Project 2: Simple MIPS emulator Report

201811118 이 구

1. Code Flow

Register의 경우 name(string), value(unsigned int) 값을 가지는 struct로 구성하였고, 전체 register를 하나의 vector 형태로 유지하여 사용하였다. PC의 경우, register와 동일한 struct로 만들어 관리하였다. Memory의 경우 map을 이용하여 구현하였고, memory 주소를 key로, memory에 저장되는 값을 value로 이용하였다.

main 함수가 호출되면 파일의 전체 내용을 읽은 후 vector 형태로 저장하고, 이후 object 파일의 윗 두 줄이 제공하는 정보를 이용하여 메모리의 text 영역과 data 영역을 해당하는 값으로 초기화 하였다.

그 후, while문을 돌며 emulation을 시작한다. while문의 경우, object 파일의 첫 줄에서 알 수 있는 text section의 크기 정보를 이용하여 종결 조건을 결정하였고, -n 옵션이 존재하는 경우 해당 횟수만큼 실행 후 break를 통해 while문을 종료한다.

각 instruction의 format에 따라 decoding 하는 형식이 달라지므로, 세 format 모두 공통적으로 가지고 있는 opcode만을 이용하여 첫 번째 parsing을 진행한다. opcode가 0인 경우 R-format, opcode가 2 또는 3인 경우 J-format, 나머지를 I-format으로 판단한다.

R-format의 경우, 모두 opcode가 0이므로 funct 값을 확인하여 어떤 종류의 연산인지 파악한다. 이후, shift 연산을 통해 rs, rt, rd, shamt 값을 parsing한 후 제시된 연산 과정을 구현하였다.

I-format의 경우, opcode를 통해 구현해야 하는 모든 연산을 구분할 수 있다. 이후, shift 연산을 통해 rs, rt, imm 값을 parsing한다. 이때, imm값의 경우 sign-extension이 일어날 수 있도록 signed int로 explicit casting 해주었다. 이 값들을 통해 제시된 연산 과정을 구현하였다. 이때, sb의 경우 기존에 memory에 할당된 값들 중 일부만을 덮어씌워야 한다. MIPS의 경우, Big-Endian 방식을 이용하므로, 주소값을 4로 나눈 나머지 값을 활용하여 구현하였다. 예를들어 나머지가 1인 경우, 0xFF00FFFF 형태의 마스크와 기존 메모리에 저장되어 있던 값의 and 연산을 수행한 후, ({1byte 정보} << 16) 값을 더해주는 방식으로 구현하였다.

J-format의 경우, I-format과 마찬가지로 opcode를 통해 구현해야 하는 모든 연산을 구분할 수 있다. 이후, shift 연산을 통해 target 값을 parsing한 후 4를 곱해주어 원하는 주소를 계산하였다. 이를 이용하여 제시된 연산 과정을 구현하였다.

Output format을 맞춰주기 위해, print_registers, print_memory 함수를 구현하여 사용하였다. -d 옵션이 존재하는 경우, while문 안에서 instruction이 수행 될 때 마다 print_registers 함수를 호출하였고, -m 옵션이 존재하는 경우 함께 print_memory 함수를 호출하였다. -d 옵션이 존재하지 않는 경우, main 함수의 종료 직전 print_registers 함수를 호출하였으며, 마찬가지로 -m 옵션이 존재하는 경우 print_memory 함수를 함께 호출하였다.

2. Compile & Execution

제출한 압축 파일은 MIPS_emulator.cpp, Makefile, Report로 구성되어있다. 압축을 해제한 폴더에서 아래의 명령어를 통해 컴파일 및 실행할 수 있다.

```
$ (sudo) make
$ (sudo) ./runfile  [-m addr1:addr2] [-d] [-n num_instruction] <object file>
```

3. Environment

코드는 Ubuntu 22.04.3 LTS 환경에서 c++로 작성되었으며, g++ 9.5.0을 사용하여 컴파일하였다. 테스트는 Windows Subsystem for Linux (Ubuntu 20.04) 환경에서 g++ 9.4.0을 사용하여 진행하였다.