은 implementation부분에 서술하였다.

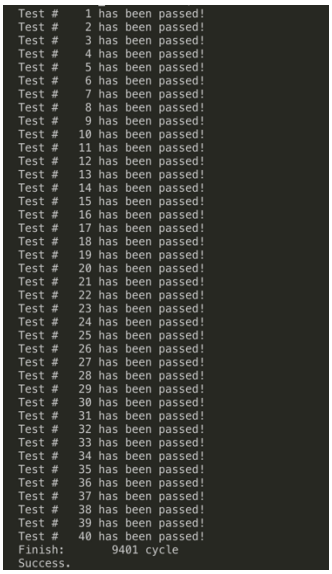
Ⅲ. Implementation

RISCV_TOP.v 파일에서 implement 한 부분은 총 3 개의 always 문으로 구성되어있다. Combinational always Block 에서는 I_MEM_DI 를 decode 하는 부분, decode 한 결과에 따라서 control signal 을 배정하는 부분, register read, memory read 에 대한 address 를 설정하는 부분이 있다. register 와 memory 의 read 는 aysynchronous 하기 때문에 address 를 설정한 즉시 데이터를 읽어오고, 이때 읽어온 데이터를 다른 reg 변수에 저장하여 다른 block 에서도 사용가능하게 했다 . sequential always 부문 중 하나는 CLK negedge signal 마다 실행되는데 이 부문에서는 combinational always 문에서 저장했던 register read, memory read 값으로 ALU 연산을 수행한 뒤 register 나 memory 에 다시 store 하고 PC 의 값을 변화하는 역할을 수행한다. 마지막 남은 sequential always block 은 CLK posedge 마다 실행되며 pc 를 실질적으로 업데이트 해주는 역할을 한다.

Ⅳ. Evaluation







위의 스크린샷에서 볼 수 있듯이 inst, forloop, 과 sort, 총 3개의 test bench의 모든 test case들을 통과한 것을 볼 수 있다. 3개의 test bench가 모든 instruction을 적어도 한번씩 test하진 않았지만 동일한 opcode안에서 다른 function3를 가진 instruction들이기 때문에 큰 문제는 없을 것이라 판단된다. 그러므로 이번 과제는 성공적으로 마무리 되었다고 해도 무방할 것 같다. 하지만 alu연산, control signal setting등의 세부적인 부분들을 module화 하지 않고 한 file에서 여러 개의 always문으로 나눠 single-cycle CPU를 구현하였다는 점이 다소 아쉽다.

Ⅴ. Discussion

처음에 코드를 작성할 때 하나의 negedge always 문으로 시작을 했고 그 안에서 register read, write 과 ALU 연산, pc 업데이트를 모두 수행하도록 코드를 작성했다. 하지만 register 의 read 와 write/memory 의 read 와 write 이 같이 수행이 안되는 것을 확인했고 두 개의 always(posedge, negedge)로 작업을 분리했는데 always block 의 연산 순서가 이상했으며 PC 값의 업데이트도 논리의 흐름처럼 진행되지 않았다. 그래서 여러가지 시도 끝에 read 에 관련한 asynchronous 연산을 따로 asynchronous 한 always 에서 해결한 뒤 그 값을 변수에 저장해 다른 always 구문에서 사용하는 방법을 선택했다. 이 문제가 발생했던 원인들을 아직 완전히 파악하진 못했지만 아마도 각각의 기능(ALU, Control Signal, Pc)를 모듈화 하지않고 하나의 파일안에서 해결하려고 했던 것이 원인일 것이라고 생각한다.

Ⅵ. Conclusion

연산과 동작이 많은 만큼 module화를 해주지 않는다면 설계한 회로를 구현하는 것이 상당히 까다롭다는 점을 몸소 느꼈다. 그러므로 다음 과제인 multi-cycle cpu를 구현함에

있어서는 우선 single-cycle cpu를 다시 module화하여 구현한 이후에 진행하는 것이 바람직해 보인다.