숭실대학교 Soongsil University

# 시스템프로그래밍 2021 보고서

## 보고서 제출서약서

나는 숭실대학교 컴퓨터학부의 일원으로 명예를 지키면서 생활하고 있습니다.

나는 보고서를 작성하면서 다음과 같은 사항을 준수하였음을 엄숙히 서약합니다.

1. 나는 자력으로 보고서를 작성하였습니다.
   1.1. 나는 동료의 보고서를 베끼지 않았습니다.
   1.2. 나는 비공식적으로 얻은 해답/해설을 기초로 보고서를 작성하지 않았습니다.
2. 나는 보고서에서 참조한 문헌의 출처를 밝혔으며 표절하지 않았습니다. (나는 특히 인터넷에서 다운로드한 내용을 보고서에 거의 그대로 복사하여 사용하지 않았습니다.)
3. 나는 보고서를 제출하기 전에 동료에게 보여주지 않았습니다.
4. 나는 보고서의 내용을 조작하거나 날조하지 않았습니다.

| 과목 | 시스템프로그래밍 2021 |
|---|---|
| 과제명 | 프로젝트2 |
| 담당교수 | 최 재 영 교 수 |
| 제출인 | 컴퓨터학부 20192698 심원준 (출석번호 124번) |
| 제출일 | 2021년 6월 6일 |

# 차 례

## 1.프로젝트 동기/목적

이 프로젝트는 SIC/XE Object Code를 실행하며 이 과정을 보여줄 수 있는 로더와 시뮬레이터를 포함한 JAVA GUI프로그램을 제작함을 목적으로 한다.

## 2.설계/구현 아이디어

SIC Loader/ResourceManager/InstLuncher/SicSimulator/SymbolTable/VisualSimulator의 6개 .java(class)와 Instluncher와 SicSimulator에서 명령어를 분석하기위한 instruction class로 구성되어있다.

가.SicLoader

입력된 objectcode를 memory에 올리는 것을 목표로한다.
objectcode 각 줄의 첫 글자를 읽어 'H','T','D','R','M'에 따라 분류하여 동작을 달리한다.
첫 pass에서는 'D'의 줄을 읽어와 symbolTable 클래스의 인스턴스에 extdef 심볼을 저장한다.
두 번째 pass에서는 'T' 줄을 읽어와 메모리에 올리고, 첫 글자가 'M'인 modification code에 정의된 위치에 symbolTable에서 찾아온 주소를 더하거나 뺀다.
이 과정을 마치면 objectcode가 memory에 올라가게된다.

나.ResourceManager

ResourceManager은 SicLoader가 objectcode를 올릴 메모리가 char[]배열로 정의되어있으며, register들 또한 정의되어있다. 또한 Memory에 저장하고, 찾아올 수 있도록 하는 getMemory, setMemory메소드가 정의되어있다. Register에 저장하고 찾아올 수 있는 setRegister, getRegister 또한 정의되어있다. WD,TD,RD작업에 이용될 디바이스를 관리하는 deviceManager arraylist가 정의되어있으며 이들 메소드들 또한 정의되어있다.

다.InstLuncher

InstLuncher에는 각 명령어에 따른 동작이 정의되어있다. SIC/XE Appendix에 나와있는 명령어들을 정의해 두었으며, 이들이 하는 동작에 따라 resourceManager의 memory/register/device가 수정 및 동작한다. 또한 Instruction class 또한 정의되어있는데, 각 명령어가 작동될 때 받은 명령어와 주소값을 받아서 flag/대상 주소/레지스터 값을 분석하는 역할을 한다.

라.SIC Simulator

Sic Simulator는 visual Simulator로부터 받은 event에 따라서 동작을 실행하게 하는 역할을 한다. visual SImulator로부터 요청을 받아 instLuncher의 명령어 메소드를 실행하게 하며,

이 결과를 다시 visualSimulator로 보내도록 한다.

마. Visual Simulator

visual Simulator는 GUI프로그램의 frame을 정의하며, 버튼입력등의 동작(event)를 관리할 eventHandler 또한 정의되어있어 이에 따라 SIC SImulator로 요청과 응답을 주고받는다.

