System programming

Programming Assignment 1

410821316

鄧祺文 CHI-WEN TENG

**Assignment Problem**

Write an SIC assembler that reads an SIC assembly program, translates SIC statements into their machine code equivalents, and generates an object file.

**Highlight Program**

**Program Listing**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Scanner;

class Data2 {

private String first = "";

private String second = "";

private String third = "";

private String str = "";

public Data2(String first, String second, String third, String str) {

this.first = first;

this.second = second;

this.third = third;

this.str = str;

}

public String getFirst() {

return first;

}

public String getSecond() {

return second;

}

public String getThird() {

return third;

}

public String getStr() {

return str;

}

}

class Pair2 {

private String symbol = "";

private String location = "";

public Pair2(String symbol, String location) {

this.symbol = symbol;

this.location = location;

}

public String getSymbol() {

return symbol;

}

public String getLocation() {

return location;

}

}

public class SICAssembler {

public static void main(String[] args) throws IOException {

String[] op\_TAB = { "ADD", "ADDF", "ADDR", "AND", "CLEAR", "COMP", "COMPF", "COMPR", "DIV", "DIVF", "DIVR",

"FIX", "FLOAT", "HIO", "J", "JEQ", "JGT", "JLT", "JSUB", "LDA", "LDB", "LDCH", "LDF", "LDL", "LDS",

"LDT", "LDX", "LPS", "MUL", "MULF", "MULR", "NORM", "OR", "RD", "RMO", "RSUB", "SHIFTL", "SHIFTR",

"SIO", "SSK", "STA", "STB", "STCH", "STF", "STI", "STL", "STS", "STSW", "STT", "STX", "SUB", "SUBF",

"SUBR", "SVC", "TD", "TIO", "TIX", "TIXR", "WD" };

String[] opCode = { "18", "58", "90", "40", "B4", "28", "88", "A0", "24", "64", "9C", "C4", "C0", "F4", "3C",

"30", "34", "38", "48", "00", "68", "50", "70", "08", "6C", "74", "04", "E0", "20", "60", "98", "C8",

"44", "D8", "AC", "4C", "A4", "A8", "F0", "EC", "0C", "78", "54", "80", "D4", "14", "7C", "E8", "84",

"10", "1C", "5C", "94", "B0", "E0", "F8", "2C", "B8", "DC" };

ArrayList<Pair2> SYM\_TAB = new ArrayList<>();

ArrayList<String> Location = new ArrayList<>();

ArrayList<String> Target = new ArrayList<>();

ArrayList<Data2> Data = new ArrayList<>();

FileReader fr = new FileReader("src/main/resources/SIC.txt");

BufferedReader br = new BufferedReader(fr);

Scanner scn = new Scanner(br);

int n = 0;

int num = 0;

int DecLoc = 0;

int j;

int i;

String HexLoc;

String str1 = " ";

String str2 = " ";

String str3 = " ";

boolean isOpCode = false;

boolean isLine = false;

while (scn.hasNext()) {

String tempString = scn.next();

if (tempString.equals("START") || tempString.equals("END") ||

tempString.equals("WORD") || tempString.equals("BYTE") ||

tempString.equals("RESB") || tempString.equals("RESW")) {

str2 = tempString;

isOpCode = true;

} else {

for (i = 0; i < op\_TAB.length; i++) {

if (tempString.equals(op\_TAB[i])) {

str2 = tempString;

isOpCode = true;

break;

}

}

}

if (tempString.equals(".")) {

str1 = ".";

isLine = true;

}

if (str2.equals("RSUB")) {

isLine = true;

}

if (!isOpCode) {

str1 = tempString;

} else if (!str2.equals("RSUB")) {

str3 = scn.next();

isLine = true;

}

if (isLine) {

Data.add(new Data2(str1,str2,str3,str1+str2+str3));

str1 = " ";

str2 = " ";

str3 = " ";

isLine = false;

isOpCode = false;

}

}

fr.close();

for (i = 0; i < Data.size(); i++) {

// Calculate location (address)

if (Data.get(i).getStr().contains(".")) {

Location.add("");

HexLoc = (Integer.toHexString(DecLoc += Integer.parseInt(Integer.toString(num),16))).toUpperCase();

