시스템프로그래밍(나) Project 2

- SIC/XE Visual Simulator -

컴퓨터학부 정해성

목차

1. 동기 및 목적

본 보고서는 오브젝트 프로그램을 입력으로 받아 가상으로 설정한 메모리 공간에 로드하고, 실제로 명령어를 실행하는 시뮬레이터를 구현하는 과제에 대한 보고서입니다. 수업에서 다룬 COPY 프로그램의 오브젝트 프로그램을 실행하는 것을 목적으로 하여 구현되어 있습니다.

2. 설계/구현 아이디어

- **GUI**

  GUI 구현은 Java의 swing 프레임워크를 사용했습니다. Intellij의 swing gui 에디터를 활용하여 ui를 구성하였으므로 실행시 Intellij IDE에서 실행해주시면 감사하겠습니다.

- **프로그램 종료**

  COPY 프로그램에서는 프로그램의 종료가 명확하지 않으므로 `J @RETADR`를 만나면 무조건 종료하는 것으로 구현하였습니다. 프로그램 종료는 `ProgramEndException`을 던지는 것으로 확인합니다.

- **프로그램 로드**

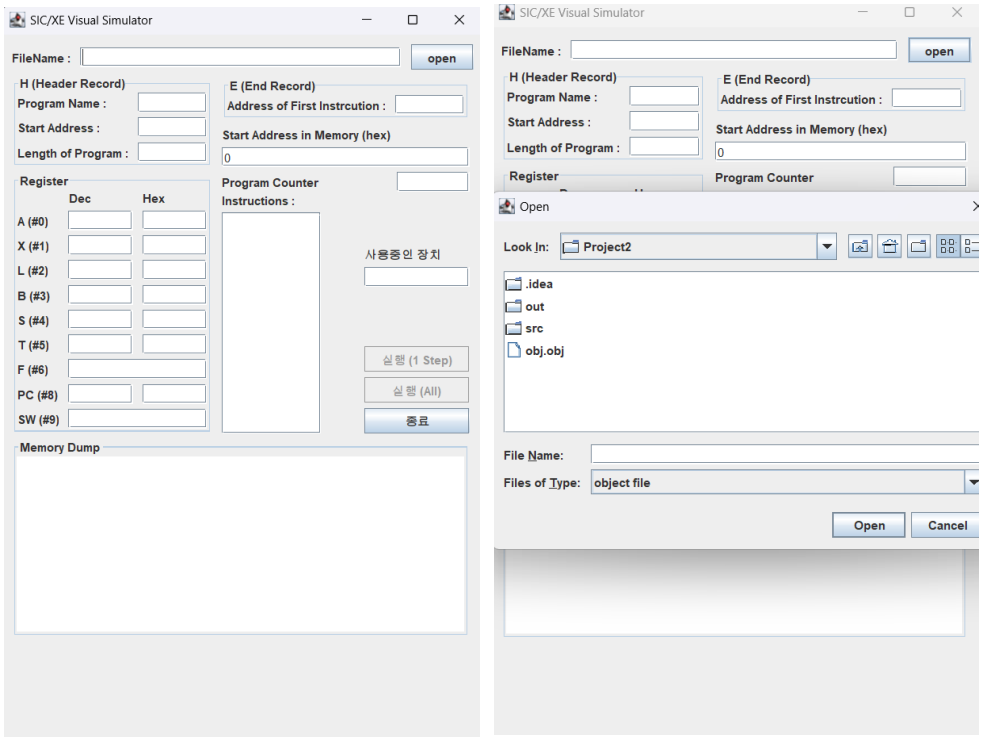
  `Loader.loadFromFile`()이라는 함수를 실행함으로써 프로그램을 가상의 메모리(Memory 클래스)에 로드합니다. 링킹을 진행하기 위해 두 번의 pass를 거치는데, pass 1에서는 `ExternalSymbolTable`의 내용을 채우고, pass 2에서는 텍스트 레코드의 내용을 로드하면서 `modification` 레코드에서 수정해야 할 부분을 수정해줍니다.

- **프로그램 수행**

  PC 레지스터를 참조하여 해당 위치에서 먼저 1바이트만 읽어서 opcode를 확인합니다. 이후 읽은 명령어의 형식을 확인하여 `InstructionExecutor`의 `executeFormat2Inst`() 또는 `executeFormat3Inst`() 메서드를 실행합니다.

