숭실대학교 Soongsil University

프로그램 보고서

나는 숭실대학교 전자정보공학부의 일원으로 명예를 지키면서 생활하고 있습니다.

나는 보고서를 작성하면서 다음과 같은 사항을 준수하였음을 엄숙히 서약합니다.

1. 나는 자력으로 보고서를 작성하였습니다.
2. 나는 보고서에서 참조한 문헌의 출처를 밝혔으며 표절하지 않았습니다.
3. 나는 보고서의 내용을 조작하거나 날조하지 않았습니다.

교과목	시스템프로그래밍 2025
프로젝트 명	Project #2 – JAVA로 SIC/XE Loader 및 simulator 만들기
교과목 교수	최 재 영
제출인	전자정보공학(학과) 학번:20203023 성명:우상욱(출석번호: 222)
제출일	2025년 5월 30일

차 례

1장 프로젝트 개요

1.1 동기 및 목적

본 프로젝트는 앞선 프로젝트에서 설계했던 assembler에서 생성한 objectcode를 받아 실행해보는 프로젝트이다. SIC/XE 머신은 ARM이나 Intel과는 다른 실제로 존재하지 않는 가상의 머신이므로 이를 시뮬레이션하기 위해 simulator를 설계하여 가상의 메모리에 load된 명령어들을 실행시키는 과정을 GUI로 테스트해본다. 기존 프로젝트에서는 SIC/XE 머신의 명령어들을 objectcode로 변환하기 위한 작업을 했다면, 다시 objectcode를 역변환하는 과정을 통해 objectcode를 해석하여 그에 맞는 동작을 수행하는 시뮬레이터를 설계한다. 이번 프로젝트까지 함으로써 변환 및 역변환의 과정을 모두 경험하고 컴퓨터 공학의 원리 및 어셈블러에 대한 이해를 높인다.

2장 설계/구현 아이디어

2.1 기술적 배경지식

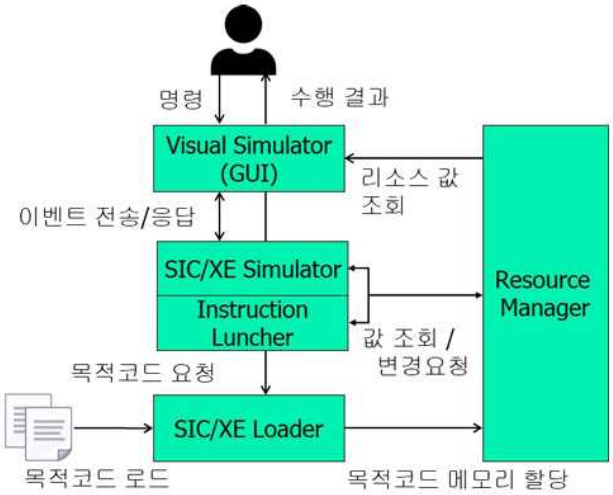
SIC/XE머신은 실제 하드웨어가 아니므로 software로 가상으로 구성한다. 이번 프로젝트의 objectcode는 control section문법을 따른다. 이번에는 총 3개의 section이 들어있다고 가정하고 프로젝트를 수행하였다. objectcode는 다음과 같이 구성되어있다.

- Header Record (H)
  Header Record는 section의 이름과 프로그램 시작 주소, 프로그램의 길이를 담고 있다.
- Definition Record (D)
  내부의 LABEL을 다른 section에게 전달하기 위한 선언을 담고 있다.
- Reference Record (R)
  외부에서 LABEL을 받아오기 위한 선언을 담고 있다.
- Text Record (T)
  실제 명령어를 담고 있다. 각 line에는 시작 주소와 line의 길이를 담고 있다.
- Modification Record (M)
  Text Record를 수정하기 위한 정보를 담고 있는 Record이다. 수정할 위치 및 길이, LABEL 정보를 담고 있다.
- End Record (E)
  Section별로 끝임을 알려준다.

<예시 코드>

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

2.2 구현 아이디어



다음과 같은 구조로 설계한다. Visual Simulator가 GUI를 통해 사용자에게 명령을 받고 수행결과를 보여준다. SIC/XE Simulator는 GUI를 통해 들어온 명령을 실질적으로 수행하며 내부 로직의 중심을 담당한다. 그러기 위해서는 SIC/XE Loader를 통해 objectcode의 정보를 읽어온다. 그 정보를 여러 record의 정보를 종합하여 메모리에 올라갈 완전한 코드로 변환한다. 실제 objectcode를 Resource manger의 메모리에 할당한다. SIC/XE Simulator는 메모리에 load된 정보들을 가지고 simulation한다. Instruction launcher를 활용하여 simulation에서 해석한 명령어에 대한 동작을 수행한다.

3장 수행 결과

3.1 전체 시스템 구현 내용

output_objectcode.txt 파일로 objectcode를 입력받아 simulation한다. 실행하고 있는 명령어를 보여주고 레지스터값을 업데이트하여 보여준다. oneStep 버튼과 allStep 버튼을 구분하여 실행한다.
이번 프로그램은 'F1' device에서 문자열을 읽어와서 '05' device에 쓰는 동작을 수행한다.

