컴퓨터 구조 과제 1 보고서

202111133 이석민
Email: jhss4475@dgist.ac.kr

1. 간단한 FLOW 소개

1) 먼저 input file을 string으로 입력받습니다.

2) string으로 입력받은 MIPS CODE를 generate_memory_map함수에 argument로 넘기며 함수를 실행합니다.

3) generate_memory_map 함수에서는 MIPS CODE를 바탕으로 각 DATA/Instruction/Label마다 메모리의 주소를 할당하고,이를 memory_map 이라는 전역변수에 저장합니다.

4) generate_data_memory 함수는 3)에서 만든 memory_map을 바탕으로 memory의 data section에 저장된 data와 data의 address를 map 형태로 (address : data) 전역변수 data_memory_map에 저장합니다.

5) generate_text_memory 함수는 3)에서 만든 memory_map을 바탕으로 memory의 text section에 저장된 instruction과 instruction의 address를 map 형태로 (address : instruction) 전역변수 text_memory_map에 저장합니다.

6) generate_instruc_to_binary 함수에서는 text_memory_map을 바탕으로 instruction을 32bit binary로 하나씩 mapping합니다. 그 후, 한 명령어 당 32 bit binary로 convert된 전체 명령어 string을 whole_answer에 저장하고 main 함수로 return합니다.

7) text,data section size, data section에 저장된 data를 추가한 real_whole_ans를 바탕으로 <inputfilename>.o file을 생성합니다.

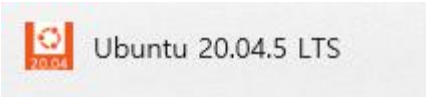
8) 2진수 answer를 bin_to_hex()함수에서 hexadecimal answer로 바꾸고 <inputfilename>.txt file을 생성합니다.

2. 컴파일/실행 방법, 환경

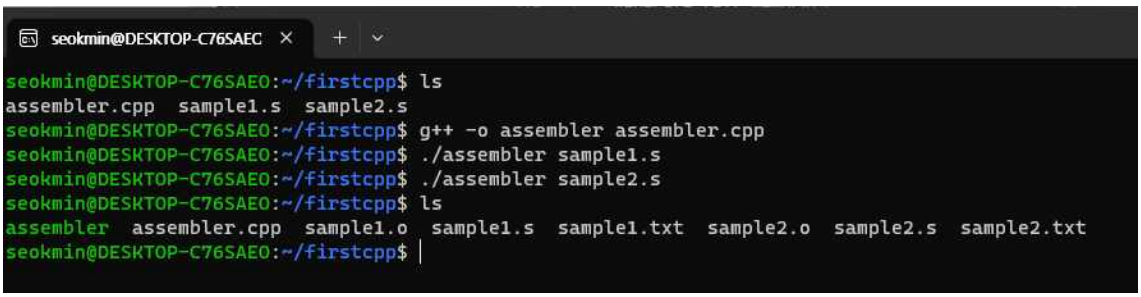
1. 제가 작성한 cpp assembler는 g++ 9.4.0으로 compile하였습니다.

```
seokmin@DESKTOP-C76SAEO:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
```

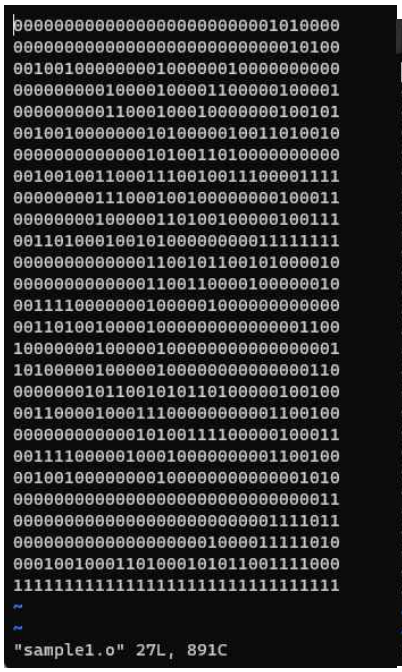
2. 컴파일을 실행한 Ubuntu의 version은 20.04.5입니다.