3.실행결과

visual simulator를 구성하지 못하여 디버깅 및 RUN 을 하지 못하였다.

## 4.결론 및 보충할 점

SicSimulator와 visualSimulator의 두 클래스를 완전히 구현하지 못하여 프로그램을 완성하지 못하였다. Java의 Swing을 이용해 GUI를 구현하는 데에 알지 못하여 이에 어려움을 겪었다. 또한 Loader와 linker의 동작에 대해서 명확하게 이해를 하지 못하여 구현할 아이디어(세부적 로직)을 구성하는 데에 어려움을 겪어 이에 대한 추가적 학습이 필요하다.

## 5.디버깅

visual simulator를 구성하지 못하여 디버깅을 하지 못하였다.

## 6.소스코드

```
package sp21_simulator;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

/**
 * SicLoader는 프로그램을 해석해서 메모리에 올리는 역할을 수행한다. 이 과정에서 linker의 역할 또한 수행한다.
 *
 * SicLoader가 수행하는 일을 예를 들면 다음과 같다. - program code를 메모리에 적재시키기 - 주어진 공간만큼 메모리에 빈
 * 공간 할당하기 - 과정에서 발생하는 symbol, 프로그램 시작주소, control section 등 실행을 위한 정보 생성 및 관리
 */
public class SicLoader {
	ResourceManager rMgr;

	public SicLoader(ResourceManager resourceManager) {
		// 필요하다면 초기화

		setResourceManager(resourceManager);
	}

	/**
	 * Loader와 프로그램을 적재할 메모리를 연결시킨다.
	 *
	 * @param rMgr
	 */
	public void setResourceManager(ResourceManager resourceManager) {
		this.rMgr = resourceManager;
	}

	/**
	 * object code를 읽어서 load과정을 수행한다. load한 데이터는 resourceManager가 관리하는 메모리에 올라가도록
	 * 한다. load과정에서 만들어진 symbol table 등 자료구조 역시 resourceManager에 전달한다.
	 *
	 * @param objectCode 읽어들인 파일
	 * @throws IOException
	 */
	int sttadd=0;
	public void load(File objectCode) throws IOException {
		FileReader finput=new FileReader(objectCode);
		BufferedReader bufread=new BufferedReader(finput);
		String line=new String();
		int sttadd=0;//starting address of program
		int length=0;
```

```
while((line=bufread.readLine())!=null) {

if(line.charAt(0)=='D')
                        {
                                for(int i=1;i<line.length();i+=12)
                                {

rMgr.symtabList.putSymbol(line.substring(i,i+6),rMgr.proglength+Integer.parseInt(line.substring(i+6,i+12),16));
                                }
                        }
                        else if(line.charAt(0)=='H')
                        {

rMgr.proglength+=length;

length=Integer.parseInt(line.substring(13),16);
                        }
                }
                bufread.close();
                finput.close();
                FileReader finput2=new FileReader(objectCode);
                BufferedReader bufread2=new BufferedReader(finput2);
                length=0;
                rMgr.proglength=0;
                int count=0;

while((line=bufread2.readLine())!=null) {

if(line.charAt(0)=='H')//Header
                                {

if(count==0)//첫 프로그램의 경우
                                        {


length=Integer.parseInt(line.substring(13),16);

sttadd=Integer.parseInt(line.substring(7,13),16);

rMgr.sttadd=sttadd;

length=Integer.parseInt(line.substring(13),16);

rMgr.symtabList.putSymbol(line.substring(1,7), sttadd);

count+=1;
                                        }
                                        else
                                        {

rMgr.proglength+=length;

length=Integer.parseInt(line.substring(13),16);

rMgr.symtabList.putSymbol(line.substring(1,7),rMgr.proglength );
                                        }
                                }
                                else
```

```
if(line.charAt(0)=='M')
                {
                    float fixadd=rMgr.proglength+(float)Integer.parseInt(line.substring(1,7),16);
                    int numf=Integer.parseInt(line.substring(7,9));

if(numf%2==1)//홀수일경우 0.5더해주어야 함 주소값
                    {

fixadd+=0.5;
                    }

if(line.charAt(9)=='+')
                    {
                        int temperary=0;

temperary=Integer.parseInt(String.valueOf(rMgr.getMemory(fixadd, numf)),16);

String refsym=new String();

refsym=line.substring(10);
                        int toadd=rMgr.symtabList.search(refsym);

temperary+=toadd;

rMgr.setMemory(fixadd,Integer.toHexString(temperary).toCharArray(), numf);
                    }
                    else
                    {
                        int temperary=0;

temperary=Integer.parseInt(String.valueOf(rMgr.getMemory(fixadd, numf)),16);