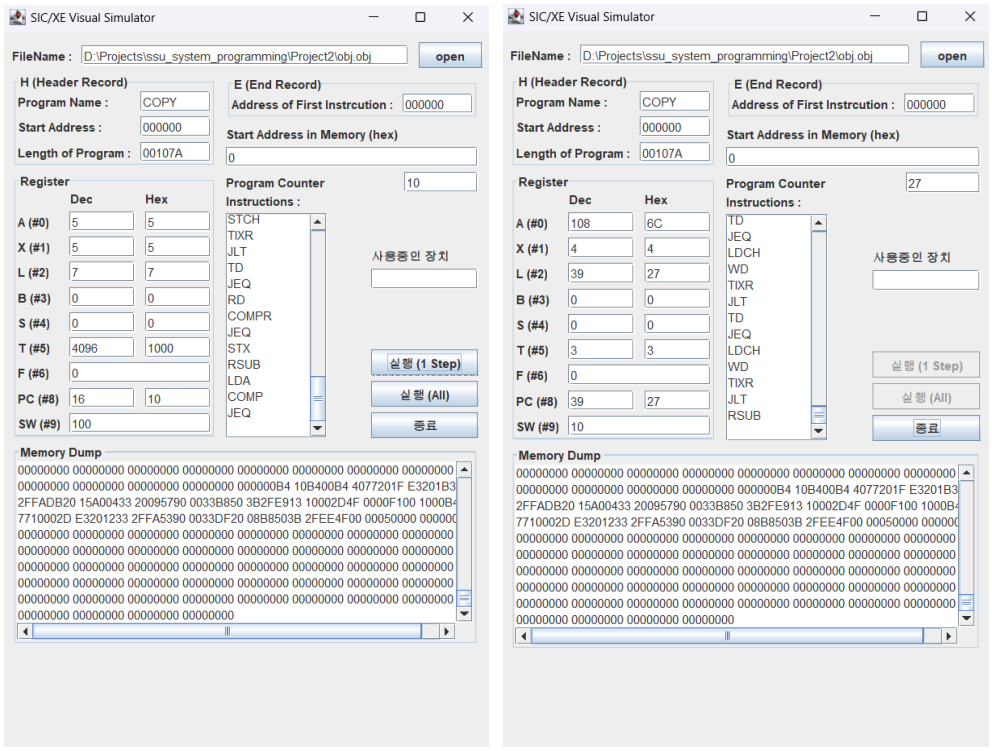
  executeFormat3Inst()에서는 점프 명령어를 수행하기 위해 목적지가 되는 PC값을 반환합니다. 점프할 필요가 없다면 현재 PC 값을 그대로 반환합니다.

3. 수행 결과



시뮬레이터 프로그램 초기 화면 / 파일 선택 화면

Start Address in Memory 텍스트 필드의 값으로 프로그램이 로드되는 주소를 설정합니다. (파일을 선택하자마자 로드되므로 파일을 선택하기 전에 필드 값을 변경해야 합니다.)



실행 중 화면 / 실행 완료 화면

input.txt

```
hello
```

output.txt

```
hello<null>EOFl
```

이러한 결과가 나오는 이유는 본래 COPY 프로그램에 버그가 있기 때문입니다. 해당 사항은 아래 보충할 점에서 설명하겠습니다.

input.txt와 output.txt는 원래 존재하는 파일이며, 이름이 고정되어 있습니다. 실행 전의 output.txt의 파일에는 내용이 존재하지 않습니다.

4. 결론 및 보충할 점

- 결론

  이번 과제를 통해 가상의 메모리 배열을 생성하여 프로그램을 로드하고, 실행할 수 있는 시뮬레이션을 구현하였습니다. 또한 실제로 실행을 시켜보면서 COPY 프로그램이 의도하지 않게 동작한다는 것도 확인할 수 있었습니다.

  클래스의 수가 많아지니까 이 클래스가 내가 의도한 대로 동작하는지 각각 테스트해봐야 할 필요를 느꼈고, 테스트가 필요한 클래스마다 main 함수를 따로 두어서 해당 클래스가 독립적으로도 정상적으로 동작하는지 확인하여 규모가 있는 프로그램이더라도 보다 안정적으로 개발을 진행되는 것을 경험하였습니다.

- 보충할 점
  - COPY 프로그램의 버그

  의도한 동작인지 확실하지 않지만, WRREC 서브 프로그램에서 output device에 문자를 write하는 WD 명령어가 인덱스를 비교하는 TIXR, JLT 명령어보다 앞에 있습니다. 따라서 예를 들어 LENGTH가 5라면 5번째 문자까지 출력하고, 그 다음에 LENGTH와 비교하게 됩니다.

  WD, TIXR, JLT 순서로 되어있는 것을 TIXR, JLT, WD로 바꾼다면 문제가 해결될

것이라고 예상됩니다.