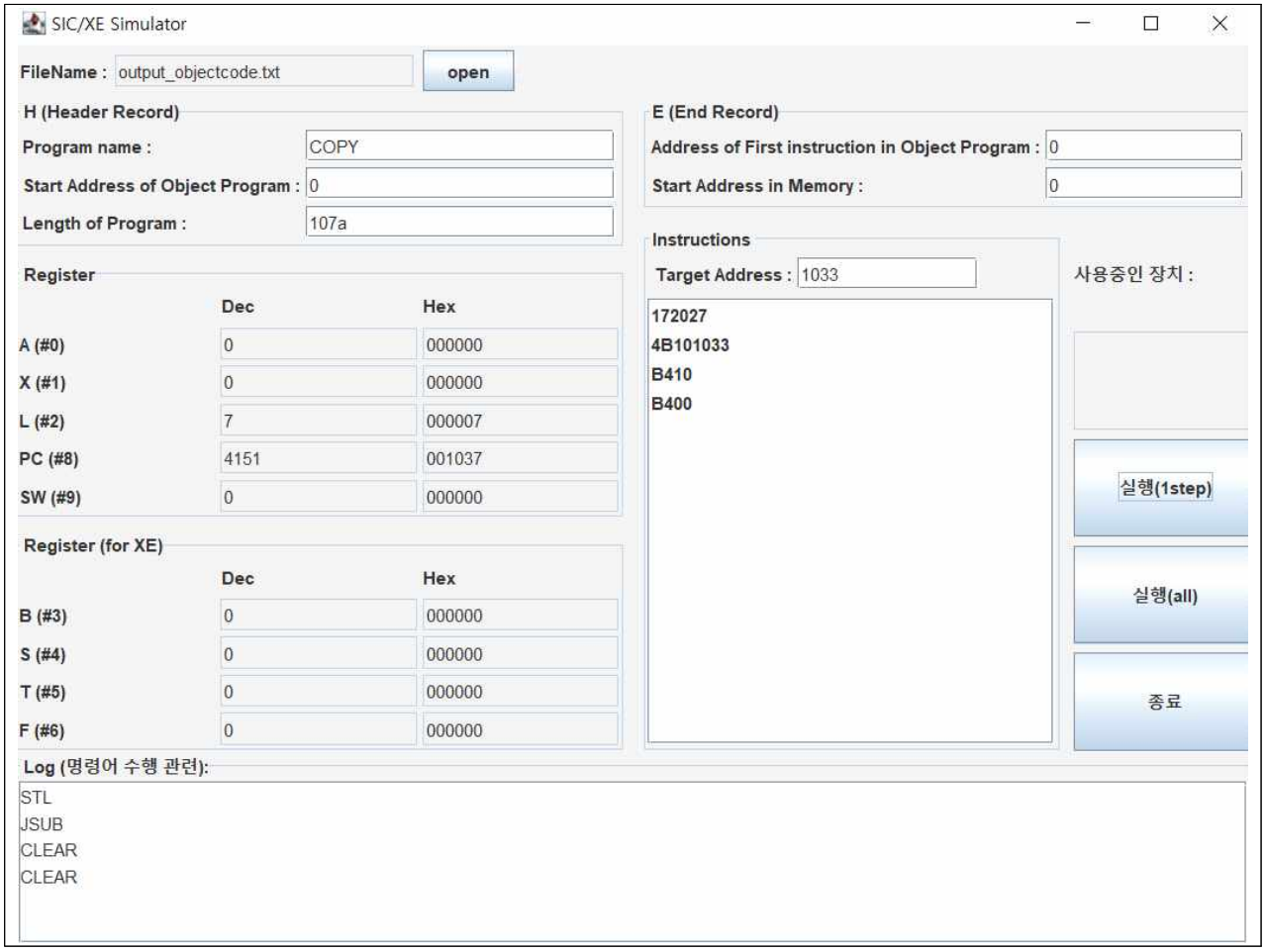
output_objectcode.txt

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002d
RRDREC WRREC
T0000001d1720274b1000000320232900003320074b1000003f2fec0320160f2016
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
```

실행 GUI 화면



F1.txt

```
ABCDEF
```

05.txt

```
ABCDEFE0F
```

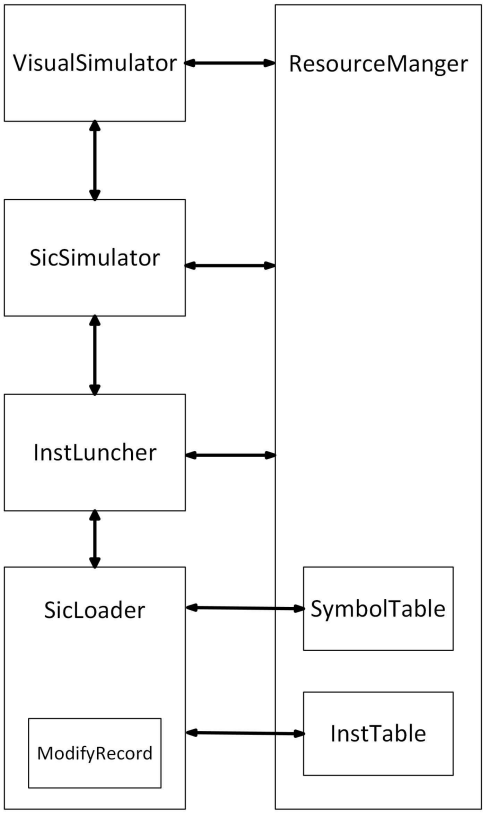
4장 결론 및 보충할 점

더 폭넓은 범위의 어셈블리 코드에 대응할 수 없고 개선이 필요하다.
control section 3개만을 처리할 수 있는 시뮬레이터이다. 또한 이번 프로젝트를 위한 명령어만 처리한다. 따라서 모든 SIC/XE 머신의 모든 명령어를 실행하지 못한다. 이번 프로젝트에서는 파일 read, write에 대해서만 test하였지만 더 다양한 코드에 대해서도 대응할 수 있도록 수정 및 구현해야 한다.

- 다양한 형식의 objectcode 지원
- 가변적 control section 개수에 대한 대응
- 모든 SIC/XE 머신 명령어 기능 구현
- 다양한 코드에 대한 처리 강화
- 예외 처리 강화 및 로직 최적화

5장 소스코드(+주석)

다음 그림은 Class의 대략적 구조도이다.



- **Class <VisualSimulator>**

VisualSimulator는 GUI를 담당하며 resourceManger와 SicSimulator를 활용하여 이벤트를 처리하고 화면으로 보여줍니다.

```
import javax.swing.*;
import javax.swing.border.TitledBorder;
import java.awt.*;
```

```java
import java.awt.event.*;
import java.io.File;
/**
 * VisualSimulator는 사용자와의 상호작용을 담당한다. 즉, 버튼 클릭등의 이벤트를 전달하고 그에
따른 결과값을 화면에 업데이트
 * 하는 역할을 수행한다.
 *
 * 실제적인 작업은 SicSimulator에서 수행하도록 구현한다.
 */
public class VisualSimulator extends JFrame {
    ResourceManager rMgr =new ResourceManager();
    //SicLoader       sicLoader       = new SicLoader(resourceManager);
    SicSimulator    sicSimulator    =new SicSimulator(rMgr);
    // --- UI components ---
    private JTextField txtFileName  =new JTextField(20);
    private JButton    btnOpen      =new JButton("open");
    private JTextField txtHName     =new JTextField(12);
    private JTextField txtHStart    =new JTextField(12);
    private JTextField txtHLength   =new JTextField(12);
    private JTextField txtEFirst    =new JTextField(12);
    private JTextField txtEStartMem =new JTextField(12);
    private String[]     basicNames  = {"A (#0)","X (#1)","L (#2)","PC (#8)","SW (#9)"};
    private JTextField[] regBasicDec =new JTextField[basicNames.length];
    private JTextField[] regBasicHex =new JTextField[basicNames.length];
    private String[]     xeNames     = {"B (#3)","S (#4)","T (#5)","F (#6)"};
    private JTextField[] regXEDec    =new JTextField[xeNames.length];
    private JTextField[] regXEHex    =new JTextField[xeNames.length];
    private JTextField           txtTarget =new JTextField(12);
    private DefaultListModel<String> instModel =new DefaultListModel<>();
    private JList<String>        lstInst   =new JList<>(instModel);
    private JTextField txtDevice    =new JTextField(12);
    private JButton    btnStep1     =new JButton("실행(1step)");
    private JButton    btnRunAll    =new JButton("실행(all)");
    private JButton    btnExit      =new JButton("종료");
    private JTextArea  txtLog       =new JTextArea(6, 60);
    public VisualSimulator() {
        super("SIC/XE Simulator");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // --- Top: File chooser panel ---
        JPanel filePanel =new JPanel(new FlowLayout(FlowLayout.LEFT, 6, 4));
        filePanel.add(new JLabel("FileName :"));
        txtFileName.setEditable(false);
        filePanel.add(txtFileName);
        filePanel.add(btnOpen);
        add(filePanel, BorderLayout.NORTH);
        // --- Left column: H record & registers ---
        JPanel leftColumn =new JPanel();
        leftColumn.setLayout(new BoxLayout(leftColumn, BoxLayout.Y_AXIS));
        leftColumn.add(createHPanel());
        leftColumn.add(Box.createVerticalStrut(8));
        leftColumn.add(createBasicRegPanel());
        leftColumn.add(Box.createVerticalStrut(8));
        leftColumn.add(createXERegPanel());
        // --- Right column: E record & instructions/controls ---
        JPanel rightColumn =new JPanel();
        rightColumn.setLayout(new BoxLayout(rightColumn, BoxLayout.Y_AXIS));
        // E 패널 (위)
```