String refsym=new String();

refsym=line.substring(10);
                        int toadd=rMgr.symtabList.search(refsym);

temperary-=toadd;

rMgr.setMemory(fixadd,Integer.toHexString(temperary).toCharArray(), numf);
                    }
                }
                else if(line.charAt(0)=='T')
                {
                    int stad=Integer.parseInt(line.substring(1,7),16);
                    int lenofline=0;

lenofline=Integer.parseInt(line.substring(7,9),16);

rMgr.setMemory(stad, line.substring(9).toCharArray(), lenofline);
                }
            }
            rMgr.proglength+=length;
```

```
rMgr.setRegister(2,rMgr.proglength);


	};


}


package sp21_simulator;

class Instruction{
	char[]instruction;
	int format=0;
	Instruction(char[]instruction){

this.instruction=instruction.clone();
	}
	int[] getflag(){
		int []flags=new int[6];
		format=3;
		for(int i=0;i<6;i++)
		{
			flags[i]=0;
		}

if((this.instruction[1]&2)==2)//n
		{
			flags[0]=1;
		}

if((this.instruction[1]&1)==1)//i
		{
			flags[1]=1;
		}

if((this.instruction[2]&8)==8)//x
		{
			flags[2]=1;
		}

if((this.instruction[2]&4)==4)//b
		{
			flags[3]=1;
		}

if((this.instruction[2]&2)==2)//p
		{
			flags[4]=1;
		}

if((this.instruction[2]&1)==1)//e
		{
			flags[5]=1;
			format=4;
		}
		return flags;
	}
	int addr(int curr,int[] flags) {
		int returningadd=0;
		if(flags[4]==1)//pc
relative
		{
			curr+=format;
			String
temp=this.instruction.toString();

returningadd=Integer.parseInt(temp.sub
string(3),16);

returningadd+=curr;
		}
		else//ref
		{
			String
temp=this.instruction.toString();

returningadd=Integer.parseInt(temp.sub
string(3),16);
		}
		return returningadd;
	}
	int [] registers() {
		int []temp=new int[2];

temp[0]=Character.getNumericValue(in
struction[2]);

temp[1]=Character.getNumericValue(in
struction[3]);
		format=2;
		return temp;
	}
	int format()
	{
		return format;
	}
}

package sp21_simulator;
import java.awt.*;
import java.awt.event.ActionListener;

import javax.swing.*;
import java.io.File;
import java.io.IOException;

/**
 * VisualSimulator는 사용자와의
상호작용을 담당한다. 즉, 버튼 클릭등의
이벤트를 전달하고 그에 따른 결과값을
화면에 업데이트
 * 하는 역할을 수행한다.
 *
 * 실제적인 작업은 SicSimulator에서
```

```
수행하도록 구현한다.
 */

@SuppressWarnings("serial")
public class VisualSimulator extends JFrame {
	ResourceManager resourceManager = new ResourceManager();
	SicLoader sicLoader = new SicLoader(resourceManager);
	SicSimulator sicSimulator = new SicSimulator(resourceManager);
	static JFrame fr;

	/**
	 * 프로그램 로드 명령을 전달한다.
	 * @throws IOException
	 */
	public void load(File program) throws IOException {
		// ...
		sicLoader.load(program);

sicSimulator.load(program);
	};

	/**
	 * 하나의 명령어만 수행할 것을 SicSimulator에 요청한다.
	 */
	public void oneStep() {

	};

	/**
	 * 남아있는 모든 명령어를 수행할 것을 SicSimulator에 요청한다.
	 */
	public void allStep() {

	};

	/**
	 * 화면을 최신값으로 갱신하는 역할을 수행한다.
	 */
	public void update() {

	};

	public static void main(String[] args) {
		fr=new JFrame("SP21 pro2 20192698");
		fr.setSize(700,300);
		fr.setVisible(true);
fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Container contentpane=fr.getContentPane();

	}
}

package sp21_simulator;

import java.util.ArrayList;

/**
 * symbol과 관련된 데이터와 연산을 소유한다. section 별로 하나씩 인스턴스를 할당한다.
 */
public class SymbolTable {
	ArrayList<String> symbolList;
	ArrayList<Integer> addressList;
	// 기타 literal, external 선언 및 처리방법을 구현한다.

	/**
	 * 새로운 Symbol을 table에 추가한다.
	 *
	 * @param symbol : 새로 추가되는 symbol의 label
	 * @param address : 해당 symbol이 가지는 주소값 <br>
	 *                <br>
	 *                주의 : 만약 중복된 symbol이 putSymbol을 통해서 입력된다면 이는 프로그램 코드에 문제가 있음을
	 *                나타낸다. 매칭되는 주소값의 변경은 modifySymbol()을 통해서 이루어져야 한다.
	 */
	public void putSymbol(String symbol, int address) {

this.symbolList.add(symbol);

this.addressList.add(address);
	}

	/**
	 * 기존에 존재하는 symbol 값에 대해서 가리키는 주소값을 변경한다.
	 *
	 * @param symbol : 변경을 원하는 symbol의 label
	 * @param newaddress : 새로 바꾸고자 하는 주소값
```

```
