

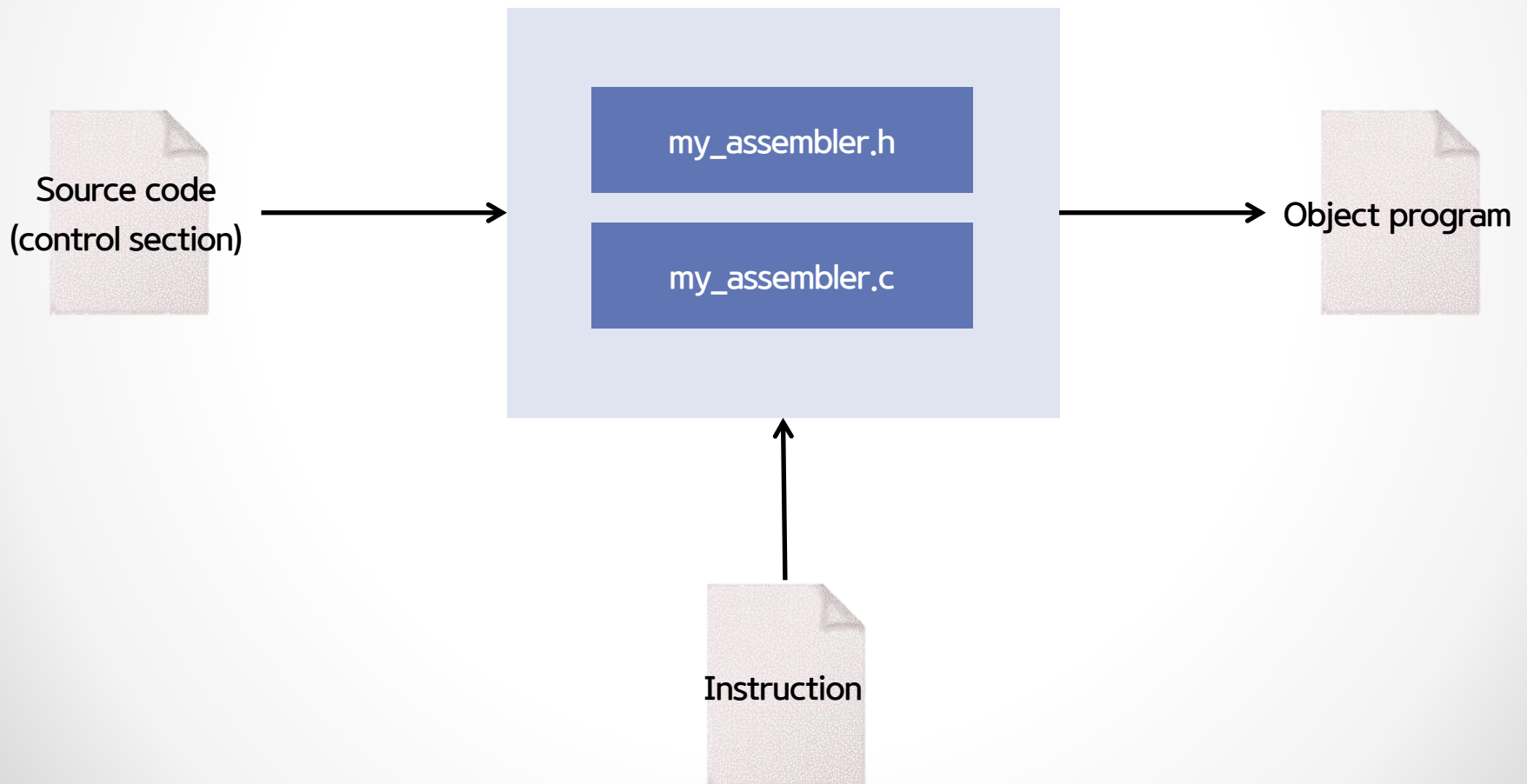
시스템 프로그래밍

프로젝트 3 시뮬레이터 구현하기

숭실대학교
시스템소프트웨어 연구실
조교 권경재

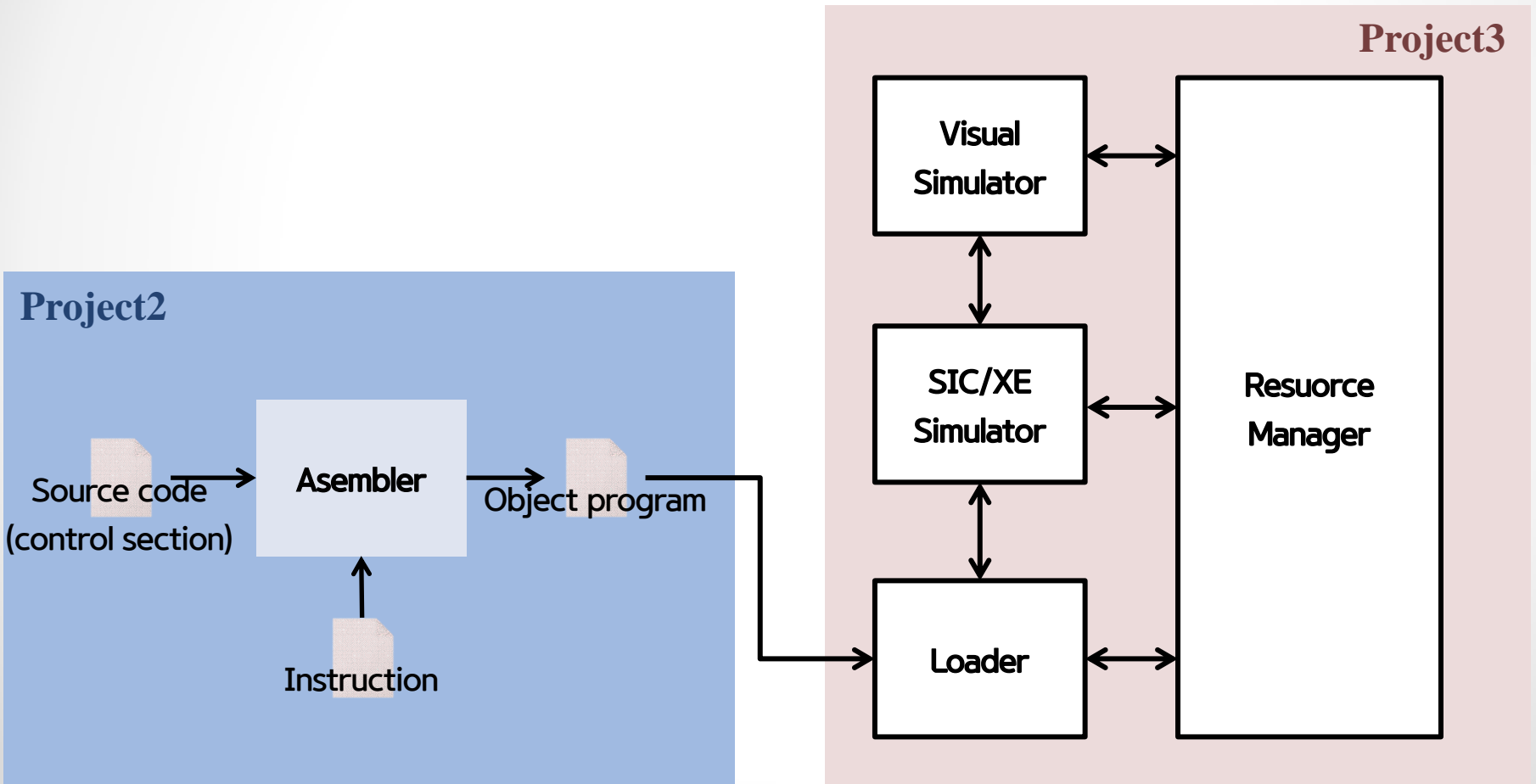
프로젝트 2 개요

- ControlSection 방식의 SIC/XE 소스를 Object Program으로 바꾸는 어셈블러 만들기



프로젝트 3 개요

- ControlSection 방식의 Object Program(프로젝트 2의 결과물)을 가상의 메모리에 올려 실행시키는 시뮬레이터



프로젝트 3 개요

- ControlSection 방식의 Object Program(프로젝트 2의 결과물)을 메모리에 올려 실행시키는 시뮬레이터

SIC/XE 시뮬레이터

FileName

H(Header Record)
Program Name E(End Record)
Addr 1th inst
Start Address
Length

Register

	Dec	Hex
A(#0)	<input type="text"/>	<input type="text"/>
X(#1)	<input type="text"/>	<input type="text"/>
L(#2)	<input type="text"/>	<input type="text"/>
PC(#8)	<input type="text"/>	<input type="text"/>
SW(#9)	<input type="text"/>	<input type="text"/>

Start addr in mem

Target addr

Instructions

사용중인 장치

Register

	Dec	Hex
B(#3)	<input type="text"/>	<input type="text"/>
S(#4)	<input type="text"/>	<input type="text"/>
T(#5)	<input type="text"/>	<input type="text"/>
F(#6)	<input type="text"/>	<input type="text"/>

Log

SIC/XE 시뮬레이터

FileName

H(Header Record)
Program Name E(End Record)
Addr 1th inst
Start Address
Length

Register

	Dec	Hex
A(#0)	<input type="text" value="0"/>	<input type="text" value="000000"/>
X(#1)	<input type="text" value="0"/>	<input type="text" value="000000"/>
L(#2)	<input type="text" value="0"/>	<input type="text" value="000000"/>
PC(#8)	<input type="text" value="0"/>	<input type="text" value="000000"/>
SW(#9)	<input type="text" value="0"/>	<input type="text"/>

