컴퓨터 구조 과제 2 보고서

202111133 이석민
Email: jhss4475@dgist.ac.kr

1. 간단한 flow 소개

1) 코드가 복잡하지 않기 때문에 거의 모든 코드를 main함수 안에 구성하였습니다.

2) main 함수의 초기 부분에서는 -m, -d, -n과 같은 옵션의 유무와 값, inputfile의 이름을 저장하고 있습니다.

3) 그 다음 while문에서는 input file의 content를 한줄 씩 읽어오면서 메모리(전역변수 data_memory)에 data와 instruction들을 할당하고 있습니다.

4) 그 다음 for문에서는 insturction들을 하나씩 읽으면서 각 insturction들의 opcode와 functional code에 따라 코드를 실행하여 register와 memory, PC의 값들을 올바르게 바꾸는 작업을 하고 있습니다.

5) while문의 마지막 부분을 보면 -d 와 -m옵션에 따라서 각 instruction이 종료될 때, register와 memory값을 출력하는 코드를 볼 수 있습니다.

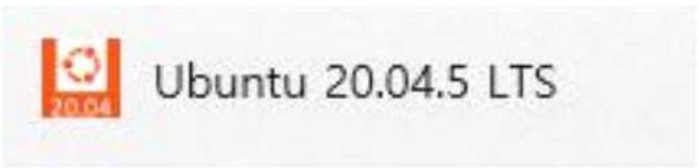
6) while문이 끝난 후 바로 등장하는 코드를 보면, -d 옵션에 따라서 instruction이 모두 종료된 후, register와 memory의 값을 출력하는 코드를 볼 수 있습니다.

2. 컴파일/실행 방법, 환경

1. 제가 작성한 cpp emulator는 g++ 9.4.0으로 compile하였습니다.

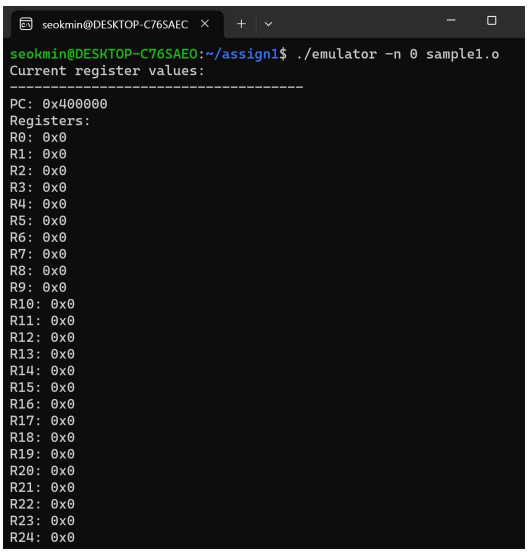
```
seokmin@DESKTOP-C76SAEO:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
```

2. 컴파일을 실행한 Ubuntu의 version은 20.04.5입니다.



3. 컴파일은 밑의 코드를 통하여 실행했습니다. 여기서 emulator.cpp는 제가 작성한 cpp emulator file입니다. 밑의 그림을 보면 성공적으로 emulator.cpp가 compile된 것을 확인할 수 있습니다.

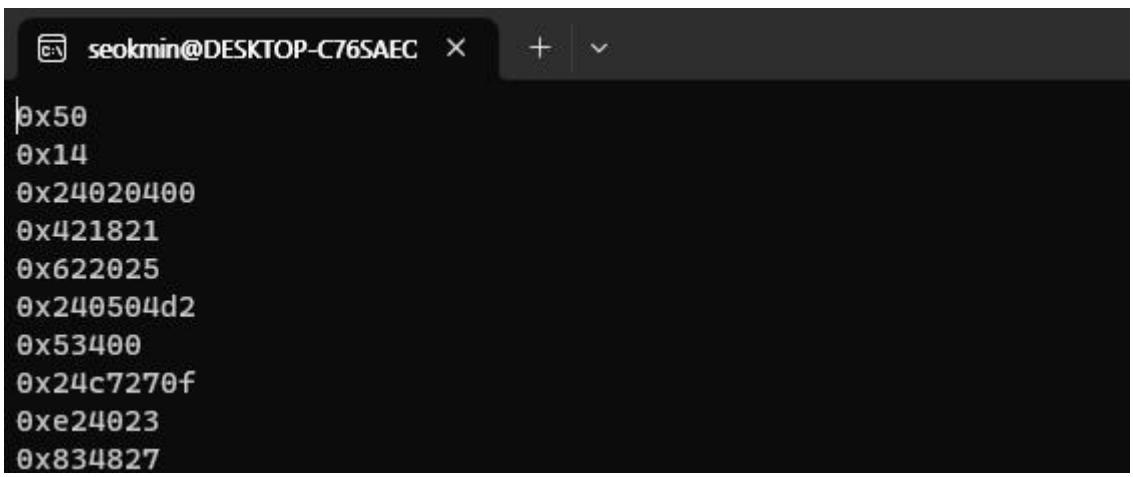
4. sample1.o와 sample2.o는 각각 sample1.s와 sample2.s를 과제 1의 assembler를 이용하여 16진수로 변환한 파일입니다. emulator는 emulator.cpp를 compile한 file입니다. 밑에 제시된 여러 명령어를 통해 emulator를 실행해보며, emulator가 정상적으로 동작하는 것을 확인 할 수 있었습니다.