```
        rightColumn.add(createEPanel());
        rightColumn.add(Box.createVerticalStrut(10));
        // Instructions+Controls (아래)
        JPanel instCtrl =new JPanel(new BorderLayout(8,8));
        instCtrl.add(createInstPanel(),    BorderLayout.CENTER);
        instCtrl.add(createControlPanel(), BorderLayout.EAST);
        rightColumn.add(instCtrl);
        // 그리고 center 영역에 leftColumn 과 함께 붙이세요
        JPanel center =new JPanel(new GridLayout(1,2,10,10));
        center.add(leftColumn);
        center.add(rightColumn);
        add(center, BorderLayout.CENTER);
        // --- Bottom: Log panel ---
        JPanel logPanel =new JPanel(new BorderLayout());
        logPanel.setBorder(new TitledBorder("Log (명령어 수행 관련):"));
        txtLog.setEditable(false);
        logPanel.add(new JScrollPane(txtLog), BorderLayout.CENTER);
        add(logPanel, BorderLayout.SOUTH);
        // --- Finalize window ---
        pack();
        setLocationRelativeTo(null);
        setVisible(true);

        // --- Event handlers ---

        btnOpen.addActionListener(e -> {
            JFileChooser chooser =new JFileChooser();
            if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
                File f = chooser.getSelectedFile();
                txtFileName.setText(f.getName());
                load(f);
            }
        });

        btnStep1.addActionListener(e -> { oneStep(); });
        btnRunAll.addActionListener(e -> { allStep(); });
        btnExit.addActionListener(e -> dispose());


    }
    private JPanel createHPanel() {
        JPanel p =new JPanel(new GridBagLayout());
        p.setBorder(new TitledBorder("H (Header Record)"));
        GridBagConstraints c =new GridBagConstraints();
        c.insets =new Insets(2,2,2,2); c.anchor = GridBagConstraints.WEST;
        // Program name
        c.gridx=0; c.gridy=0; p.add(new JLabel("Program name :"), c);
        c.gridx=1; c.fill=GridBagConstraints.HORIZONTAL; c.weightx=1; p.add(txtHName, c);
        // Start Address
        c.gridy=1;    c.gridx=0;    c.fill=GridBagConstraints.NONE;    c.weightx=0;    p.add(new
JLabel("Start Address of Object Program :"), c);
        c.gridx=1; c.fill=GridBagConstraints.HORIZONTAL; c.weightx=1; p.add(txtHStart, c);
        // Length
        c.gridy=2;    c.gridx=0;    c.fill=GridBagConstraints.NONE;    c.weightx=0;    p.add(new
JLabel("Length of Program :"), c);
        c.gridx=1; c.fill=GridBagConstraints.HORIZONTAL; c.weightx=1; p.add(txtHLength, c);
        return p;
```

```
    }
    private JPanel createEPanel() {
        JPanel p =new JPanel(new GridBagLayout());
        p.setBorder(BorderFactory.createTitledBorder("E (End Record)"));
        GridBagConstraints c =new GridBagConstraints();
        c.insets =new Insets(2,2,2,2);
        c.anchor = GridBagConstraints.WEST;
        // 1행: Address of First Instruction
        c.gridx =0; c.gridy =0;
        p.add(new JLabel("Address of First instruction in Object Program :"), c);
        c.gridx =1; c.fill = GridBagConstraints.HORIZONTAL; c.weightx =1;
        p.add(txtEFirst, c);
        // 2행: Start Address in Memory
        c.gridx =0; c.gridy =1; c.fill = GridBagConstraints.NONE; c.weightx =0;
        p.add(new JLabel("Start Address in Memory :"), c);
        c.gridx =1; c.fill = GridBagConstraints.HORIZONTAL; c.weightx =1;
        p.add(txtEStartMem, c);
        Dimension pref = p.getPreferredSize();
        p.setMaximumSize(new Dimension(pref.width, pref.height));
        return p;
    }
    private JPanel createBasicRegPanel() {
        JPanel p =new JPanel(new GridLayout(basicNames.length+1,3,4,4));
        p.setBorder(new TitledBorder("Register"));
        p.add(new JLabel("")); p.add(new JLabel("Dec")); p.add(new JLabel("Hex"));
        for (int i =0; i < basicNames.length; i++) {
            p.add(new JLabel(basicNames[i]));
            regBasicDec[i]     =new     JTextField(6);     regBasicDec[i].setEditable(false);
p.add(regBasicDec[i]);
            regBasicHex[i]     =new     JTextField(6);     regBasicHex[i].setEditable(false);
p.add(regBasicHex[i]);
        }
        return p;
    }
    private JPanel createXERegPanel() {
        JPanel p =new JPanel(new GridLayout(xeNames.length+1,3,4,4));
        p.setBorder(new TitledBorder("Register (for XE)"));
        p.add(new JLabel("")); p.add(new JLabel("Dec")); p.add(new JLabel("Hex"));
        for (int i =0; i < xeNames.length; i++) {
            p.add(new JLabel(xeNames[i]));
            regXEDec[i]       =new       JTextField(6);       regXEDec[i].setEditable(false);
p.add(regXEDec[i]);
            regXEHex[i]       =new       JTextField(6);       regXEHex[i].setEditable(false);
p.add(regXEHex[i]);
        }
        return p;
    }
    private JPanel createInstPanel() {
        JPanel p =new JPanel(new BorderLayout(4,4));
        p.setBorder(new TitledBorder("Instructions"));
        JPanel top =new JPanel(new FlowLayout(FlowLayout.LEFT,4,2));
        top.add(new JLabel("Target Address :")); top.add(txtTarget);
        p.add(top, BorderLayout.NORTH);
        p.add(new JScrollPane(lstInst), BorderLayout.CENTER);
        return p;
    }
    private JPanel createControlPanel() {
```