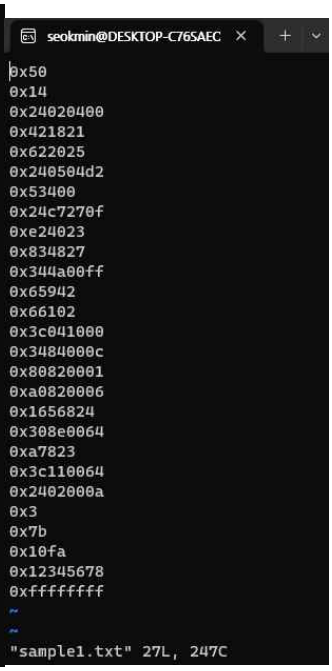
3. 컴파일은 밑의 코드를 통하여 실행했습니다. 여기서 assembler.cpp는 제가 작성한 cpp assembler file이고, sample1.s와 sample2.s는 lms에 업로드 되어있는 sample file입니다. 밑의 그림을 보면 성공적으로 compile된 것을 확인할 수 있습니다. assembler는 assembler를 compile한 file이고, sample1.s는 sample1.s를 assembler를 통해 binary file로 바꾼 것입니다. sample1.txt는 sample1.s를 assembler를 통해 hexadecimal file로 바꾼 것입니다. sample2의 경우도 마찬가지입니다.



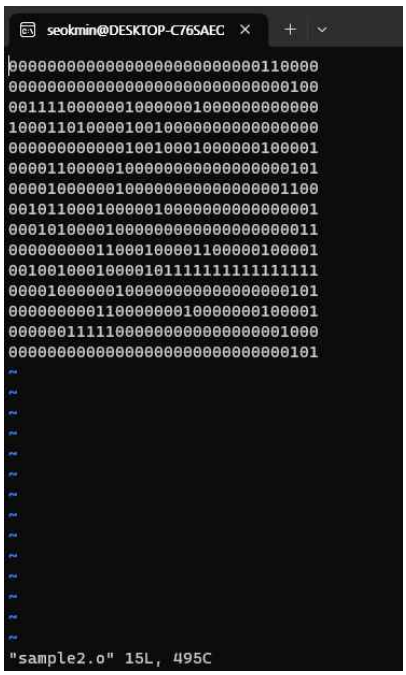
4. sample1.o, sample1.txt, sample2.o, sample2.txt를 보면, 성공적으로 MIPS Assembly file이 binary,hexadecimal file로 convert 되었음을 확인할 수 있습니다.



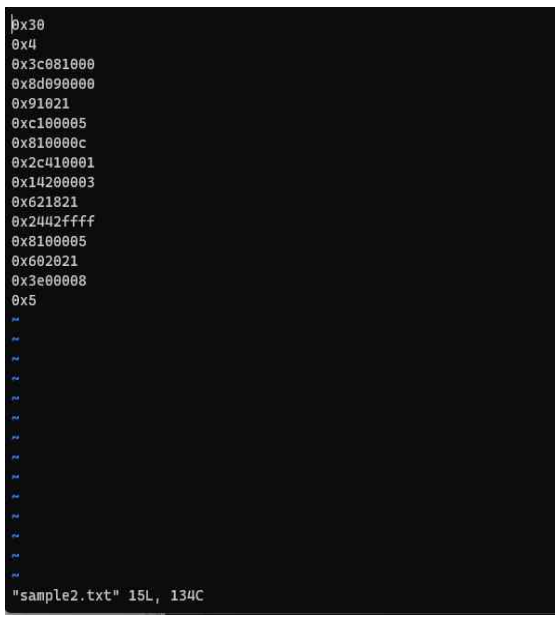
<그림 4 : sample1.o>



<그림 5 : sample.txt>



<그림 6 : sample2.o>



<그림 7 : sample2.txt>

감사합니다.