# 시스템 프로그래밍 Project #3

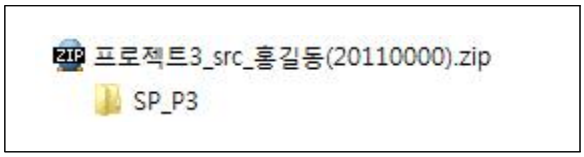
- **과제내용**
  - 프로젝트#2의 결과물을 메모리에 올려 실행함
  - ControlSection 방식으로 생성된 Object Program을 메모리에 적재하고, 적재된 Object Program을 실행시키는 시뮬레이터의 구현
  - 시뮬레이션 과정이 Step-by-Step으로 Visual하게 보여주는 Java GUI 프로그램
  - 시뮬레이터 구현 모듈: GUI 모듈, 연산 모듈, 가상 장치(메모리, 레지스터) 모듈, 로더

- **과제 목적**
  - 가상머신의 시뮬레이터를 통해 ControlSection 기반의 오브젝트 프로그램을 메모리에 적재 및 실행시켜 봄으로써, 실질적인 프로그램의 적재 및 실행과정을 이해함

- **과제 제출 마감**
  - **소스파일**: 6/8/수/23:59, Soft Copy
  - **보고서**: 6/10/금/수업시간 시작 전, Hard Copy, Hard Copy
  - 당일 이후부터는 매일 10 point씩 추가 패널티를 부가함

- **제출형태**
  - HARD COPY : 보고서를 출력하여 수업시간 시작 전 제출(06/10/금/수업시간)
  - SOFT COPY : 소스파일 E-campus (06/08/23:59)
    : 보고서 E-campus (06/10/수업시간 시작전)
  - 보고서 파일은 한글/워드문서로 작성
  - 제목을 제외한 본문의 내용은 font 10으로 함

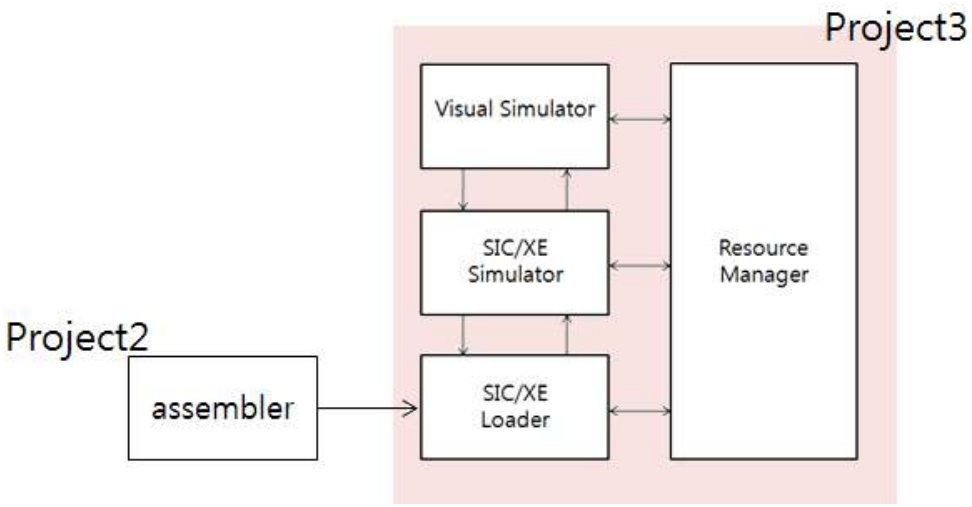
- **보고서 작성 방법 ( 50 point )**
  - 요구사항
  : 표지 반드시 넣을 것(5p) (학번, 이름, 과제명, 수업 구분<가,나>)
  : 목차 및 내용
  1. 동기/목적(5p)  2. 설계/구현 아이디어(10p)  3. 주요 소스코드 설명(10p)
  4. 수행결과(10p)  5. 결론 및 보충할 점(10p)
  - 목차 중 소스코드는 설명해야 할 주요 소스코드(주석포함)를 포함하고 설명함
  - 점수 평가에서 보고서의 비중이 높으므로 제출 마감 전까지 성심껏 작성할 것
- **소스코드 제출 방법 ( 50 point )**
  - 파일 이름: **프로젝트3_report_이름(학번).hwp(or doc), 프로젝트3_src_이름(학번).zip**
  - 제출 파일을 보고서와 소스파일을 각각의 제출날짜에 맞추어 제출함
  - 소스코드는 아래 예시와 같이 제출함(이클립스 또는 비쥬얼스튜디오의 SP_P3 폴더에 그대로 첨부)