         */
        public void
modifySymbol(String symbol, int
newaddress) {
                int
tindex=this.symbolList.indexOf(symbol)
;
                addressList.set(tindex,
newaddress);
        }

        /**
         * 인자로 전달된 symbol이 어떤
주소를 지칭하는지 알려준다.
         *
         * @param symbol : 검색을
원하는 symbol의 label
         * @return symbol이 가지고
있는 주소값. 해당 symbol이 없을 경우
-1 리턴
         */
        public int search(String symbol)
{
                int address =
this.symbolList.indexOf(symbol);
                return address;
        }

}
package sp21_simulator;

import java.io.IOException;
import java.util.ArrayList;

// instruction에 따라 동작을 수행하는
메소드를 정의하는 클래스

public class InstLuncher {
    ResourceManager rMgr;
    public
InstLuncher(ResourceManager
resourceManager) {
         this.rMgr = resourceManager;
    }

    public void add(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        int temp=rMgr.getRegister(0);
        rMgr.setRegister(0, temp+ add);
    }
    public void ADDR(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []regs=temi.registers();
        int
r1=rMgr.getRegister(regs[0]);
        int
r2=rMgr.getRegister(regs[1]);
        rMgr.setRegister(r2, r2+ r1);
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void CLEAR(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []regs=temi.registers();
        rMgr.setRegister(regs[0],
0xFF);
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void COMP(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        if(add==rMgr.getRegister(0))
        {
                rMgr.setRegister(9, 0);
        }
        else if(rMgr.getRegister(0)<add)
        {
                rMgr.setRegister(9, -1);
        }
        else
        {
                rMgr.setRegister(9, 1);
        }
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void COMPR(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []regs=temi.registers();

if(rMgr.getRegister(regs[0])==rMgr.get
Register(regs[1]))
        {
                rMgr.setRegister(9,0);
        }
```

```
        else
if(rMgr.register[regs[0]]<rMgr.registe
r[regs[1]])
        {
                rMgr.setRegister(9, -1);
        }
        else
        {
                rMgr.setRegister(9, 1);
        }
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void DIV(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);

rMgr.setRegister(0,rMgr.getRegister(0)
/add);
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void DIVR(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []regs=temi.registers();
        rMgr.setRegister(regs[1],
rMgr.getRegister(regs[1])/rMgr.getReg
ister(regs[0]));
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void J(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        rMgr.setRegister(8, add);
        //pc register에 대상
address저장
    }
    public void JEQ(int
currloc,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(currloc,flags);
        if(rMgr.getRegister(9)==0)
        {
                rMgr.setRegister(8,add);
        }
        else
        {
                rMgr.setRegister(8,
currloc+ temi.format());
        }
    }
    public void JLT(int
currloc,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(currloc,flags);
        if(rMgr.getRegister(9)<0)
        {
                rMgr.setRegister(8,add);
        }
        else
        {

rMgr.setRegister(8,currloc+ temi.forma
t());
        }
    }
    public void JSUB(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        rMgr.setRegister(2,
rMgr.getRegister(8));
        rMgr.setRegister(8, add);
    }
    public void LDA(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        rMgr.setRegister(0, add);
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void LDCH(int
curr,char[]instruction)
    {
        Instruction temi=new
```

```
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        int
target=add+(rMgr.getRegister(1)/2);
        char []
temp=rMgr.getMemory(target, 1);