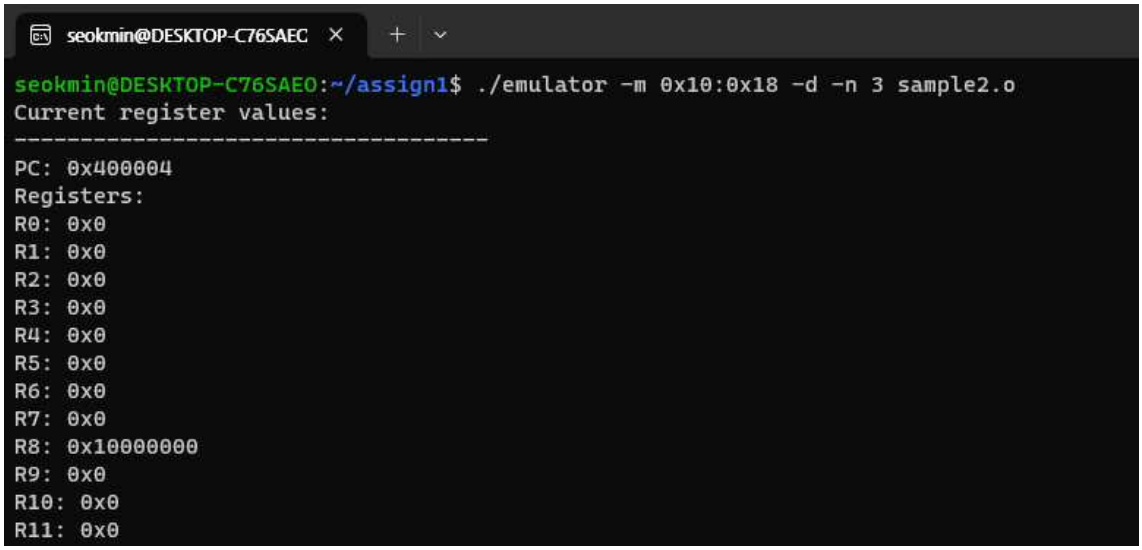


- 명령어를 하나도 실행하지 않고, -m 옵션이 없는 경우에, 초기 레지스터 값만 출력되는 것을 알 수 있습니다.

```
seokmin@DESKTOP-C76SAEO:~/assign1$ ./emulator -m 0x400000:0x400010 -n 0 sample1.o
Current register values:
-----------------------------------
PC: 0x400000
Registers:
R0: 0x0
R1: 0x0
R2: 0x0
R3: 0x0
R4: 0x0
R5: 0x0
R6: 0x0
R7: 0x0
R8: 0x0
R9: 0x0
R10: 0x0
R11: 0x0
R12: 0x0
R13: 0x0
R14: 0x0
R15: 0x0
R16: 0x0
R17: 0x0
R18: 0x0
R19: 0x0
R20: 0x0
R21: 0x0
R22: 0x0
R23: 0x0
R24: 0x0
R25: 0x0
R26: 0x0
R27: 0x0
R28: 0x0
R29: 0x0
R30: 0x0
R31: 0x0
Memory content [400000..400010]:
-----------------------------------
400000: 0x24020400
400004: 0x421821
400008: 0x622025
40000c: 0x240504d2
400010: 0x53400
seokmin@DESKTOP-C76SAEO:~/assign1$
```

- 명령어를 하나도 실행하지 않고, 특정 메모리 부분을 출력하라는 옵션을 넣어줬을 때, 초기 레지스터와 메모리의 값이 출력되는 것을 확인할 수 있습니다. 메모리의 해당 부분은 text section 즉 instruction이 저장되어 있는 부분인데, 주소에 맞게 insturction이 16진수 형태로 잘 출력되는 것을 아래그림을 참조하여 알 수 있습니다. 아래그림은 sample1.o file입니다.(smaple1.o 의 3번째 줄과 memory출력 부분의 첫 번째 부분이 일치함을 알 수 있습니다.)





- 이외에 여러 옵션들과 상황에서도 알맞게 잘 동작하는 것을 관찰 할 수 있었습니다.

감사합니다.