**CSED311 Lab 2: Single-Cycle CPU**

Team number 3

20210643 김현준

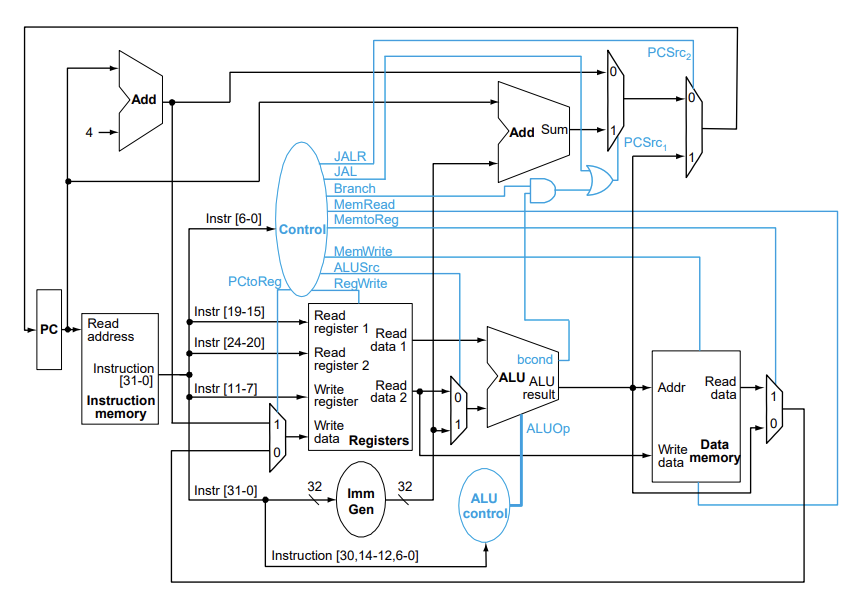
20210273 하태혁

1. Introduction

Single-cycle RISC-V CPU는 단일 클록 사이클 안에서 RISC-V 명령어를 실행하는 CPU의 구현 방법 중 하나이다. 이 보고서에서는 Single-cycle RISC-V CPU의 개요와 동작 방식에 대해 설명하고 이를 구현하기 위해 작성한 코드에 대한 전반적인 설명을 제시한다.

Single-cycle RISC-V CPU는 하나의 Instruction을 실행하는 데 필요한 모든 단계를 하나의 사이클 안에서 수행한다. 하나의 Instruction이 실행되는 동안 고정된 수의 사이클이 작동하고 모두 완료되었을 때, 다음 Instruction이 실행된다. 아래 Design에 나타나 있듯이 간단한 디자인으로 구성되며, 전력 소비와 복잡성을 줄이는 것을 목표로 한다.

1. Design



CPU module에는 positive reset signal, clock signal, finish simulation signal이 들어오며, 서브 모듈에 input, output signal을 전달하기 위한 wire와 register를 선언하고 각 Unit에 연결하는 것이 전체적인 CPU module의 구조이다. 서브 모듈에는 Program counter, Instruction memory, Register file, Control unit, Immediate generator, ALU control unit, ALU, Data memory 가 존재한다.

1. Implementation

먼저 Single-cycle RISC-V CPU의 동작 방식을 설명하면 다음과 같다.

* IF(Instruction fetch) : Instruction에 의한 Write Back이 이루어 진다. Instruction 주소는 PC register에 저장되어 있다.
* ID(Instruction decode and operand fetch) : Fetch된 Instruction을 해독하는 과정을 거친다. Instruction의 bit로부터 decode하여, opcode, funct3, funct7 등을 결정한다.
* EX(ALU/execute) : Instruction으로부터 지정된 연산을 ALU에서 수행한다. 결과는 Register에 저장된다.
* MEM(Data Memory access) : Instruction type에 따라 메모리에 접근한 뒤, 데이터를 저장하거나 가져오는 작업을 수행한다.
* WB(Write-back) : 연산 결과를 register에 저장한다.