**※ 주의사항!!!**

이클립스 또는 비쥬얼스튜디오 프로젝트 파일을 그대로 첨부 안 할시 소스코드 0점 처리함

▪ **시뮬레이터 프로그램 구현 명세**




- **SIC/XE Loader**
  - 정의 : 어셈블러의 결과물(Object Program)을 읽어 들여 파싱하고, 메모리에 적재(loading)함
  - 기능 : Resource Manager에 변수로 지정되어 있는 가상의 메모리 영역에 Object Program을 적재함

- **Resource Manager**
  - 정의 : 시뮬레이터를 구동시키기 위한 가상의 하드웨어
  - 기능 : Object Program을 적재하기 위한 메모리영역
    : 프로세서가 연산에 사용하는 레지스터영역 등을 각각의 변수로 배정하여 사용할 것

- **SIC/XE Simulator**
  - 정의 : SIC/XE 머신이 Ojbect Program을 수행시키는 시뮬레이터 모듈
  - 기능 : Resource Manager에 적재된 Object Program을 실행

- **Visual Simulator**
  - 정의 : 시뮬레이터의 동작을 GUI 기반으로 보여주는 모듈
  - 기능 : SIC/XE Simulator를 동작시킨 이후 Resource Manager 내의 값들을 읽어 들여 보여줌

▪ **지침**

**: 위 모듈은 필수적인 구성**

**: 필수 모듈 외에 추가 모듈을 설계하여 구현할 경우, 추가 내용은 보고서에 반드시 기술할 것**

**: 주 기능 외에 프로그래밍의 편의를 위해 클래스나 메소드를 추가로 만들 수 있음**

**: 비주얼 시뮬레이터의 경우 GUI 구현할 대 Swing 등을 사용하여야 하므로, JFrame 등을 extend하여야 한다.)**

**: 각 모듈은 page 3~4에 제시한 인터페이스에 따라 구현함**

### 1. VisualSimulator.java

```
//시뮬레이터의 동작을 보여주는 GUI
public interface VisualSimulator {

    //각각 필요한 오브젝트를 생성해주어 실행가능 상태로 만듬
    public void initialize();

    //하나의 명령어만 실행하는 메소드로써 SIC 시뮬레이터에게 작업을 전달
    public void oneStep();

    //남은 명령어를 모두 실행하는 메소드로써 SIC 시뮬레이터에 작업을 전달
    public void allStep();

    //이외에 GUI 관련 메소드들(set, view 등은 자유구현)
}
```

### 2. SicSimulator.java

```
// 로드된 코드를 실질적으로 실행하여 동작
public interface SicSimulator (

    //메모리 작업 등 실질적인 초기화 작업을 수행
    //각 명령어가 저장되어있는 inst.data파일을 읽고 저장
    public void initialize(File instFile);

    //하나의 명령어만 수행한다. 해당 명령어가 수행되고 난 값의 변화를
    //보여주고, 다음 명령어를 포인팅
    public void oneStep();

    //남은 명령어를 모두 수행하는 메소드
    public void allStep();

    //실행한 결과를 로그에 추가하는 메소드
    public void addLog();
}
```

### 3. SicLoader.java

```
//목적코드를 읽어와 메모리에 로드
public interface SicLoader {

    //파일로부터 목적코드를 읽어와 메모리에 로드
    public void load(File objFile);

    //목적코드의 한 줄을 읽고, 각 헤더(H, T, M 등)에 맞는 기능을 수행하여
    //각 메모리 및 명령어 리스트를 초기화 한다.
    public void readLine(String line);
}
```