```
        JPanel p =new JPanel(new GridLayout(5,1,6,6));
        p.add(new JLabel("사용중인 장치 :")); txtDevice.setEditable(false); p.add(txtDevice);
        p.add(btnStep1); p.add(btnRunAll); p.add(btnExit);
        btnStep1.setEnabled(false); btnRunAll.setEnabled(false);
        return p;
    }
    /**
     * 프로그램 로드 명령을 전달한다.
     */
    public void load(File program) {
        rMgr.initializeResource();
        sicSimulator.load(program);
        instModel.clear();
        btnStep1.setEnabled(true);
        btnRunAll.setEnabled(true);

        //header
        txtHName.setText( sicSimulator.loader.getProgramName() );
        txtHStart.setText( Integer.toHexString(sicSimulator.loader.getStartAddress()) );
        txtHLength.setText( sicSimulator.loader.getProgramLength() );

        //end box
        txtEStartMem.setText(Integer.toHexString(sicSimulator.loader.getStartAddress()));
        txtEFirst.setText(Integer.toHexString(sicSimulator.loader.getStartAddress()));

        oneStep();

    }
    /**
     * 하나의 명령어만 수행할 것을 SicSimulator에 요청한다.
     */
    public void oneStep() {
        sicSimulator.oneStep();
        update();//미리 pc값 업데이트 및 실행
    }
    /**
     * 남아있는 모든 명령어를 수행할 것을 SicSimulator에 요청한다.
     */
    public void allStep() {
        Timer t =new Timer(10, null);//timer를 통해 10ms 단위로 명령어 실행과정 및 gui 업데이
트

        t.addActionListener(ev -> {
            if(rMgr.getRegister(8) ==0) {
                t.stop();
            }else {
                oneStep();
            }

        });
        t.setInitialDelay(0);
        t.start();
    }
    /**
     * 화면을 최신값으로 갱신하는 역할을 수행한다.
     */
    public void update() {
        // register & log 업데이트 로직
```

```
        instModel.addElement(sicSimulator.inst_field);
        lstInst.ensureIndexIsVisible(instModel.getSize() -1);
        //target address
        txtTarget.setText(Integer.toHexString(sicSimulator.target_address));

        for (int i =0; i < basicNames.length; i++) {
            int regIdx =switch(i) {
                case 0 ->0;   // A
                case 1 ->1;   // X
                case 2 ->2;   // L
                case 3 ->8;   // PC
                case 4 ->9;   // SW
                default ->0;
            };
            int dec = rMgr.getRegister(regIdx);
            regBasicDec[i].setText(String.valueOf(dec));
            regBasicHex[i].setText(String.format("%06X", dec));

        }
        // XE 레지스터(B,S,T,F)
        for (int i =0; i < xeNames.length; i++) {
            int regIdx =switch(i) {
                case 0 ->3;  // B
                case 1 ->4;  // S
                case 2 ->5;  // T
                case 3 ->6;  // F
                default ->0;
            };
            int dec = rMgr.getRegister(regIdx);
            regXEDec[i].setText(String.valueOf(dec));
            regXEHex[i].setText(String.format("%06X", dec));
        }
        txtLog.append(sicSimulator.logs.get(0));
        txtLog.append("\n");
        sicSimulator.logs.clear();
    }
    public static void main(String[] args) {
        EventQueue.invokeLater(VisualSimulator::new);
    }
}
```

- **Class <SicSimulator>**

VisualSimulator에서 받은 이벤트를 처리하고 simulation의 메인 로직을 담당한다.

```
import java.io.File;
import java.util.ArrayList;

public class SicSimulator {
    ResourceManager rMgr;
    SicLoader loader;

    int target_address;
    String inst_field;//Log 기록용

    public ArrayList<String> logs =new ArrayList<>();
    InstLuncher instluncher;
```

```
    public SicSimulator(ResourceManager resourceManager) {
        // 필요하다면 초기화 과정 추가
        this.rMgr = resourceManager;
        this.loader =new SicLoader(resourceManager);
        this.instluncher =new InstLuncher(resourceManager);
    }
    /**
     * 레지스터, 메모리 초기화 등 프로그램 load와 관련된 작업 수행. 단, object code의 메모리
적재 및 해석은
     * SicLoader에서 수행하도록 한다.
     */
    public void load(File program) {
        /* 메모리 초기화, 레지스터 초기화 등 */
        rMgr.initializeResource();
        loader.load(program);
        target_address =0;
    }

    public String charToString(char[] a) {
        StringBuilder sb =new StringBuilder();
        for (char b : a) {
            sb.append(String.format("%02X", (int)b &0xFF));
        }
        String hexInstr = sb.toString();
        return hexInstr;
    }
    /**
     * 1개의 instruction이 수행된 모습을 보인다.
     */
    public void oneStep() {
        int operand =0;
        int pc = rMgr.getRegister(8);

        char[] a = rMgr.getMemory(pc,3);

        String hexInstr = charToString(a);//hex instruction 변수

        int inst = Integer.parseInt(hexInstr,16);//string -> int
        int nixbpe = (inst)>>12 &0x3f;

        int op_format2 = (inst)>>16;//op 8bit(6bit + 2bit) format2 처리

        int op = ((inst)>>16) &0xFC;//1111_1100
        int value=0;

        boolean isImediate =false;

        if(rMgr.inst.getFormat(op_format2) ==2) {//format 2
            pc+=2;
            rMgr.setRegister(8, pc);
            inst_field = hexInstr.substring(0,4);
            isImediate =false;
            Instruction temp_inst = rMgr.inst.getInstruction(op_format2);
            inst = inst>>8;

            int reg1, reg2;
```

```
            reg1 = (inst>>4) &0x0f;
            reg2 = inst &0x0f;

            exec(temp_inst.mnemonic,0,reg1,reg2,pc,isImediate);
            return;
        }//////////format 4, 3 처리
        else if((nixbpe &1) ==1) {//format 4
            char[] a2 = rMgr.getMemory(pc+3,1);
            hexInstr += charToString(a2);
            inst = Integer.parseInt(hexInstr,16);
            inst_field = hexInstr;
            pc+=4;
            rMgr.setRegister(8, pc);
            operand = inst &0xFFFFF;// 3byte opeanrd

            if((nixbpe &0x30) ==16) {//immediate
                value = operand;
                isImediate =true;
            }else {
                if((nixbpe &0x02) ==2) {//pc relative
                    target_address = pc + operand;
                }else {
                    target_address = operand;
                }

                value = target_address;
            }

            if( (nixbpe &8) ==8 ) {//x bit 확인
                value += rMgr.getRegister(1);
            }