Start addr in mem

Target addr

Instructions

사용중인 장치

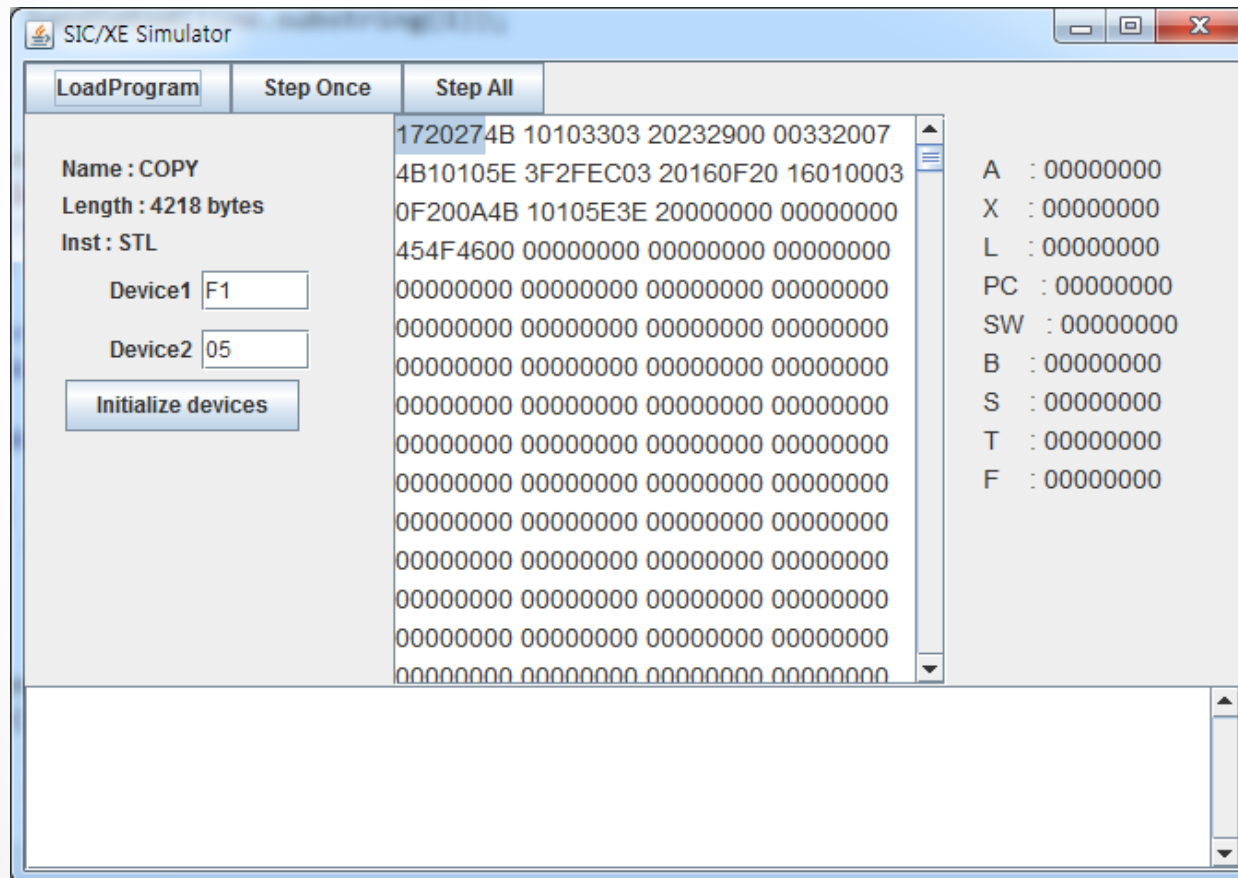
Register

	Dec	Hex
B(#3)	<input type="text" value="0"/>	<input type="text" value="000000"/>
S(#4)	<input type="text" value="0"/>	<input type="text" value="000000"/>
T(#5)	<input type="text" value="0"/>	<input type="text" value="000000"/>
F(#6)	<input type="text"/>	<input type="text"/>

Log

프로젝트 3 개요

- ControlSection 방식의 Object Program(프로젝트 2의 결과물)을 메모리에 올려 실행시키는 시뮬레이터

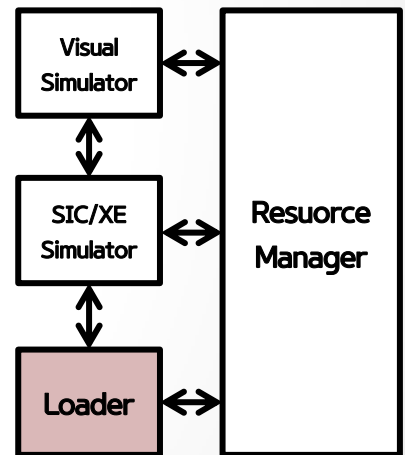


프로젝트 3 명세

Loader

- 어셈블러의 결과물(Object Program)을 읽고 파싱
- Resource Manger에 존재하는 가상 메모리에 Object Program을 적절한 위치에 올림
- load와 readLine 메서드

```
//파일로부터 오브젝트 프로그램을 읽어와 파싱 후 메모리에 로드
public void load(File objFile){
    Scanner scanner = new Scanner(objFile);
    ...
    while(scanner.hasNextLine()) {
        String line = scanner.nextLine();
        readLine(line);
    }
    ...
    resourceManager.affectVisualSimulator();
}
```