### 4. ResourceManager.java

```
public interface ResourceManager {
    //메모리 영역을 초기화 하는 메소드
    public void initializeMemory();
    //각 레지스터 값을 초기화 하는 메소드
    public void initializeRegister();

    //디바이스 접근에 대한 메소드
    //디바이스는 각 이름과 매칭되는 파일로 가정한다
    //(F1 이라는 디바이스를 읽으면 F1 이라는 파일에서 값을 읽는다.)
    //해당 디바이스(파일)를 사용 가능한 상태로 만드는 메소드
    public void initialDevice(String devName);
    //선택한 디바이스(파일)에 값을 쓰는 메소드. 파라미터는 변경 가능하다.
    public void writeDevice(String devName, byte[] data, int size);
    //선택한 디바이스(파일)에서 값을 읽는 메소드. 파라미터는 변경 가능하다.
    public byte[] readDevice(String devName, int size);

    //메모리 영역에 값을 쓰는 메소드
    public void setMemory(int locate, byte[] data, int size);
    //레지스터에 값을 세팅하는 메소드. regNum은 레지스터 종류를 나타낸다.
    public void setRegister(int regNum, int value);
    //메모리 영역에서 값을 읽어오는 메소드
    public byte[] getMemory(int locate, int size);
    //레지스터에서 값을 가져오는 메소드
    public int getRegister(int regNum);
    //바뀐 값들을 GUI에 적용시키는 메소드
    public void affectVisualSimulator();
}
```

