# **Personal Feelings**

이번 과제는 Single-Cycle MIPS Simulator를 작성하기 위한 나만의 micro architecture를 작성하고, C 언어로 구현하며, 구현한 결과물을 평가하는 것이었다. 첫 번째 과제인 Simple Calculator와는 또 다른 느낌의 과제였다. Simple Calculator는 Von Neumann Computer의 실행 과정 4단계인 Fetch, Decode, Execute, Store의 과정을 지키기만 하여 Switch 문을 통해 이를 구현하였는데, 두 번째 과제인 Single-Cycle MIPS Simulator는 MIPS ISA를 읽고 Single-Cycle micro architecture의 구조와 data path를 고려하여 실행되게 하는 것이 과제였다.

많은 시행착오가 있었다. 처음에는 내 생각대로 잘 안 되어 Git Hub에 올라와 있는 Single-Cycle MIPS Simulator 프로그램들을 보고 개념을 잡으려고 했지만 Verilog를 사용하여 하드웨어적으로 구현한 코드와 micro architecture는 무시하고 Switch 문을 사용해 작성한 코드들이 대부분이라 그리 큰 도움이 되지는 않았다. 결국, 처음부터 내 힘으로 시작할 수밖에 없었다. Single-Cycle 구조를 하나하나 손으로 그려보며 각 하드웨어가 왜 필요한지 곱씹으며 여러 모듈화를 통해 해당 과제를 완성하게 되었다.

이번 과제를 끝마치고, 개인적으로 만족스러운 결과를 얻었다고 생각한다. 논리적으로만 생각하고 구현한 구조를 C 언어로 구현했으며 그 결과물이 정상적으로 작동했기 때문이다. 비록 많은 Segment Fault들을 만나 좌절하기도 하였으나, 결국엔 이를 극복하고 과제를 마칠 수 있었다. 한가지 나를 괴롭혔던 항목이 존재하기는 한다. 피보나치 함수의 JAL 명령어를 JALR 명령어로 교체하는 것이 그 범인인데, 보고서에서도 언급했듯이, 원래대로의 JALR 연산을 수행한다고 가정하고 코드를 수정한다고 하면 JALR 명령어 뒤에 NOP 명령어를 삽입해줘야 하므로 해당 명령어를 제외한 다른 모든 주소를 참조하는 명령어들의 주소 값을 바꿔줘야 한다는 문제점에 봉착하였었다. 하지만 그냥 예외를 두어서 이 문제를 해결하였는데, 시간을 조금 더 주어졌다면 아예 이 바이너리 코드를 다시 작성하여 실행시켜 보고 싶다는 아쉬움이 남았다.

현재로서 제일 궁금한 것은 다음 과제인 파이프라인 구현이다. 어떤 식으로 해당 결과물의 단점을 해결할 것인지가 궁금하다. 이에 대해 한번 알아보고 싶다.