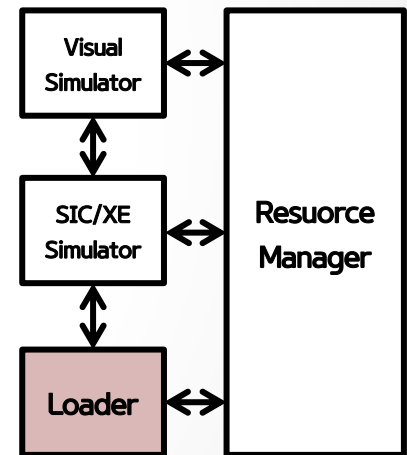


프로젝트 3 명세

Loader

- 어셈블러의 결과물(Object Program)을 읽고 파싱
- Resource Manger에 존재하는 가상 메모리에 Object Program을 적절한 위치에 올림
- load와 readLine 메서드

```
//오브젝트 프로그램의 한 줄을 읽고, 각 헤더(H, T, M 등)에 맞는 기능을 수행
public void readLine(String line){
    if( line.charAt(0) == 'H' ) {
        handleHeader(line.substring(1));
    } else if( line.charAt(0) == 'D' ) {
        handleExternalDefine(line.substring(1));
    } else if( line.charAt(0) == 'R' ) {
        handleExternalReference(line.substring(1));
    } else if( line.charAt(0) == 'T' ) {
        handleText(line.substring(1));
    } else if( line.charAt(0) == 'M' ) {
        handleModification(line.substring(1));
    } else if( line.charAt(0) == 'E' ) {
        handleEnd(line.substring(1));
    }
}
```



프로젝트 3 명세

Loader

- 헤더 레코드일 경우
- 프로그램 이름, 시작주소, 길이
- 컨트롤 섹션, 심볼 테이블 등

HCOPY 000000001033

The screenshot shows the SIC/XE Simulator window. The 'Header Record' section is highlighted with a red box, containing the following fields:

Field	Value
Program Name	COPY
Start Address	000000
Length	001033

Other visible fields in the simulator include:

- FileName: objectCode.txt
- Open button
- E(End Record) Addr 1st inst: 000000
- Start addr in mem: 000000
- Target addr: 000000
- Instructions list: 172027, 4B101033, 032023, 290000, 332007, 4B10105E, 3F2FEC, 032016, 0F2016, 010003, 0F200A
- Buttons: 실행 (1step), 실행 (all), 종료
- Log area at the bottom.

프로젝트 3 명세

Loader

- 텍스트 레코드일 경우
- 각각의 오브젝트 코드를 해당하는 메모리에 올림

```
resourceManager.setMemory(currSectionStartAddress + startAddress, bytes, bytes.length);
```

T0000001D1720274B1000000320232900003320074B1000003F2FEC0320160F2016

T00001D0D0100030F200A4B1000003E2000

T00003003454F46

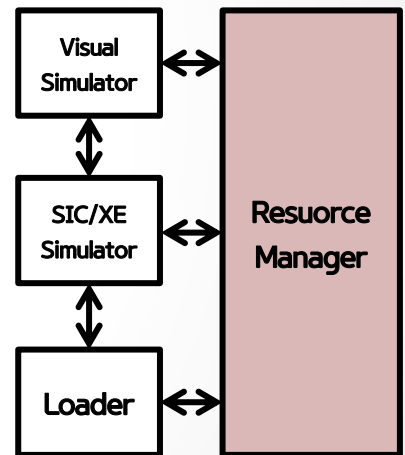
프로젝트 3 명세

Resource Manager

- 시뮬레이터를 구동시키기 위한 가상의 하드웨어
- 메모리영역, 레지스터영역 등
- initializeMemory, setMemory, affectVisualSimulator 등

```
//메모리 영역을 초기화 하는 메소드
public void initializeMemory(){
    ...
}

//메모리 영역에 값을 쓰는 메소드
public void setMemory(int locate, byte[] data, int size) {
    ...
    for (int i = 0; i < size; i++) {
        memory[locate + i] = data[i];
    }
    ...
}
```