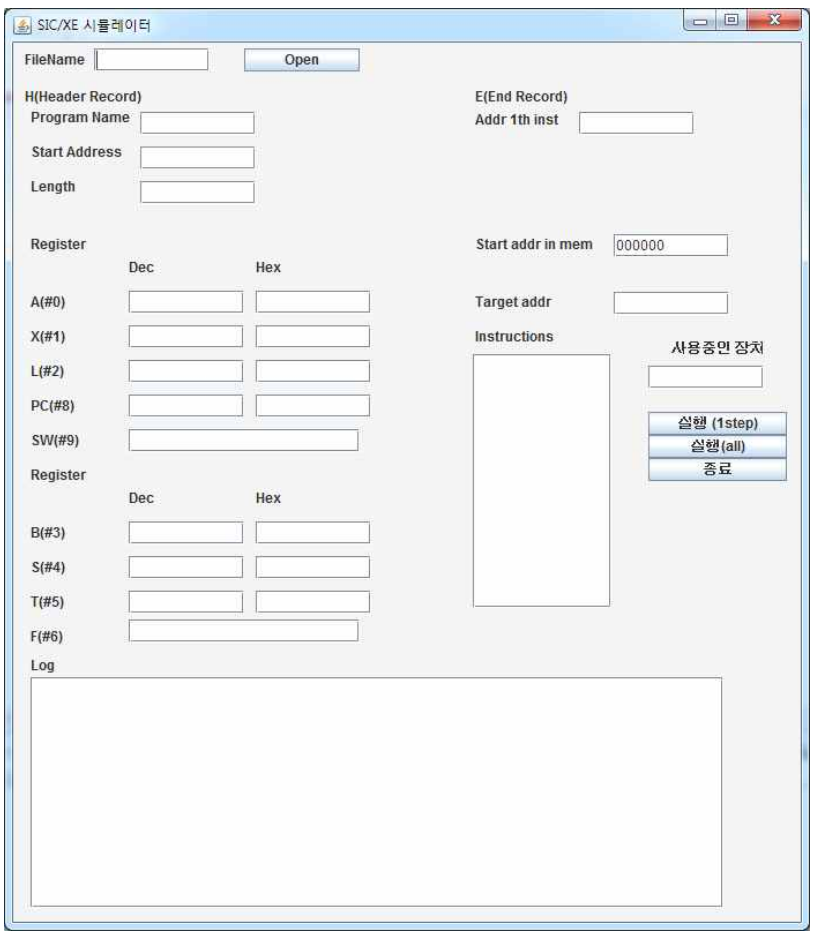
- **구현하여야 하는 GUI에서의 기능**

- 프로그램 종료 버튼
- 파일 오픈기능(파일 오픈 다이얼로그 창 이용)
- 레지스터 영역(SIC 및 SIC/XE머신의 레지스터를 모두 포함)
- 메모리 영역(다음과 같은 두 가지 표현 방식 중 한 가지 방식으로 보여주어야 함)
  1. 가상으로 설정한 메모리를 직접 보여주고, 현재 수행되고 있는 Instruction의 주소를 포인팅 해준다(가능하면 해당 Instruction을 영역지정까지)
  2. 메모리에 올라간 코드를 파싱하여 명령어 목록을 만든다. 그리고 해당 리스트를 표시해준다. 현재 수행되고 있는 명령어를 리스트에서 선택하여 표시해준다.
- 프로그램 정보: 프로그램 길이, 현재 포인팅 주소, 사용 중인 장치, 현재 수행중인 명령어 정보 등
- 1 step 및 all step 기능
- 실행된 결과(변경된 사항)에 대한 로그 출력
- 꼭, 예시와 똑같이 GUI를 만들 필요는 없으며, 각 기능이 잘 수행되기만 하면 됨
- 첨부된 spProejct2.jar을 Java Build Path의 Libraries에 추가하고, 각 interface를 implements하여 위의 코드를 구현할 수 있음

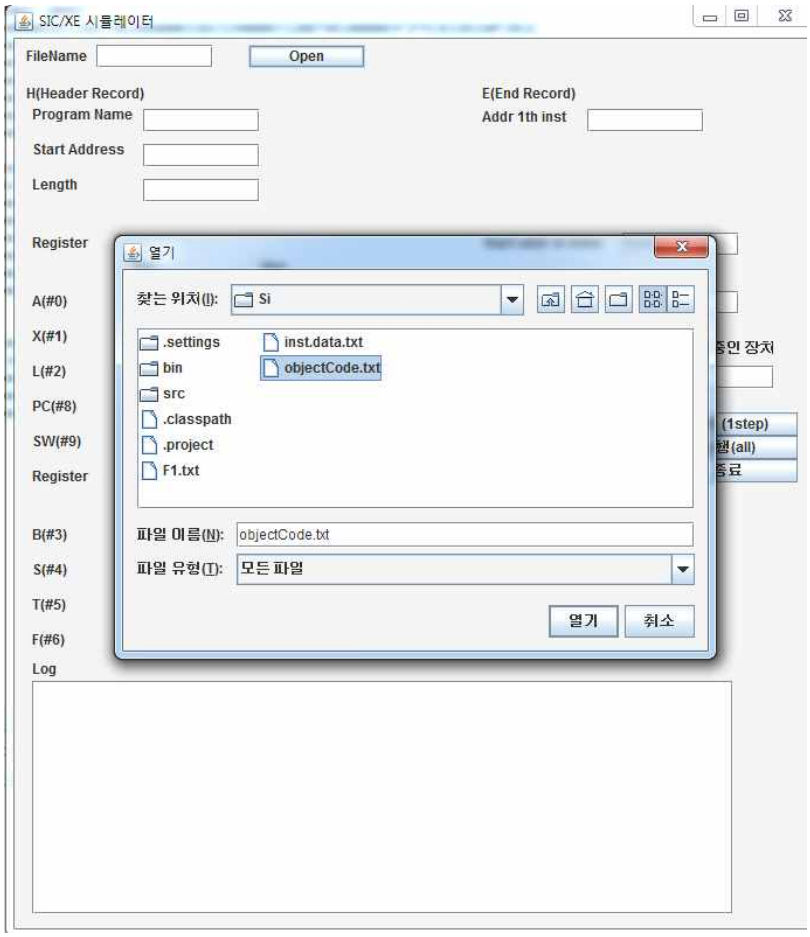
- **입력 파일의 내용**

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T0000001D1720274B10000003202329000003320074B1000003F2FEC0320160F2016
T00001D0D0100030F200A4B1000003E2000
T00003003454F46
M00000405+RDREC
M00001105+WRREC
M00002405+WRREC
E000000

HRDREC 00000000002B
RBUFFERLENGTHBUFEND
T0000001DB410B400B44077201FE3201B332FFADB2015A00433200957900000B850
T00001D0E3B2FE9131000004F0000F1000000
M00001805+BUFFER
M00002105+LENGTH
M00002806+BUFEND
M00002806-BUFFER
E