            Instruction temp_inst2 = rMgr.inst.getInstruction(op);
            exec(rMgr.inst.getInstruction(op).mnemonic,value,0,0,pc,isImediate);

        }else if(rMgr.inst.getFormat(op) ==3){//format 3
            pc+=3;
            rMgr.setRegister(8, pc);
            inst_field = hexInstr;
            operand = inst &0xFFF;//3byte operand

            if ((operand  &0x800) !=0) {
                operand |= ~0xFFF;//signed extension
            }
            if((nixbpe &0x30) ==16) {//immediate
                value = operand;
                isImediate =true;
            }else if((nixbpe &0x30) ==32) {
                if((nixbpe &0x02) ==2) {//pc relative
                    target_address = pc + operand;
                }else {
                    target_address = operand;
                }
                value = target_address;
                value = Integer.parseInt(charToString(rMgr.getMemory(target_address,3)),16);
            }
```

```
            else {
                if((nixbpe &0x02) ==2) {//pc relative
                    target_address = pc + operand;
                }else {
                    target_address = operand;
                }
                value = target_address;
            }

            Instruction temp_inst2 = rMgr.inst.getInstruction(op);
            exec(rMgr.inst.getInstruction(op).mnemonic,value,0,0,pc,isImediate);

        }else {
            System.err.println("wrong op");
        }
    }

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

    //instLuncher 실행전 분배
    public void exec(String op, int addr, int reg1, int reg2,int pc,boolean isImediate) {
        addLog(op);

        if (op.equals("STL")) {
            instluncher.STL(addr);
        }
        else if (op.equals("JSUB")) {
            instluncher.JSUB(addr,pc);
        }
        else if (op.equals("LDA")) {
            instluncher.LDA(addr, isImediate);
        }
        else if (op.equals("COMP")) {
            instluncher.COMP(addr,isImediate);
        }
        else if (op.equals("COMPR")) {
            instluncher.COMPR(reg1, reg2);
        }
        else if (op.equals("JEQ")) {
            instluncher.JEQ(addr);
        }
        else if (op.equals("J")) {
            instluncher.J(addr);
        }
        else if (op.equals("STA")) {
            instluncher.STA(addr);
        }
        else if (op.equals("CLEAR")) {
            instluncher.CLEAR(reg1); // here value is actually a register number
        }
        else if (op.equals("LDT")) {
            instluncher.LDT(addr,isImediate);// value holds the constant or address
```

```
        }
        else if (op.equals("TD")) {
            instluncher.TD(addr);
        }
        else if (op.equals("RD")) {
            instluncher.RD(addr);
        }
        else if (op.equals("WD")) {
            instluncher.WD(addr);
        }
        else if (op.equals("STCH")) {
            instluncher.STCH(addr);
        }
        else if (op.equals("TIXR")) {
            instluncher.TIXR();
        }
        else if (op.equals("JLT")) {
            instluncher.JLT(addr);
        }
        else if (op.equals("STX")) {
            instluncher.STX(addr);
        }
        else if (op.equals("RSUB")) {
            instluncher.RSUB();
        }else if(op.equals("LDCH")) {

            instluncher.LDCH(addr,isImediate);
        }else {
            throw new IllegalArgumentException("Unknown instruction: "+ op);
        }
    }
}
```

- **Class <SicLoader>**

SicLoader는 objectcode를 SymbolTable, InstTable을 활용하여 코드를 해석하며 modify한 다음 memory에 load하는 역할을 한다. (linker and loader)

```
import java.io.File;
import java.util.stream.Collectors;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

public class SicLoader {
    ResourceManager rMgr;

    String programName;

    int startAddress[] =new int[3];//함수 3개
    int programLength[] =  new int[3];

    int h_flag =0;//h section number
    int h_addr=0;//시작 주소 나머지 함수들 보정용
    int whole_length=0;//program 전체 길이
```

```
    //for text variables/////////////////
    ArrayList<String> text =new ArrayList<>();//text 저장용
    ArrayList<Integer> lineAddrs =new ArrayList<>();//text별 address 저장용
    ArrayList<Integer> text_secs =new ArrayList<>();//text별 section 저장용


    ////////////////////////////////

    ArrayList<ModifyRecord> modifyRecords =new ArrayList<>();

    public SicLoader(ResourceManager resourceManager) {
        setResourceManager(resourceManager);
    }
    /**
     * Loader와 프로그램을 적재할 메모리를 연결시킨다.
     *
     * @param rMgr
     */
    public void setResourceManager(ResourceManager resourceManager) {
        this.rMgr = resourceManager;
        this.rMgr.symtabList=new SymbolTable();
    }
    /**
     * 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
     */
    public void load(File objectCode) {
        try (BufferedReader br =new BufferedReader(new FileReader(objectCode))) {
            String line;
            while ((line = br.readLine()) !=null) {
                if (line.isEmpty()) continue;
                //System.out.println(line);
                char type = line.charAt(0);
                if(type =='H') {
                    h_flag++;
                    parseHeader(line);
                }
                if(type =='T') {
                    parseText(line);
                }else if(type =='D') {
                    parseDefine(line);
                }else if(type =='M') {
                    parseModify(line);
                }

            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        //modify records를 반영하여 text 수정
        modify_text();

        //load memory
```

```
        for (int i =0; i < text.size(); i++) {
            String instrHex = text.get(i);      // ex: "4B110033"
            int    startAddr = lineAddrs.get(i); // ex: 0x0010
            rMgr.loadHex(startAddr, instrHex);
        }