        rMgr.setRegister(0,temp);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void RD(int
curr,char[]instruction) throws
IOException
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        String
devname=String.valueOf(rMgr.getMem
ory(add,2));

rMgr.setRegister(0,rMgr.readDevice(de
vname, 1));
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void RSUB(int
curr,char[]instruction)
    {
        rMgr.setRegister(8,
rMgr.getRegister(2));
    }
    public void STA(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);

rMgr.setMemory(add,Integer.toHexStri
ng(rMgr.getRegister(0)).toCharArray(),
1);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void STB(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);

rMgr.setMemory(add,Integer.toHexStri
ng(rMgr.getRegister(3)).toCharArray(),
1);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void STCH(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        char[]temp=new char[1];

temp[0]=(char)rMgr.getRegister(0);
        rMgr.setMemory(add, temp, 1);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void STL(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        char[]temp=new char[1];

temp[0]=(char)rMgr.getRegister(2);
        rMgr.setMemory(add, temp, 1);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public void STX(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        char[]temp=new char[1];

temp[0]=(char)rMgr.getRegister(1);
        rMgr.setMemory(add, temp, 1);
        rMgr.setRegister(8,
curr+temi.format());
    }
    public int TD(int
curr,char[]instruction) throws
IOException
    {
        Instruction temi=new
```

```
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        char[]temp=new char[2];
        rMgr.getMemory(add, 2);
        int
res=rMgr.testDevice(String.valueOf(te
mp));
        rMgr.setRegister(8,
curr+ temi.format());
        return res;
    }
    public void TIXR(int
curr,char[]instruction)
    {
        Instruction temi=new
Instruction(instruction);
        int []regs=temi.registers();
        rMgr.setRegister(1,
rMgr.getRegister(1)+ 1);
        int x=rMgr.getRegister(1);
        if(x<rMgr.getRegister(regs[0]))
        {
                rMgr.setRegister(2,-1);
        }
        else
if(x==rMgr.getRegister(regs[0]))
        {
                rMgr.setRegister(2, 0);
        }
        else
        {
                rMgr.setRegister(2, 1);
        }
        rMgr.setRegister(8,
curr+ temi.format());
    }
    public void WD(int
curr,char[]instruction) throws
IOException
    {
        Instruction temi=new
Instruction(instruction);
        int []flags=new int[6];
        flags=temi.getflag();
        int add=temi.addr(curr,flags);
        String
devname=String.valueOf(rMgr.getMem
ory(add,2));
        char[]temp=new char[1];
        temp[0]=(char)
rMgr.getRegister(0);

rMgr.writeDevice(devname,temp, 1);
    }
    }
```

```
package sp21_simulator;

import java.io.File;
import java.io.IOException;

/**
 * 시뮬레이터로서의 작업을 담당한다.
VisualSimulator에서 사용자의 요청을
받으면 이에 따라 ResourceManager에
접근하여
 * 작업을 수행한다.
 *
 * 작성중의 유의사항 : 1) 새로운
클래스, 새로운 변수, 새로운 함수 선언은
얼마든지 허용됨. 단, 기존의 변수와
함수들을 삭제하거나
 * 완전히 대체하는 것은 지양할 것. 2)
필요에 따라 예외처리, 인터페이스 또는
상속 사용 또한 허용됨. 3) 모든 void
타입의 리턴값은
 * 유저의 필요에 따라 다른 리턴
타입으로 변경 가능. 4) 파일, 또는
콘솔창에 한글을 출력시키지 말 것.
(채점상의 이유. 주석에 포함된
 * 한글은 상관 없음)
 *
 *
 *
 * + 제공하는 프로그램 구조의
개선방법을 제안하고 싶은 분들은
보고서의 결론 뒷부분에 첨부 바랍니다.
내용에 따라 가산점이 있을 수
 * 있습니다.
 */
public class SicSimulator {
        ResourceManager rMgr;
        SicLoader load;
        InstLuncher lunch;
        int currloc;
        public
SicSimulator(ResourceManager
resourceManager) {
                // 필요하다면 초기화 과정
추가
                this.rMgr =
resourceManager;
                this.lunch=new
InstLuncher(resourceManager);
        }

        /**
         * 레지스터, 메모리 초기화 등
프로그램 load와 관련된 작업 수행. 단,
object code의 메모리 적재 및 해석은
         * SicLoader에서 수행하도록
한다.
         * @throws IOException
```

```
         */
        public void load(File program)
throws IOException {
                load=new