한 개의 Instruction을 실행하기 위해서는, 위의 5 generic steps을 따르게 된다. 위 과정을 하나의 사이클 안에서 수행하기 때문에 Single-cycle RISC-V CPU는 비교적 단순하며 예측 가능한 성능을 보이게 된다.

다음으로 Single-cycle RISC-V CPU를 구성하는 서브 모듈에 대해 설명하면 다음과 같다.

1. Program Counter (PC): 다음에 실행할 Instruction의 메모리 주소를 저장한다. clk signal이 들어올 때마다, 다음 instruction이 존재한다면 해당 주소로 current\_pc가 업데이트 된다.
2. Instruction Memory: 프로그램의 Instruction들이 저장되는 메모리 영역이다. PC에 의해 가리키는 주소의 Instruction을 output으로 내보낸다. clk signal에서 reset될 경우 Instruction memory를 0으로 초기화한다.
3. Register File: 프로세서에서 사용되는 레지스터들의 집합으로 레지스터들은 데이터 저장, 연산의 입력 및 출력을 위해 사용된다. Asynchronously하게 register file로부터 데이터를 읽어와 rs1\_dout과 rs2\_dout에 저장하며, x17 레지스터 값이 10일 경우 rf17 값에 저장하여 이를 다른 Unit으로 전달한다. 이후 is\_ecall이 1이고 x17이 10일 경우 is\_halted가 1이 된다.
4. Control Unit: opcode의 해독 결과에 따라 control value에 각각 해당하는 제어 신호를 전달한다. Register file, ALU, Data Memory 등에 전달되어 다음 연산이나 실행을 결정한다.
5. Immediate Generator: Instruction에서 값을 추출하여 ALU 또는 다른 부분으로 전달하는 역할을 한다. Instruction type에 따라 asychronize하게 값을 재배열하여 사용한다. ALU에서 연산에 사용되거나, 분기 주소 계산에 사용된다.
6. ALU Control Unit: Instruction type에 따라 ALU가 수행할 연산의 종류를 결정하는 모듈이다. opcode에 따라 type을 구별하고 이후 funct7, funct3 값을 통해 다음 ALU연산을 결정한다.
7. Arithmetic and Logic Unit (ALU): 산술 및 논리 연산을 수행하며, ALU Control Unit의 지시에 따라 연산을 수행한 후 결과를 레지스터 파일에 저장하거나, Data Memory에 쓰기 작업을 수행한다.
8. Data Memory: 프로그램이 사용하는 데이터를 저장하는 메모리 영역이다. read, write 값에 따라 데이터를 읽어서 output으로 내보내거나 memory에 값을 저장한다.
9. Discussion

각 모듈에 필요한 wire와 register를 선언하는 부분에서 is\_harted을 고려하지 않고 연산을 수행했기 때문에, 시뮬레이션에서 시간이 무한대로 나오고 종료되지 않는 경우가 많았다. 해당 연산을 처리하기 위해서 Register File에서 x17 레지스터 값을 저장하는 wire를 선언하고 이후 control unit에서 is\_ecall 값을 계산하여 ecall을 호출하는 경우를 만들어 에러를 해결할 수 있었다.

1. Conclusion

간단한 Single-cycle RISC-V CPU를 구현하고 복잡성과 성능에 대해 생각해볼 수 있었다. Single-cycle CPU는 간단한 구조로 구성되기 때문에 간단한 Instruction을 처리하는 시스템에서는 효율적으로 작동할 수 있을 것이라고 생각해볼 수 있었다. 그러나, 한 사이클이 끝나는 동안 다른 Instruction을 처리할 수 없는 구조를 갖기 때문에 복잡한 Instruction을 테스트 벤치로 만들어 작동하게 된다면 비효율적이고 많은 시간이 걸릴 것이라 예상해 볼 수 있었다. 이후, Multi-cycle CPU, Pipelined CPU를 공부하고 코드를 짜보며, 이번 랩과제와 비교하여 생각해보며 CPU구조를 더 정확하게 이해할 수 있을 것이라 생각된다.