} else {

if (i != 1) {

HexLoc = (Integer.toHexString(DecLoc += Integer.parseInt(Integer.toString(num),16))).toUpperCase();

if (i == Data.size() - 1) {

HexLoc = "";

}

} else { // Starting point "1"

HexLoc = Integer.toString(n);

}

Location.add(HexLoc);

}

if (i == 0) { // Default location (address)

n = Integer.parseInt(Data.get(i).getThird());

DecLoc = Integer.parseInt(Integer.toString(n), 16);

Location.remove(0);

Location.add(0, Integer.toString(n));

}

// Calculate the length

if (Data.get(i).getStr().contains(".") || i == 0) {

num = 0;

} else if (Data.get(i).getSecond().equals("BYTE")) { // "BYTE" Type C & Type X

if (Data.get(i).getThird().contains("C")) { // C'EOF' with length 3

char[] c = Data.get(i).getThird()

.substring(Data.get(i).getThird().indexOf('\'') + 1,

Data.get(i).getThird().length() - 1).toCharArray();

num = c.length;

} else {

num = 1; // X'F1' with length 1

}

} else if (Data.get(i).getSecond().contains("RESW")) { // "RESW" with 3 times

num = Integer.parseInt(Data.get(i).getThird()) \* 3;

} else if (Data.get(i).getSecond().contains("RESB")) {

num = Integer.parseInt(Integer.toHexString(Integer.parseInt(Data.get(i).getThird())));

} else { // Leftovers with "WORD"

num = 3;

}

// Create "SYM\_TAB"

if (!Data.get(i).getFirst().contains(" ") && i != 0 && !Data.get(i).getFirst().contains(".")) {

SYM\_TAB.add(new Pair2(Data.get(i).getFirst(), HexLoc));

}

}

/// Pass 2 create object code

for (i = 0; i < Data.size(); i++) {

StringBuilder s = new StringBuilder("");

for (j = 0; j < op\_TAB.length; j++) {

if (Data.get(i).getSecond().equals(op\_TAB[j])) {

s.append(opCode[j]);

break;

}

}

for (j = 0; j < SYM\_TAB.size(); j++) {

if (Data.get(i).getThird().contains(",X")) {

if (SYM\_TAB.get(j).getSymbol().equals(Data.get(i)

.getThird().substring(0,Data.get(i).getThird().length()-2))) { // Add 8000 for BUFFER with "X"

// Hexa to Deci addition, then exchange back to Hexa

s.append(Integer.toHexString(Integer.parseInt(SYM\_TAB.get(j)

.getLocation(), 16) + Integer.parseInt("8000",16)));

break;

}

} else if (SYM\_TAB.get(j).getSymbol().equals(Data.get(i).getThird())) {

// "SYM\_TAB" table check & fill up digits

if (s.length() + SYM\_TAB.get(j).getLocation().length() != 6) {

int len = s.length() + SYM\_TAB.get(j).getLocation().length();

s.append("0".repeat(Math.max(0, 6 - len)));

s.append(SYM\_TAB.get(j).getLocation());

} else {

s.append(SYM\_TAB.get(j).getLocation());

}

break;

}

}

switch (Data.get(i).getSecond()) {

case "BYTE" -> { // Deal with "X" & "C"

char[] c = Data.get(i).getThird()

.substring(Data.get(i).getThird().indexOf('\'') + 1, Data.get(i).getThird().length() - 1).toCharArray();

for (char value : c) {

if (Data.get(i).getThird().contains("C")) {

s.append(Integer.toHexString(value).toUpperCase()); // ASCII code from 10 to 16 decimal

} else {

s.append(value);

}

}

}

case "WORD" -> { // Deal with "Data.get(i).getThird()" and fill up to six digit object code

s.append(Integer.toHexString(Integer.parseInt(Data.get(i).getThird())).toUpperCase());

if (s.length() < 6) {

s.reverse();

int len = s.length();

s.append("0".repeat(Math.max(0, 6 - len)));

s.reverse();

} else if (s.length() > 6) {

s = new StringBuilder(s.substring(2, 8));

}

}

case "RSUB" -> // "RSUB" fill up with "Zeros" for six digit

s.append("0000");

}

Target.add(s.toString()); // Store the final result to the object code section

if(Data.get(i).getFirst().equals(".")){

Target.remove(i);

Target.add("");

}

}

Target.remove(Target.size() - 1);