SicLoader(rMgr);
                load.load(program);
                currloc=rMgr.sttadd;
        }

        /**
         * 1개의 instruction이 수행된
모습을 보인다.
         * @throws IOException
         */
        public void oneStep() throws
IOException {

char[]op=rMgr.getMemory(currloc, 1);
                String
opc=String.valueOf(op);
                char
[]eflag=rMgr.getMemory(currloc+1,1);
                String
ef=String.valueOf(eflag);
                int
efla=Integer.parseInt(ef.substring(0,1),
16);
                boolean e;
                if((efla&1)==1)
                {
                        e=true;
                }
                else
                {
                        e=false;
                }
                if(opc.equals("1B"))//ADD
                {
                        if(e==true)
                        {

lunch.add(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.add(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals(90))//ADDR
                {

lunch.ADDR(currloc,
rMgr.getMemory(currloc,2));

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("B4"))//clear
                {

lunch.CLEAR(currloc,
rMgr.getMemory(currloc,2));

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("2B"))//COMP
                {
                        if(e==true)
                        {

lunch.COMP(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.COMP(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("A0"))//COMPR
                {

lunch.COMPR(currloc,
rMgr.getMemory(currloc,2));

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("27"))//DIV
                {
                        if(e==true)
                        {

lunch.DIV(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.DIV(currloc,
```

```
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("9C"))//DIVR
                {

lunch.DIVR(currloc,
rMgr.getMemory(currloc,2));

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("3F"))//J
                {
                        if(e==true)
                        {

lunch.J(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.J(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("33"))//JEQ
                {
                        if(e==true)
                        {

lunch.JEQ(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.JEQ(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("3B"))//JLT
                {
                        if(e==true)
                        {

lunch.JLT(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.JLT(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("4B"))//JSUB
                {
                        if(e==true)
                        {

lunch.JSUB(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.JSUB(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("03")||opc.equals("01"))//
LDA
                {
                        if(e==true)
                        {

lunch.LDA(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.LDA(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("53"))//LDCH
                {
                        if(e==true)
```

```
                        {
lunch.LDCH(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.LDCH(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("DB"))//RD
                {
                        if(e==true)
                        {
lunch.RD(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.RD(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("4F"))//RSUB
                {
                        if(e==true)
                        {
lunch.RSUB(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.RSUB(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("0F"))//STA
                {
                        if(e==true)
                        {
lunch.STA(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.STA(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("7B"))//STB
                {
                        if(e==true)