    }
    void parseHeader(String line) {
        if(h_flag ==1) {
            programName   = line.substring(1,7).trim();
        }
        startAddress[h_flag-1]  = Integer.parseInt(line.substring(7, 13)) + h_addr;
        programLength[h_flag-1] = Integer.parseInt(line.substring(13,19),16);
        whole_length += programLength[h_flag-1];
        String sym = line.substring(1,  7).trim();
        rMgr.symtabList.putSymbol(sym, h_addr,h_flag);//copy RDREC WRREC labael symtable에 저
장

        h_addr += programLength[h_flag-1];
    }
    void parseDefine(String line) {
        int idx =1;
        while (idx +12 <= line.length()) {
            String sym = line.substring(idx, idx +6).trim();
            int    addr=   Integer.parseInt(line.substring(idx   +6,   idx   +12),   16)   +
startAddress[h_flag];
            rMgr.symtabList.putSymbol(sym, addr,h_flag);
            idx +=12;
        }
    }
    void parseText(String line) {

        int idx =9;// 1 + 6 + 2;
        int addr = Integer.parseInt(line.substring(5,7),16) + startAddress[h_flag-1];
        while (idx+2 <  line.length()) {
            String temp =new String();
            int first = Integer.parseInt(line.substring(idx, idx+2),16);
            int op = first & (0xFC);//1111_1100

            int slice = Integer.parseInt(line.substring(idx+1, idx+3),16);
            int nixbpe = slice &0x3f;
            if(rMgr.inst.getFormat(op) ==2) {//format 2
                temp = line.substring(idx, idx +4).trim();
                idx +=4;
                lineAddrs.add(addr);
                addr +=2;
            }else if((nixbpe &1) ==1) {//format 4
                temp = line.substring(idx, idx +8).trim();
                idx +=8;
                lineAddrs.add(addr);
                addr +=4;
            }else if(rMgr.inst.getFormat(op) ==3){ // format 3
                temp = line.substring(idx, idx +6).trim();
                idx +=6;
                lineAddrs.add(addr);
                addr +=3;
            }else if(first ==0xF1) {//f1 일때 이번 프로젝트에서만 입력을 f1에서만 받는다고 가
```

```
정 나중에 다시 처리하기
                text.add("F1");
                lineAddrs.add(addr++);
                text_secs.add(h_flag);
                temp = line.substring(idx+2, idx +8).trim();
                text.add(temp);
                lineAddrs.add(addr);
                text_secs.add(h_flag);
                addr+=3;
                break;
            }
            text_secs.add(h_flag);
            text.add(temp);
        }
        if(idx +2 == line.length()) {//남은 text 처리
            text.add(line.substring(idx, idx+2));
            lineAddrs.add(addr);
        }
    }

    void parseModify(String line) {

        int addr      = Integer.parseInt(line.substring(1, 7), 16);
        int length = Integer.parseInt(line.substring(7, 9), 16);
        char sign = line.charAt(9); // + or - (필요하면 따로 보관)
        String sym    = line.substring(10).trim();
        modifyRecords.add(new ModifyRecord(addr, length, sym,sign,h_flag));
    }

    /*text modify해주는 함수 */
    void modify_text() {

        for(ModifyRecord m : modifyRecords) {

            int address_real = rMgr.symtabList.search(m.symbol.trim());

            int index =-1;//text list 접근 index 찾기

            //수정할 길이별 처리
            if(m.length ==5) {
                index = lineAddrs.indexOf(m.address -1 + startAddress[m.sec-1]);
            }else if(m.length ==6) {
                index = lineAddrs.indexOf(m.address+ startAddress[m.sec-1]);
            }

            if(index ==-1) {
                System.out.printf("%x\n",m.address);
                continue;
            }

            String oldStr = text.get(index);
            String newStr;//modify string

            if(m.length ==5) {

                if(m.sign =='+') {
                    newStr  =  Integer.toHexString((Integer.parseInt(oldStr.substring(3),16)  +
```

```
address_real));
                }else {
                    newStr  =  Integer.toHexString((Integer.parseInt(oldStr.substring(3),16)  -
address_real));
                }

                if(newStr.length() ==4){
                    text.set(index, (oldStr.substring(0,4)  + newStr));
                }else if(newStr.length() ==3) {
                    text.set(index, (oldStr.substring(0,5)  + newStr));
                }
                else if(newStr.length() ==2) {
                    text.set(index, (oldStr.substring(0,6)  + newStr));
                }

            }else if(m.length ==6) {

                if(m.sign =='+') {
                    newStr      =      Integer.toHexString((Integer.parseInt(oldStr,16)      +
address_real));
                }else {
                    newStr      =      Integer.toHexString((Integer.parseInt(oldStr,16)      -
address_real));
                }

                if(newStr.length() ==4) {
                    text.set(index, "00"+newStr);
                }else {
                    text.set(index, newStr);
                }

            }

        }
    }

    /*get 함수들*/
    public String getProgramName() {
        return this.programName;
    }
    public int getStartAddress() {
        return this.startAddress[0];
    }
    public String getProgramLength() {
        return Integer.toHexString(this.whole_length);
    }

}
```

- **Class <ModifyRecord>**

SicLoader에서 사용되는 클래스로 modifyRecord를 parsing하여 text정보를 수정할 때 사용된다.

```
public class ModifyRecord {
      public final int    address, length;
      public final String symbol;
      public char sign;
```

```
      public int sec;

      public ModifyRecord(int address,int length,String symbol, char sign, int sec){
        this.address=address; this.length =length; this.symbol=symbol; this.sign = sign;
        this.sec = sec;
      }
      public void modi_print() {
          System.out.println(address +" "+length +" "+symbol+" "+sign+" "+sec );
      }
}
```

- **Class <SymbolTable>**

SymbolTable을 관리하기 위한 변수와 method가 선언되어있다.

```
import java.util.ArrayList;
/**
 * symbol과 관련된 데이터와 연산을 소유한다. section 별로 하나씩 인스턴스를 할당한다.
 */
public class SymbolTable {
    ArrayList<String> symbolList;
    ArrayList<Integer> addressList;
    ArrayList<Integer> secList;
    // 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 SymbolTable() {
        this.symbolList =new ArrayList<>();
        this.addressList =new ArrayList<>();
        this.secList =new ArrayList<>();
    }

    public void putSymbol(String symbol, int address,int sec) {
        symbolList.add(symbol);
        addressList.add(address);
        secList.add(sec);
    }
    /**
     * 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) {
        for(String s : symbolList) {
            if(s.equals(symbol)) {
                addressList.set(symbolList.indexOf(symbol), 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 =-1;

        for(String s : symbolList) {
            if(s.equals(symbol)) {
                return addressList.get(symbolList.indexOf(symbol));
            }
        }
        return address;
    }