Target.add("");

// Result Display & File Output

PrintWriter Write = new PrintWriter("SIC\_Result.txt");

System.out.printf("%s\t%-6s\t%-6s\t%-5s\t%s\t\r\n", "Address", " ", "Statement", " ", "Object Code");

Write.printf("%s\t%-6s\t%-6s\t%-5s\t%s\t\r\n", "Address", " ", "Statement", " ", "Object Code");

System.out.println("------------------------------------------------");

Write.println("------------------------------------------------");

for (j = 0; j < Data.size(); j++) {

Write.printf("%s\t%-6s\t%-6s\t%-10s\t%s\t\r\n", Location.get(j), Data.get(j).getFirst(),

Data.get(j).getSecond(), Data.get(j).getThird(), Target.get(j));

System.out.printf("%s\t%-6s\t%-6s\t%-10s\t%s\t\r\n", Location.get(j), Data.get(j).getFirst(),

Data.get(j).getSecond(), Data.get(j).getThird(), Target.get(j));

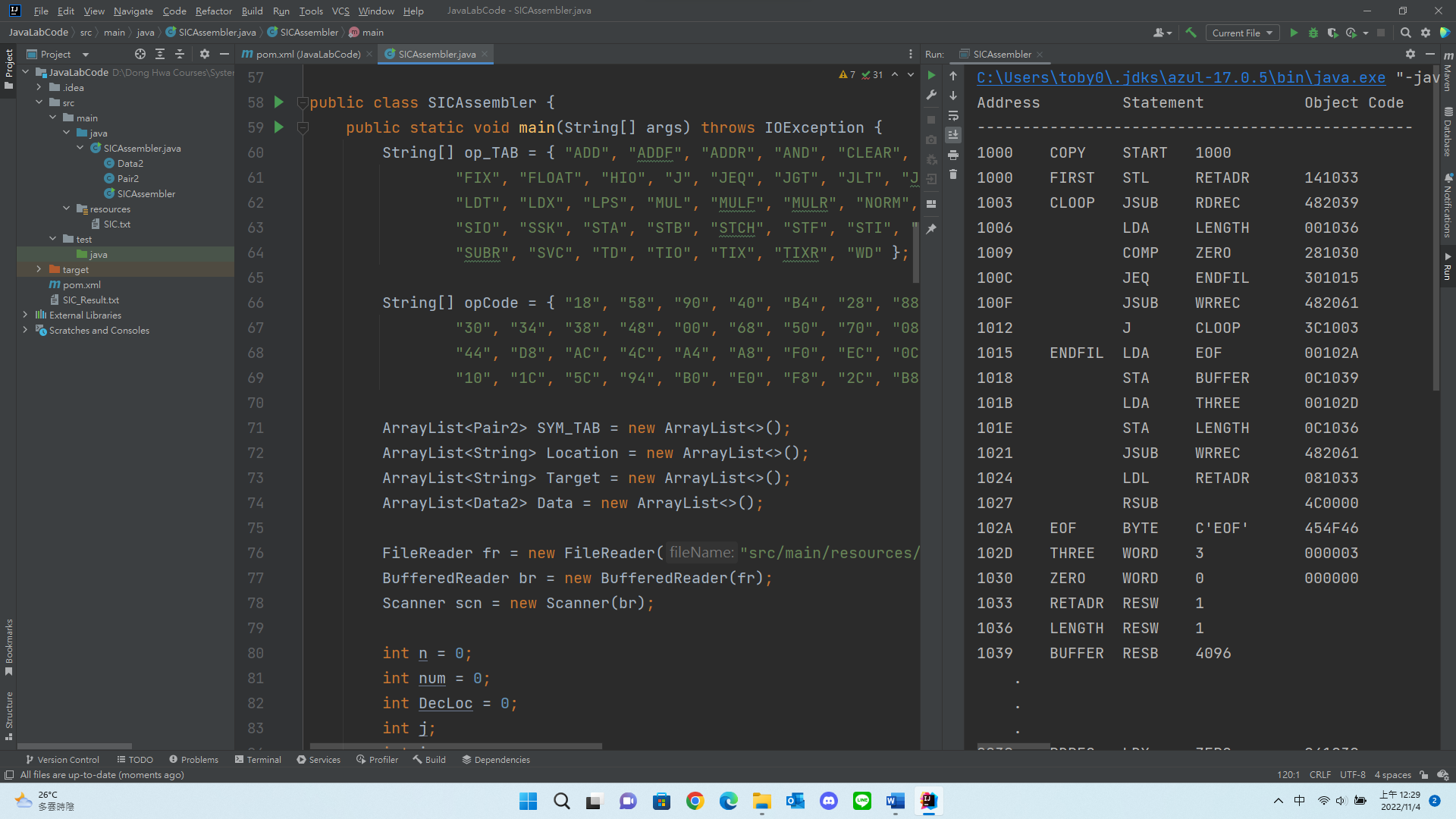
}

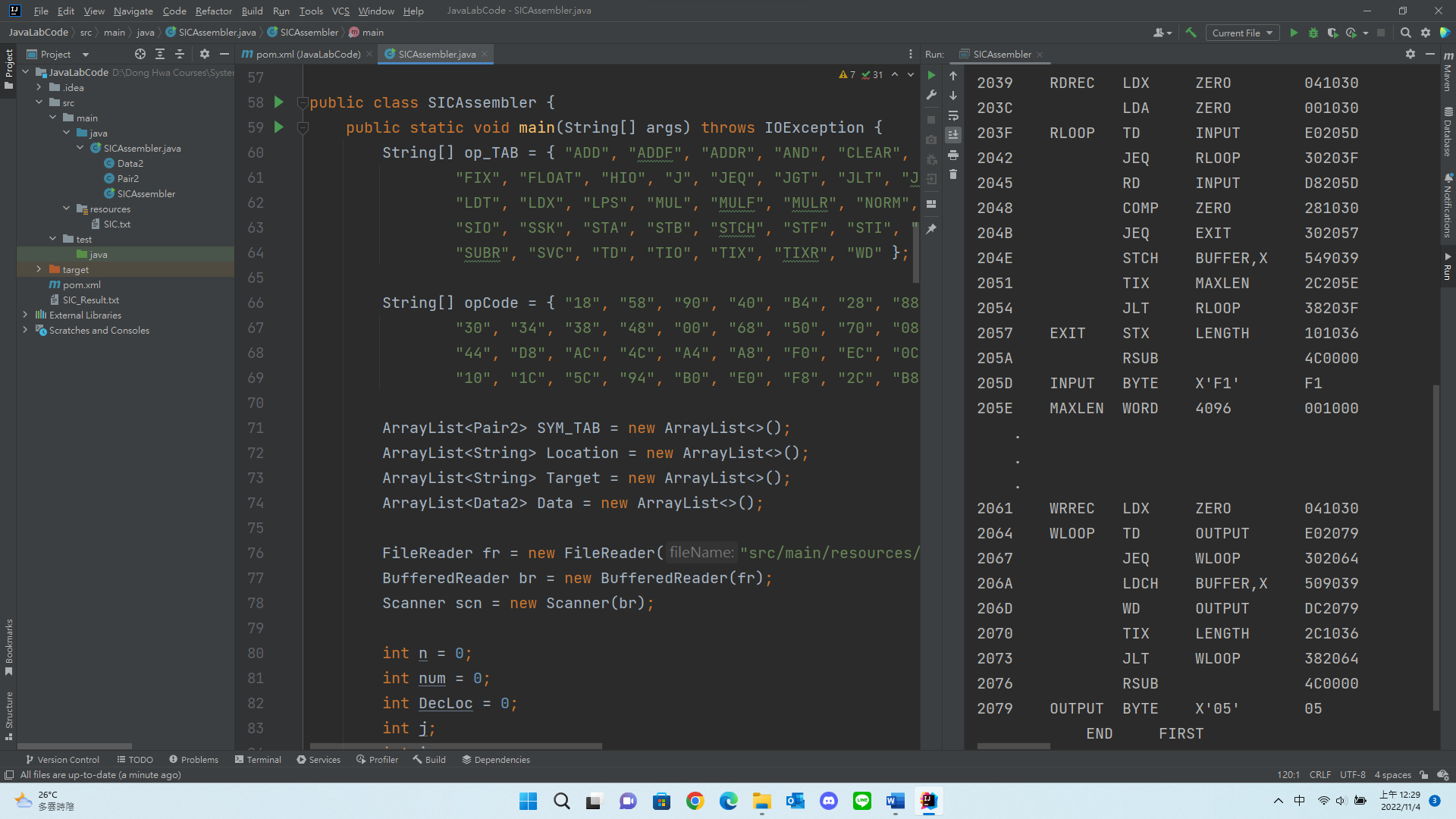
Write.close();