HWRREC 00000000001C
RLENGTHBUFFER
T0000001CB41077100000E32012332FFA53900000DF2008B8503B2FEE4F000005
M00000305+LENGTH
M00000D05+BUFFER
E
```

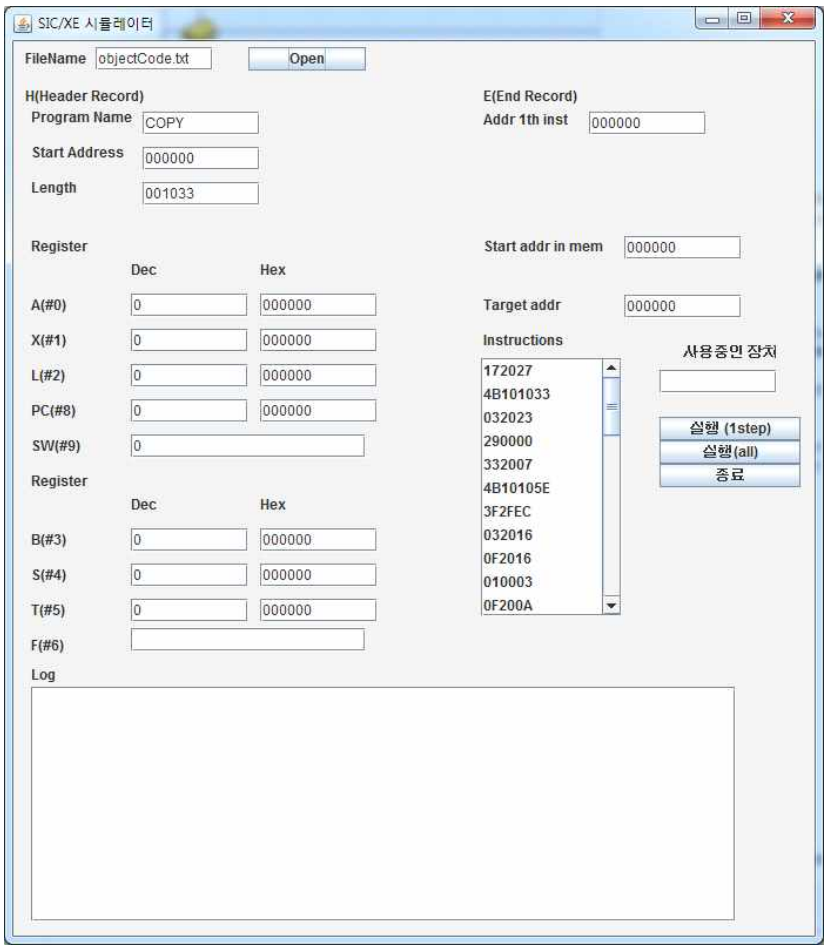
- **초기화면(예시)**



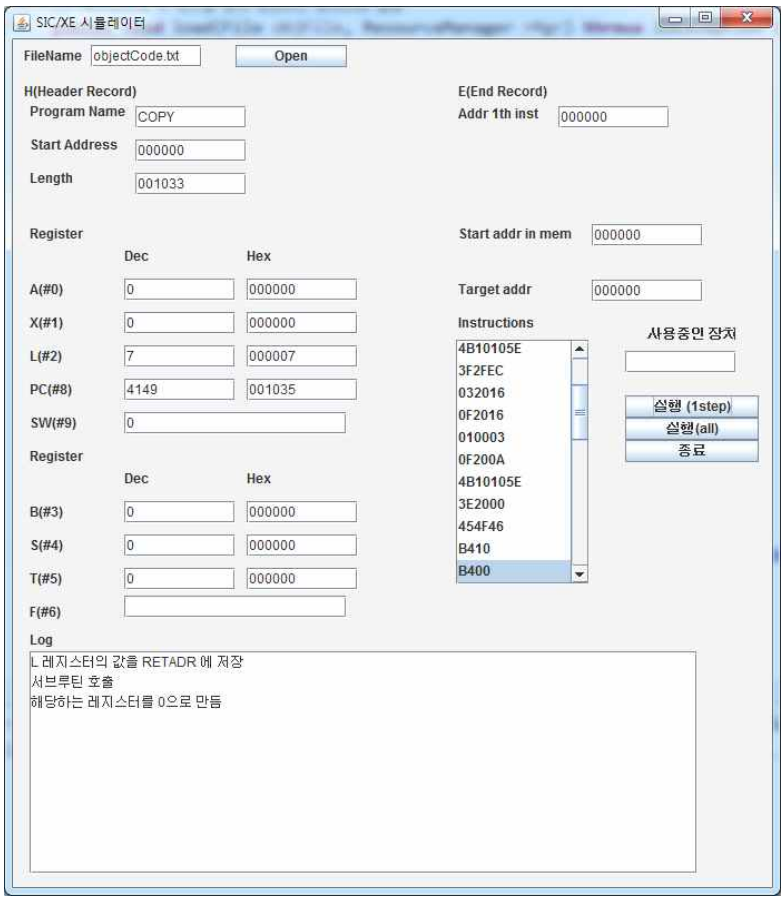
- **파일 오픈(파일 다이얼로그 창)**



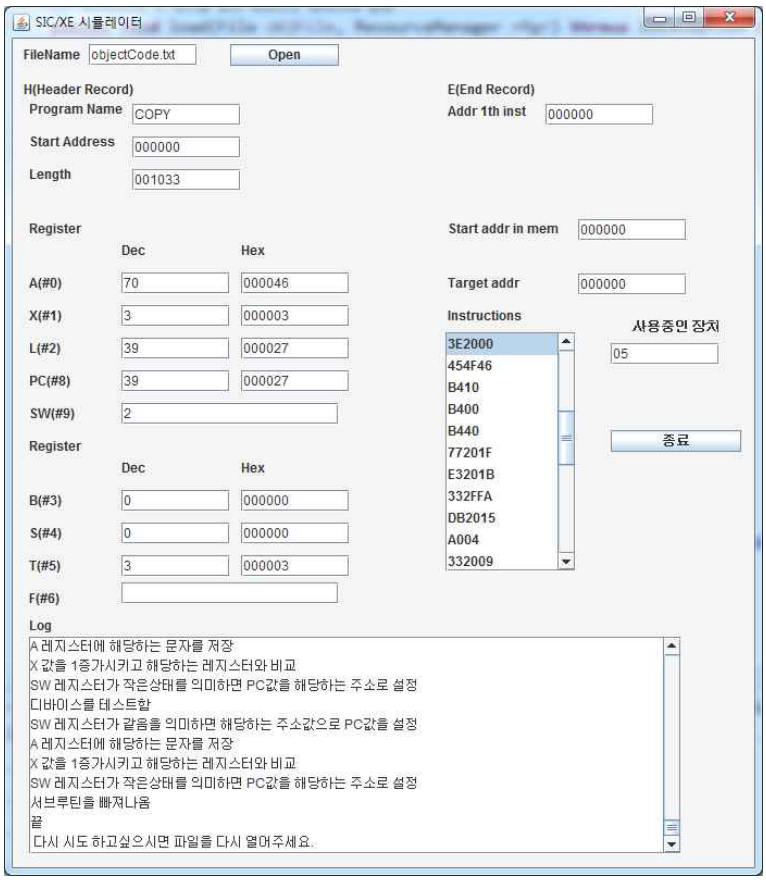

- **파일 오픈 후, 초기 세팅화면(Loader를 통해 오브젝트코드를 Resource Manager에 올림)**



- **1step 수행 시, 해당 오브젝트코드를 실행하고 변화된 값들을 GUI에 적용하고 Log를 남김**
  (ex : B410을 수행 시, X레지스터의 값을 0으로 바꿔주며, PC값이 2만큼 증가)




- **All 수행 시, 남은 오브젝트코드를 모두 실행하고 변화된 값들을 GUI에 적용하고 Log를 남김**



- **수행을 완료하면 F1 파일의 텍스트가 05 파일에 쓰여짐**