                        {
lunch.STB(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.STB(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("57"))//STCH
                {
                        if(e==true)
                        {
lunch.STCH(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {
lunch.STCH(currloc,
rMgr.getMemory(currloc,3));
                        }
currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("17"))//STL
                {
                        if(e==true)
                        {
```

```
lunch.STL(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.STL(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("13"))//STX
                {
                        if(e==true)
                        {

lunch.STX(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.STX(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("E3"))//TD
                {
                        if(e==true)
                        {

lunch.TD(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.TD(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }
                else
if(opc.equals("B8"))//TIXR
                {

lunch.TIXR(currloc,
rMgr.getMemory(currloc,2));

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동.
                }
                else
if(opc.equals("DF"))//WD
                {
                        if(e==true)
                        {

lunch.WD(currloc,
rMgr.getMemory(currloc,4));
                        }
                        else
                        {

lunch.WD(currloc,
rMgr.getMemory(currloc,3));
                        }

currloc=rMgr.getRegister(8);//PC레지스
터의 주소로 이동
                }

        }

        /**
         * 남은 모든 instruction이
수행된 모습을 보인다.
         * @throws IOException
         */
        public void allStep() throws
IOException {
                while(true)
                {

if(currloc==rMgr.proglength)
                        {
                                break;
                        }
                        else
                        {
                                oneStep();
                        }
                }
        }

        /**
         * 각 단계를 수행할 때 마다
관련된 기록을 남기도록 한다.
         */
        public void addLog(String log)
{
        }
}


package sp21_simulator;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;

/**
 * ResourceManager는 컴퓨터의 가상 리소스들을 선언하고 관리하는
클래스이다. 크게 네가지의 가상 자원 공간을 선언하고, 이를
 * 관리할 수 있는 함수들을 제공한다.
 *
 *
 * 1) 입출력을 위한 외부 장치 또는 device 2) 프로그램 로드 및 실행을 위한
메모리 공간. 여기서는 64KB를 최대값으로 잡는다.
 * 3) 연산을 수행하는데 사용하는 레지스터 공간. 4) SYMTAB 등
simulator의 실행 과정에서 사용되는 데이터들을 위한 변수들.
 *
 * 2번은 simulator위에서 실행되는 프로그램을 위한 메모리공간인 반면,
4번은 simulator의 실행을 위한 메모리 공간이라는 점에서
 * 차이가 있다.
 */
public class ResourceManager {
	/**
	 * 디바이스는 원래 입출력 장치들을 의미 하지만 여기서는 파일로
디바이스를 대체한다. 즉, 'F1'이라는 디바이스는 'F1'이라는 이름의
	 * 파일을 의미한다.
deviceManager는 디바이스의 이름을 입력받았을 때 해당 이름의 파일 입출력
관리 클래스를 리턴하는 역할을 한다.
	 * 예를 들어, 'A1'이라는
디바이스에서 파일을 read모드로 열었을 경우, hashMap에 <"A1", scanner(A1)>
등을
	 * 넣음으로서 이를 관리할 수 있다.
	 *
	 * 변형된 형태로 사용하는 것 역시 허용한다. 예를 들면 key값으로
String대신 Integer를 사용할 수 있다. 파일 입출력을 위해
	 * 사용하는 stream 역시 자유로이 선택, 구현한다.
	 *
	 * 이것도 복잡하면 알아서 구현해서 사용해도 괜찮습니다.
	 */
	HashMap<String, Integer>
deviceManager = new
HashMap<String,Integer>();
	char[] memory; // String으로
수정해서 사용하여도 무방함.
	int[] register;
	double register_F;
	int proglength=0;
	int sttadd;
	SymbolTable symtabList;
	// 이외에도 필요한 변수 선언해서
사용할 것.

	/**
	 * 메모리, 레지스터등 가상
리소스들을 초기화한다.
	 */
	public void initializeResource()
{
		memory= new
char[65536];
		register = new int[10];
	}

	/**
	 * deviceManager가 관리하고
있는 파일 입출력 stream들을 전부 종료시키는 역할. 프로그램을 종료하거나
연결을 끊을 때
	 * 호출한다.
	 * @throws IOException
	 */
	public void closeDevice()
throws IOException {
		reader.close();
		writer.close();
	}

	/**
	 * 디바이스를 사용할 수 있는
상황인지 체크. TD명령어를 사용했을 때 호출되는 함수. 입출력 stream을 열고
deviceManager를
	 * 통해 관리시킨다.
	 *
	 * @param devName
확인하고자 하는 디바이스의 번호,또는 이름
	 */

	@SuppressWarnings("resource")
	public int testDevice(String
devName) throws IOException {
