**Lab 3. Single-cycle CPU Lab**

**1. Introduction**

Single-Cycle CPU를 Verilog를 사용하여 실제로 구현한다. 이를 위해서는 Single-cycle CPU의 기본적인 동작 단계 (IF, ID, EX, MEM, WB)에 대한 이해와 설계가 필요하며, 세부적으로는 Control path와 Data path가 각각 어떤 동작을 해야 하는지 사전에 이해하고 설계하여야 한다. 무엇보다 어떤 Instruction으로 각 장치들을 동작 시키게 할 것인지에 ISA (Instruction Set Architecture)과 해당 랩에서 사용하게 될 32bit RISC-V ISA의 instruction이 어떤 구조를 가지는지 (Decoding Stage), 각 instruction이 어떤 동작을 요구하는지에 대한 이해가 선행되어야 앞선 설계가 가능하다. 해당 랩을 위해서 수업에서 배운 Single-cycle CPU의 구조를 기반으로 큰 설계도와 함께 Instruction 별로 각 요소들에 어떤 신호를 보내야 하는지에 대한 Truth Table, Immediate Value를 어떻게 합칠 것인지에 대한 사전 설계를 진행하였다. 이러한 사전 설계들을 기반으로 실제 구현을 하였다.

**2. Design**

Single-Cycle CPU에는 IF (Instruction Fetch), ID (Instruction Decode), EX (Execute), MEM (Memory Operation), WB (Write Back)이 모두 포함되어야 한다. 이는 사전에 주어진 명령문 (Instruction)을 동작하기 위해 필수적인 요소들이다. 이를 다음과 같이 회로도로 디자인하였다.

|  |
| --- |
|  |
| Figure 1. Lab 3에서 구현하고자 하는 Single-Cycle CPU의 설계도 |

설계도에 존재하는 Datapath 요소들 (ALU, MUX, Register, Memory)은 operation에 따라서 다르게 동작해야 한다. 즉, instruction마다 이러한 각각의 요소들을 Control하는 신호들에 어떤 signal을 줄 것인지 결정해야 한다. 각 instruction에 따른 signal들을 truth table로 나타내면 다음과 같다.

|  |
| --- |
|  |
| Figure 2. Truth Table of control signal for components according to Instruction |

**3. Implementation**

Control signal truth table을 바탕으로 instruction을 입력 받고, 각 유닛들에게 전달할 control signal 을 출력으로 생성하는 CTRL 모듈을 구현하였다. Instruction에 immediate이 포함되어 있는 경우 이를 ALU 등의 요소에 입력하기 전에 일정한 format으로 맞춰주어야 한다. 따라서 Instruction을 입력으로 받고 사용할 immediate value를 출력으로 생성하는 IMM 모듈을 구현하였다. ALU도 instruction에 따라 연산을 하도록 하는 모듈로 따로 구현하였고, 모듈들을 RISCV\_TOP에서 모두 연결해주었다. 각 control path들과 data path간의 연결은 위 설계도와 동일하게 진행하였으며, MUX의 경우 RISCV\_TOP에서 직접 구현하였다.

**4. Evaluation**

완성된 코드를 컴파일한 이후 주어진 3개의 testbench 파일을 사용하여 시뮬레이션 결과를 확인하였다. 그 결과 모든 테스트에 대해서 다음과 같이 정상적인 결과값이 나온다는 사실을 확인하였다.

|  |
| --- |
|  |
| Figure 3. 주어진 Testbench 파일로 실행한 시뮬레이션 결과 – inst |

|  |
| --- |
|  |
| Figure 4. 주어진 Testbench 파일로 실행한 시뮬레이션 결과 – forloop |

|  |
| --- |
|  |
| Figure 4. 주어진 Testbench 파일로 실행한 시뮬레이션 결과 - sort |

이를 통해 구현한 Single-Cycle CPU가 정상적으로 잘 작동한다는 사실을 파악할 수 있다.

**5. Discussion**

구현해야 하는 Single-Cycle이 동작해야 하는 Instruction마다 어떤 data path 요소들에게 어떤 control signal을 보내줘야 정상적으로 작동하는지를 파악하고, 사전에 설계하는 과정에 많은 시간이 소요되었다. 이전 랩들과 마찬가지로 구현 단계에서의 디버깅에 어려움이 있었으나, 기존 랩 들로부터의 경험이 도움이 되었다.

**6. Conclusion**

이번 Single-Cycle CPU을 구현하며 앞으로 더 진행할 CPU 랩에서 가장 근간이 되는 동작 요소들이 무엇인지 파악하였고 각각의 요소들이 Verilog에서 어떠한 방식으로 실제 구현될 수 있는지 확인할 수 있었다. 무엇보다 이전 랩보다 복잡하고 큰 시스템을 구현하는 과정에서부터 철저한 사전 설계가 얼마나 중요한지 확인할 수 있었고, 이후 진행될 Multi-Cycle CPU와 Pipeline의 구현을 위해 필요한 사전 지식들을 파악할 수 있었다.