    public ArrayList<String> getSymList(){return this.symbolList;}
    public ArrayList<Integer> getAddrList(){return this.addressList;}
}
```

- **Class <InstTable>**

InstTable은 Instruction을 관리하는 클래스이다. input.txt를 입력받아 명령어 별 format, opcode 등을 저장하고 이번 프로젝트에 맞게 op(hex)를 입력받으면 여러 정보를 return할 수 있게 하였다.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
/**
 * 모든 instruction의 정보를 관리하는 클래스.
 * Assembler에서 사용한 클래스 변형
 */
public class InstTable {

    /** opcode(정수) → Instruction 정보 */
    private final Map<Integer,Instruction> instMap;
    /**
     * 파싱과 동시에 instMap을 초기화
     * @param instFile : 명세 파일 경로
     */
    public InstTable(String instFile) {
        instMap =new HashMap<>();
        openFile(instFile);
    }
    /**
     * 파일을 열어 한 줄씩 파싱하고 instMap에 저장
     */
    private void openFile(String fileName) {
        try (BufferedReader br =new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = br.readLine()) !=null) {
```

```
                Instruction inst =new Instruction(line);
                instMap.put(inst.opcode, inst);
//              System.out.printf("Loaded opcode 0x%02X → %s%n",inst.opcode, inst.mnemonic);
            }
        } catch (IOException e) {
            throw new RuntimeException("Error reading inst file “"+ fileName +"”: "+
e.getMessage(), e);
        }
    }
    /** opcode로 Instruction 객체를 꺼냅니다. */
    public Instruction getInstruction(int opcode) {
        return instMap.get(opcode);
    }

    /** opcode로부터 format return */
    public int getFormat(int opcode) {
        Instruction inst = instMap.get(opcode);
        if (inst ==null) return -1;
        return inst.format;
    }
}
/**
 * 명령어 하나하나의 구체적인 정보는 Instruction클래스에 담깁니다.
 */
class Instruction {
    public String  mnemonic;     // 예: "LDA"
    public int     format;       // 1,2,3,4
    public int     opcode;       // 예: 0x00
    public int     operandCount; // 예: 2
    /**
     * "LDA 3 00 2" 같은 한 줄을 파싱
     * parts[0]=mnemonic, parts[1]=format, parts[2]=opcode(16진수), parts[3]=operandCount
     */
    public Instruction(String line) {
        String[] parts = line.trim().split("\\s+");
        if (parts.length <4) {
            throw new IllegalArgumentException("Invalid instruction entry: "+ line);
        }
        mnemonic     = parts[0];
        format       = Integer.parseInt(parts[1]);
        opcode       = Integer.parseInt(parts[2], 16);
        operandCount = Integer.parseInt(parts[3]);
    }
}
```

- **Class <InstLuncher>**

InstLuncher 클래스는 Simulator에서 해석한 실제 명령어의 동작의 정의되어있는 클래스이다. LDA, LDT 등의 operator들의 동작이 정의되어있다.

```
// instruction에 따라 동작을 수행하는 메소드를 정의하는 클래스
public class InstLuncher {
    ResourceManager rMgr;
    public InstLuncher(ResourceManager resourceManager) {
        this.rMgr = resourceManager;
    }
```

```
    public String charToString(char[] a) {
        StringBuilder sb =new StringBuilder();
        for (char b : a) {
            sb.append(String.format("%02X", (int)b &0xFF));
        }
        String hexInstr = sb.toString();
        return hexInstr;
    }

    // instruction 별로 동작을 수행하는 메소드를 정의
    // ex) public void add(){...}
    public void STL(int addr) {
        char[] bytes = rMgr.intToChar(rMgr.getRegister(2));
        rMgr.setMemory(addr,bytes,bytes.length);
    }

    public void JSUB(int addr,int addr_return) {
        rMgr.setRegister(8, addr);//pc 값 업데이트
        rMgr.setRegister(2, addr_return);
    }

    public void LDA(int addr,boolean isImediate) {
        int          value          =          (isImediate)          ?          addr          :
Integer.parseInt(charToString(rMgr.getMemory(addr,3)),16);
        rMgr.setRegister(0, value);
        //System.out.printf("%x %x\n",value, addr);
    }

    public void COMP(int addr,boolean isImediate) {
        int          value          =          (isImediate)          ?          addr          :
Integer.parseInt(charToString(rMgr.getMemory(addr,3)),16);

        if( (rMgr.getRegister(0) == value) ) {//eqaul
            rMgr.setRegister(9, 1);
        }else if(rMgr.register[0] > value) {//GT
            rMgr.setRegister(9, 2);
        }
        else if(rMgr.register[0] < value) {//LT
            rMgr.setRegister(9, 3);
        }
        else {
            rMgr.setRegister(9, 0);
        }
    }

    public void JEQ(int addr) {
        if(rMgr.getRegister(9) ==1) {
            rMgr.setRegister(8, addr);//pc=8
        }
    }
    public void J(int addr) {
        rMgr.setRegister(8, addr);
    }

    public void STA(int addr) {
        char[] bytes = rMgr.intToChar(rMgr.getRegister(0));
        rMgr.setMemory(addr,bytes,3);
```

```
    }

    public void CLEAR(int reg) {
        rMgr.setRegister(reg, 0);
    }

    public void LDT(int addr, boolean isImediate) {
        int          value          =          (isImediate)          ?          addr          :
Integer.parseInt(charToString(rMgr.getMemory(addr,3)),16);
        rMgr.setRegister(5, value);
    }

    public void COMPR(int reg1, int reg2) {
        if( ( rMgr.getRegister(reg1) == rMgr.getRegister(reg2) ) ) {//eqaul
             rMgr.setRegister(9, 1);
        }else if( ( rMgr.getRegister(reg1) > rMgr.getRegister(reg2) ) ) {//GT
            rMgr.setRegister(9, 2);
        }else if( ( rMgr.getRegister(reg1) < rMgr.getRegister(reg2) ) ) {//LT
            rMgr.setRegister(9, 3);
        }
        else {
            rMgr.setRegister(9, 0);
        }
    }

    public void TD(int addr) {
        String dev = charToString(rMgr.getMemory(addr,1));
        dev+=".txt";
        rMgr.testDevice(dev);
    }

    public void RD(int addr) {
        String dev = charToString(rMgr.getMemory(addr,1));
        dev+=".txt";
        // 1 바이트 읽기
        char[] buf = rMgr.readDevice(dev, 1);
        // unsigned -> int로 바꿔서 A 레지스터에 저장
        int value = buf.length >0 ? (buf[0] &0xFF) : 0;
        rMgr.setRegister(0, value);
    }

    public void WD(int addr) {
         String dev = charToString(rMgr.getMemory(addr,1));
         dev+=".txt";
         int    aVal = rMgr.getRegister(0) &0xFF;
         char[] out  =new char[]{ (char)aVal };
         rMgr.writeDevice(dev, out, 1);
    }

    public void STCH(int addr) {
        int value = rMgr.getRegister(0) &0xFF;//a register의 하위 8bit만
        char[] temp =new char[] {(char) value};
        rMgr.setMemory(addr, temp, 1);
    }

    public void TIXR() {//x register 증가
        rMgr.register[1]++;
```

```
        if(rMgr.getRegister(1) < rMgr.getRegister(5)) {//LT
            rMgr.setRegister(9, 3);
        }else if(rMgr.getRegister(1) > rMgr.getRegister(5)) {
            rMgr.setRegister(9, 2);
        }else {
            rMgr.setRegister(9, 1);
        }
    }

    public void JLT(int addr) {
        if(rMgr.getRegister(9) ==3) {
            rMgr.setRegister(8, addr);//pc <- addr
        }
    }

    public void STX(int addr) {
        char[] bytes = rMgr.intToChar(rMgr.getRegister(1));
        rMgr.setMemory(addr,bytes,bytes.length);//memory<-x
    }

    public void RSUB() {
        rMgr.setRegister(8, rMgr.getRegister(2));//L register => pc
    }

    public void LDCH(int addr,boolean isImediate) {
        int value = (isImediate) ? addr :
Integer.parseInt(charToString(rMgr.getMemory(addr,1)),16);
        rMgr.setRegister(0, value);
    }
}
```