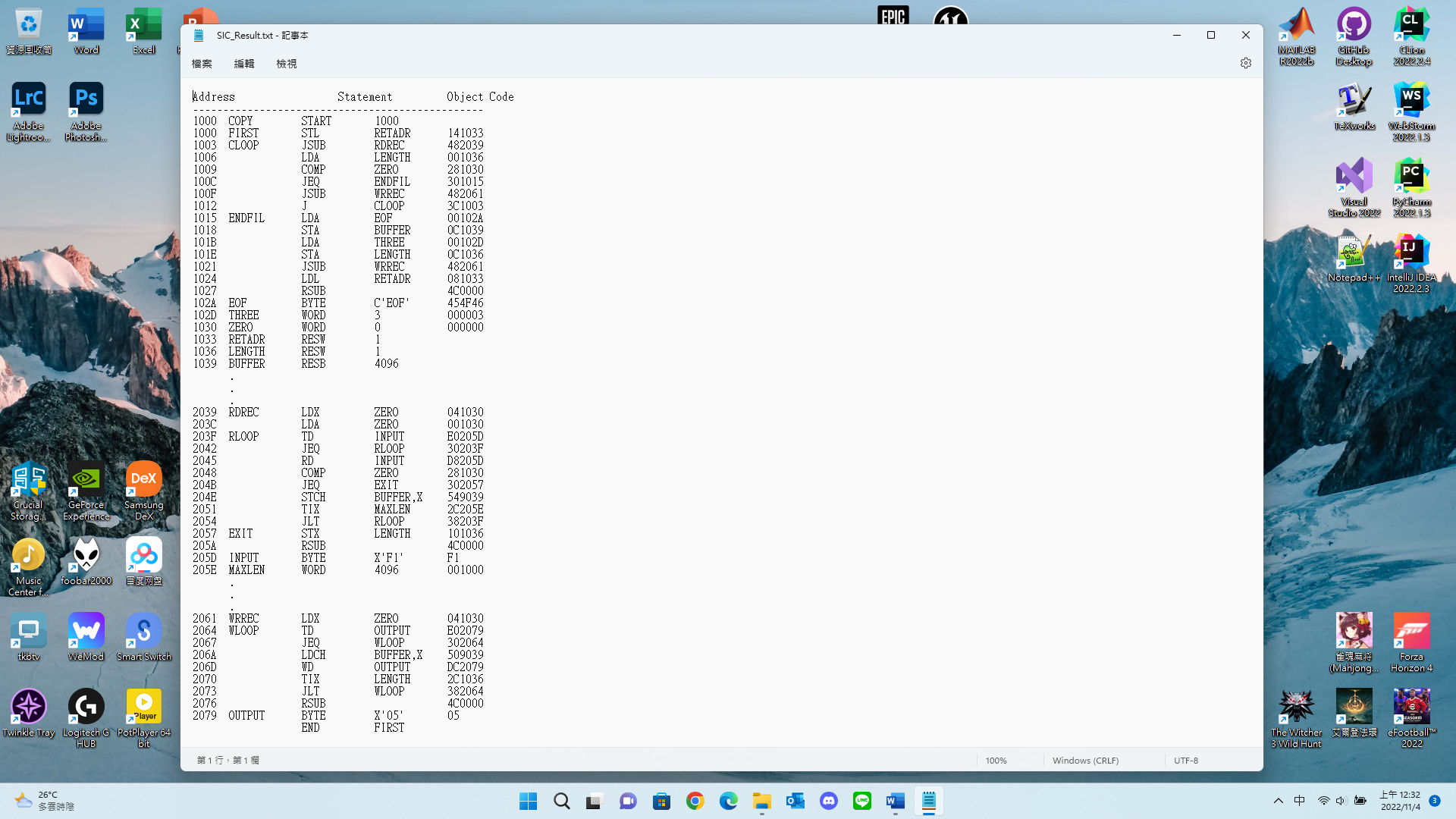
}

}

**Test Run & File Output**







**Discussion**