		FileReader ft=new
FileReader (devName);
		if(ft.ready())
```

```
                {

if(deviceManager.get(devName)==null)
                        {

deviceManager.put(devName, 0);
                                return 1;
                        }
                }
                else
                {
                        return 0;
                }
                ft.close();
                return 1;
        }

        /**
         * 디바이스로부터 원하는
개수만큼의 글자를 읽어들인다.
RD명령어를 사용했을 때 호출되는 함수.
         *
         * @param devName
디바이스의 이름
         * @param num       가져오는
글자의 개수
         * @return 가져온 데이터
         * @throws IOException
         */
        FileReader reader;
        public char[] readDevice(String
devName, int num) throws
IOException {

if(deviceManager.get(devName)==0)
                {

deviceManager.replace(devName, 1);
                        reader=new
FileReader(devName);
                }
                char[]temp=new
char[num];
                reader.read(temp,0,num);
                return temp;
        }

        /**
         * 디바이스로 원하는 개수
만큼의 글자를 출력한다. WD명령어를
사용했을 때 호출되는 함수.
         *
         * @param devName
디바이스의 이름
         * @param data      보내는
데이터
         * @param num       보내는
글자의 개수
         * @throws IOException
         */
        FileWriter writer;
        public void writeDevice(String
devName, char[] data, int num)
throws IOException {

if(deviceManager.get(devName)==0)
                {

deviceManager.replace(devName, 1);
                        writer=new
FileWriter(devName);
                }
                for(int i=0;i<num;i++)
                {

writer.write(data[i]);
                }
        }

        /**
         * 메모리의 특정 위치에서
원하는 개수만큼의 글자를 가져온다.
         *
         * @param location 메모리 접근
위치 인덱스
         * @param num       데이터
개수
         * @return 가져오는 데이터
         */
        public char[] getMemory(float
location, int num) {
                int loc=(int)location*2;
                char[]retch=new
char[num*2];

retch=Arrays.copyOfRange(memory,
loc, loc+(num*2));
                return retch;

        }

        /**
         * 메모리의 특정 위치에 원하는
개수만큼의 데이터를 저장한다.
         *
         * @param locate 접근 위치
인덱스
         * @param data    저장하려는
데이터
         * @param num     저장하는
데이터의 개수
         */
        public void setMemory(int
locate, char[] data, int num) {
                int loc=locate*2;
                for(int
```

```
i=loc;i<loc+(num*2);i++)
                {

memory[i]=data[i-loc];
                }

        }
        public void setMemory(float
locate,char[]data,int num) {
                int loc=(int)locate*2;
                for(int
i=loc;i<loc+(num*2);i++)
                {

memory[i]=data[i-loc];
                }
        }

        /**
         * 번호에 해당하는 레지스터가
현재 들고 있는 값을 리턴한다.
레지스터가 들고 있는 값은 문자열이
아님에 주의한다.
         *
         * @param regNum 레지스터
분류번호
         * @return 레지스터가 소지한
값
         */
        public int getRegister(int
regNum) {
                return register[regNum];

        }

        /**
         * 번호에 해당하는 레지스터에
새로운 값을 입력한다. 레지스터가 들고
있는 값은 문자열이 아님에 주의한다.
         *
         * @param regNum 레지스터의
분류번호
         * @param value  레지스터에
집어넣는 값
         */
        public void setRegister(int
regNum, int value) {
                register[regNum]=value;
        }
        public void setRegister(int
regNum,char[]value) {

register[regNum]=(int)value[0];
        }

        /**
         * 주로 레지스터와 메모리간의
데이터 교환에서 사용된다. int값을
char[]형태로 변경한다.
         *
         * @param data
         * @return
         */
        public char[] intToChar(int
data) {

char[]temp=Integer.toHexString(data).t
oCharArray();
                return temp;
        }

        /**
         * 주로 레지스터와 메모리간의
데이터 교환에서 사용된다. char[]값을
int형태로 변경한다.
         *
         * @param data
         * @return
         */
        public int byteToInt(char[]
data) {
                String
temp=String.valueOf(data);
                return
Integer.parseInt(temp,16);
        }
}
```