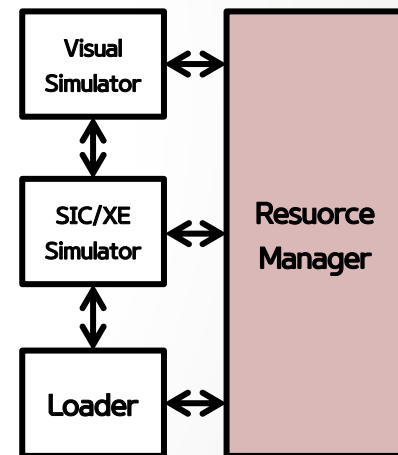


프로젝트 3 명세

Resource Manager

- 시뮬레이터를 구동시키기 위한 가상의 하드웨어
- 메모리영역, 레지스터영역 등
- initializeMemory, setMemory, affectVisualSimulator 등

```
//바뀐 값들을 GUI에 적용시키는 메소드
public void affectVisualSimulator() {
    visualSimulator.updateRegisters(this.registers);
    visualSimulator.updateProgramInformation(programName, lastMemoryAddress);
    visualSimulator.updateMemoryDump();
}
```

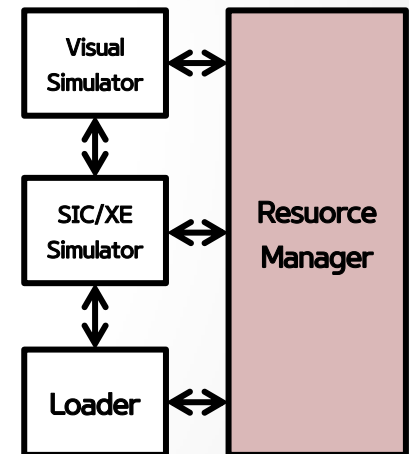


프로젝트 3 명세

Resource Manager

- 시뮬레이터를 구동시키기 위한 가상의 하드웨어
- 메모리영역, 레지스터영역 등
- initializeMemory, setMemory, affectVisualSimulator 등

The screenshot shows the 'SIC/XE 시뮬레이터' window. It includes fields for 'FileName' (objectCode.txt), 'Program Name' (COPY), 'Start Address' (000000), 'Length' (001033), and 'Addr 1th inst' (000000). There are sections for 'Register' with 'Dec' and 'Hex' values for A(#0), X(#1), L(#2), PC(#8), SW(#9), B(#3), S(#4), T(#5), and F(#6). A 'Log' section is at the bottom. The 'Instructions' list includes: 172027, 4B101033, 032023, 290000, 332007, 4B10105E, 3F2FEC, 032016, 0F2016, 010003, and 0F200A. Buttons for '실행 (1step)', '실행 (all)', and '종료' are present.

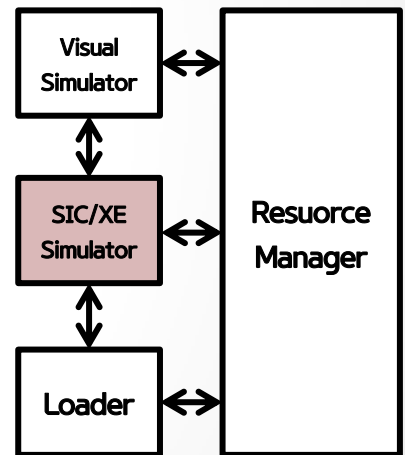


프로젝트 3 명세

SIC/XE Simulator

- 실제로 Object Program을 수행하는 모듈
- Resource Manager에 올라간 Object Program을 실제로 돌아가도록 각각의 명령어 작업을 수행
- Initialize, oneStep, allStep, addLog

```
//각 명령어가 저장되어있는 inst.data파일을 읽고 저장
public void initialize(File instFile){
    File file=new File(instFile);
    ...
    while((line=read.readLine())!=null)
        ...
}
```