- **Class <ResourceManger>**

ResourceManger 클래스는 이번 프로젝트의 모든 자원을 관리한다. 대표적으로 register, symbol, insturction, memory에 대한 정보를 가지며 다른 클래스들에서 사용된다.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.RandomAccessFile;
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 {

    HashMap<String, Object> deviceManager =new HashMap<String, Object>();
    char[] memory =new char[65536]; // String으로 수정해서 사용하여도 무방함.
    int[] register =new int[10];//register number정의에 따름
    double register_F;

    SymbolTable symtabList;
    InstTable inst =new InstTable("inst_table.txt");
    // 이외에도 필요한 변수 선언해서 사용할 것.

    /**
     * 메모리, 레지스터등 가상 리소스들을 초기화한다.
     */
    public void initializeResource() {
        /*모두 0으로 초기화*/
        Arrays.fill(memory, (char)0);
        Arrays.fill(register, 0);
    }

    // memory load method
    public void loadHex(int addr, String hex) {
        int len = hex.length() /2;
        char[] data =new char[len];
        for (int i =0; i < len; i++) {
            // 2글자씩 잘라서 byte 값으로 파싱
            int b = Integer.parseInt(hex.substring(2 * i, 2 * i +2), 16);
            data[i] = (char) b;
        }
        setMemory(addr, data, len);
    }

     public String dumpMemory(int start, int length) {
        StringBuilder sb =new StringBuilder();
        int end = Math.min(start +length, memory.length);

        for (int addr = start; addr < end; addr +=16) {
            sb.append(String.format("%04X: ", addr));
            int rowLen = Math.min(16, end - addr);
            for (int i =0; i < rowLen; i++) {
                sb.append(String.format("%02X ", (int) memory[addr + i]));
            }
            sb.append(System.lineSeparator());
        }
        return sb.toString();
     }
     public void printMemory(int start, int length) {
         System.out.print(dumpMemory(start, length));
     }

    /**
     * deviceManager가 관리하고 있는 파일 입출력 stream들을 전부 종료시키는 역할. 프로그램을
종료하거나 연결을 끊을 때
     * 호출한다.
     */
```

```
    public void closeDevice() {
        for (Object o : deviceManager.values()) {
            if (o instanceof RandomAccessFile) {
                try {
                    ((RandomAccessFile) o).close();
                } catch (IOException ignored) {}
            }
        }
        deviceManager.clear();
    }
    /**
     * 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
     */
    public void testDevice(String devName) {
        register[9] =0;
        if (deviceManager.containsKey(devName)) return;
        try {
            // devName을 파일 이름으로 사용
            RandomAccessFile raf =new RandomAccessFile(devName, "rw");
            deviceManager.put(devName, raf);
            // device ready
        } catch (IOException e) {
            register[9] =1;
            throw new RuntimeException("TD: cannot open device "+ devName, e);
        }
    }
    /**
     * 디바이스로부터 원하는 개수만큼의 글자를 읽어들인다. RD명령어를 사용했을 때 호출되는 함수.
     *
     * @param devName 디바이스의 이름
     * @param num     가져오는 글자의 개수
     * @return 가져온 데이터
     */
    public char[] readDevice(String devName, int num) {
        testDevice(devName);
        RandomAccessFile raf = (RandomAccessFile) deviceManager.get(devName);
        char[] buf =new char[num];
        try {
            for (int i =0; i < num; i++) {
                int b = raf.read();
                if (b ==-1) {// EOF면 0 채우기
                    buf[i] =0;
                } else {
                    buf[i] = (char) b;
                }
            }

        } catch (IOException e) {
            throw new RuntimeException("RD: error reading from "+ devName, e);
        }
        return buf;
    }
```

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

        RandomAccessFile raf = (RandomAccessFile) deviceManager.get(devName);
        try {
            for (int i =0; i < num; i++) {
                raf.write((byte)data[i]);
                System.out.println(data[i]);
            }
        } catch (IOException e) {
            throw new RuntimeException("WD: error writing to "+ devName, e);
        }
    }
    /**
     * 메모리의 특정 위치에서 원하는 개수만큼의 글자를 가져온다.
     *
     * @param location 메모리 접근 위치 인덱스
     * @param num      데이터 개수
     * @return 가져오는 데이터
     */
    public char[] getMemory(int addr, int length) {
        char[] buf =new char[length];
        System.arraycopy(memory, addr, buf, 0, length);
        return buf;
    }
    /**
     * 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 len) {
        int i;
        for (i =0; i < len; i++) {
            memory[locate + i] = data[i];
        }
    }
    /**
     * 번호에 해당하는 레지스터가 현재 들고 있는 값을 리턴한다. 레지스터가 들고 있는 값은 문
자열이 아님에 주의한다.
     *
     * @param regNum 레지스터 분류번호
     * @return 레지스터가 소지한 값
     */
    public int getRegister(int regNum) {
        if(regNum >=10) {
            System.err.println("regNumm error");
            return -1;
```

```
        }
        return register[regNum];
    }
    /**
     * 번호에 해당하는 레지스터에 새로운 값을 입력한다. 레지스터가 들고 있는 값은 문자열이 아
님에 주의한다.
     *
     * @param regNum 레지스터의 분류번호
     * @param value  레지스터에 집어넣는 값
     */
    public void setRegister(int regNum, int value) {
        if(regNum >=10) { //reg num limit
            System.err.println("regNumm error");
            return;
        }
        register[regNum] = value;
    }
    /**
     * 주로 레지스터와 메모리간의 데이터 교환에서 사용된다. int값을 char[]형태로 변경한다.
     *
     * @param data
     * @return
     */
    public char[] intToChar(int data) {
        char[] result =new char[3];
        result[0] = (char)((data >>>16) &0xFF);
        result[1] = (char)((data >>>  8) &0xFF);
        result[2] = (char)( data         &0xFF);
        return result;
    }
    /**
     * 주로 레지스터와 메모리간의 데이터 교환에서 사용된다. char[]값을 int형태로 변경한다.
     *
     * @param data
     * @return
     */
    public int byteToInt(byte[] data) {
        int value =0;
        for (int i =0; i < data.length; i++) {
            value = (value <<8) | (data[i] &0xFF);
        }
        return value;
    }
}
```