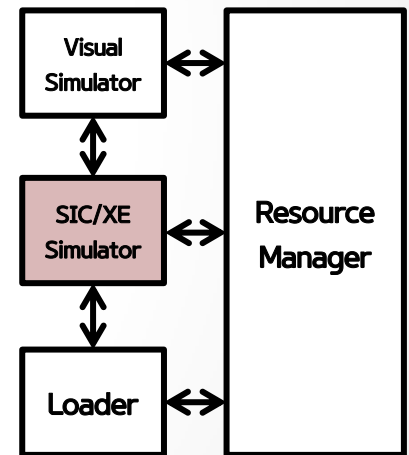


프로젝트 3 명세

SIC/XE Simulator

- 실제로 Object Program을 수행하는 모듈
- Resource Manager에 올라간 Object Program을 실제로 돌아가도록 각각의 명령어가 작업을 수행하도록 구현
- Initialize, oneStep, allStep, addLog

```
//하나의 명령어만 수행한다. 해당 명령어가 수행되고 난 값의 변화를 보여주고,  
//다음 명령어를 포인팅  
public void oneStep() {  
    ...  
    int opcode = getOpCode(resourceManager.getRegisterPC());  
    Hashtable<String, Opcode> table = instTable.getTable();  
    if (table.containsKey(opcode)) {  
        switch (table.get(opcode).getFormat()) {  
            case 1:  
                ...  
            case 2:  
                switch (table.get(opcode).getName()) {  
                    case "CLEAR":  
                        resourceManager.setRegister(objcode.charAt(2), 0);  
                        log += "해당하는 레지스터를 0으로 만듬\n";  
                        break;  
                    case "COMPR":  
                        ...  
                }  
            case 3:  
            case 4:  
                ...  
        }  
    }  
}
```



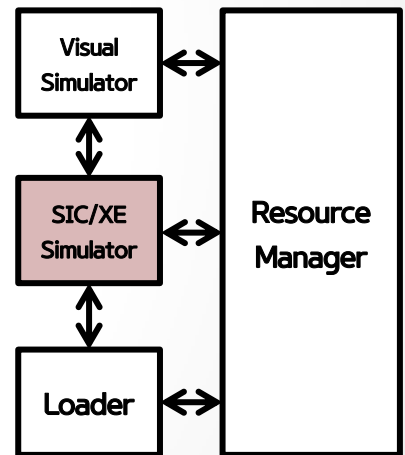
프로젝트 3 명세

SIC/XE Simulator

- 실제로 Object Program을 수행하는 모듈
- Resource Manager에 올라간 Object Program을 실제로 돌아가도록 각각의 명령어가 작업을 수행하도록 구현
- Initialize, oneStep, allStep, addLog

//남은 명령어를 모두 수행하는 메소드

```
public void allStep() {  
    ...  
    while(true) {  
        oneStep();  
        ...  
        if(...) break;  
    }  
}
```



프로젝트 3 명세

SIC/XE Simulator

- 실제로 Object Program을 수행하는 모듈
- Resource Manager에 올라간 Object Program을 실제로 돌아가도록 각각의 명령어가 작업을 수행하도록 구현
- Initialize, oneStep, allStep, addLog

```
//실행한 결과를 로그에 추가하는 메소드
public void addLog() {
    visualSimulator.updateLogs(log);
}
```

The screenshot shows the SIC/XE Simulator interface. At the top, there's a title bar 'SIC/XE 시뮬레이터'. Below it, a 'FileName' field contains 'objectCode.txt' with an 'Open' button. The 'H(Header Record)' section includes 'Program Name' (COPY), 'Start Address' (000000), and 'Length' (001033). The 'E(End Record)' section includes 'Addr 1th inst' (000000). Below these are 'Register' fields for A(#0), X(#1), L(#2), PC(#8), SW(#9), B(#3), S(#4), T(#5), and F(#6), each with 'Dec' and 'Hex' input boxes. The 'Start addr in mem' and 'Target addr' fields are both set to 000000. The 'Instructions' list on the right includes 4B10105E, 3F2FEC, 032016, 0F2016, 010003, 0F200A, 4B10105E, 3E2000, 454F46, B410, and B400. To the right of the instructions are buttons for '실행 (1step)', '실행 (all)', and '종료'. At the bottom, a 'Log' section contains text: '레지스터의 값을 RETADR 에 저장', '서브루틴 호출', and '해당하는 레지스터를 0으로 만듦'.

프로젝트 3 명세

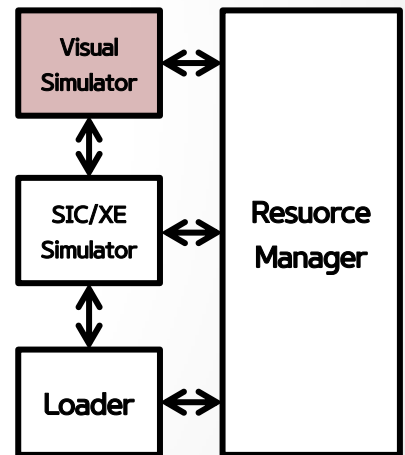
Visual Simulator

- 시뮬레이터의 동작을 GUI 기반으로 보여주는 모듈
- SIC/XE Simulator를 동작시킨 이후 Resource Manager 내의 값들을 읽어 들여 보여줌
- Initialize, oneStep, allStep

```
//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)
```



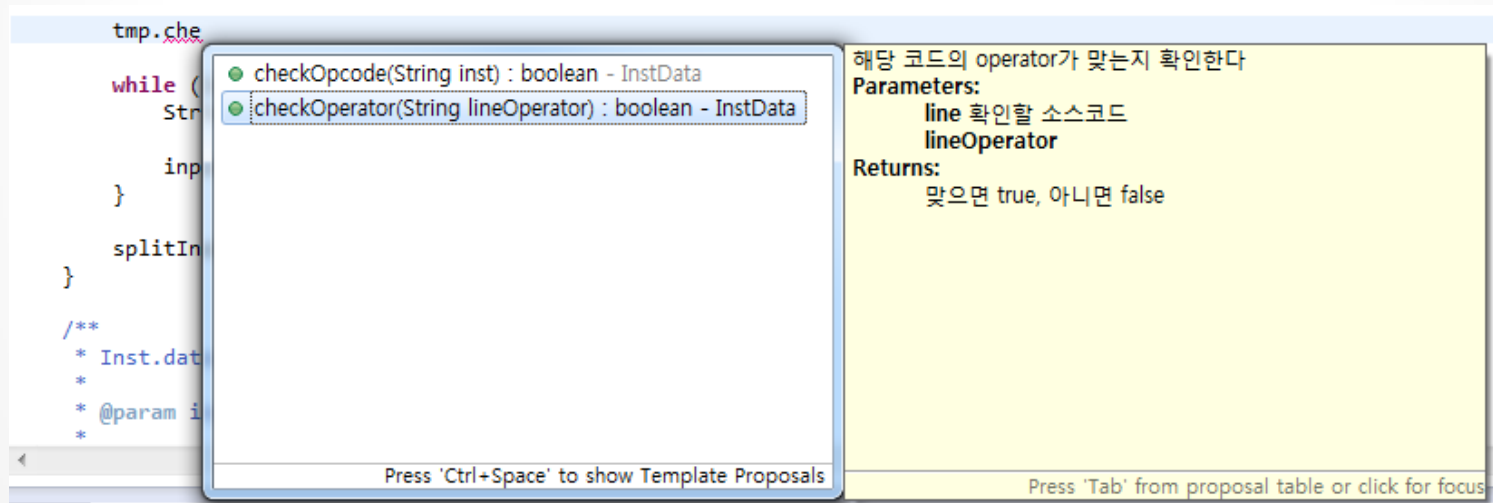
Java Doc

- 제공하는 모든 기능들을 문서로 제공
- Eclipse에서 메서드의 선언부 만든 후 그 위에서 `/**` 엔터를 입력하면 자동 생성

```
/**
 * 해당 코드의 operator가 맞는지 확인한다
 * @param line 확인할 소스코드
 * @return 맞으면 true, 아니면 false
 */
public boolean checkOperator(String lineOperator) {
    //자신의 operator와 맞는지 비교하여 리턴
    if ( operator.equals(lineOperator) ) {
        return true;
    }
    return false;
}
```

Java Doc

- 제공하는 모든 기능들을 문서로 제공
- Eclipse에서 메서드의 선언부 만든 후 그 위에서 `/**` 엔터를 입력하면 자동 생성



질의응답