|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 04** | **ASSEMBLER** | | |
| Date of Exercise | \_\_\_\_\_\_\_\_\_\_\_\_ | Date of Output Verification | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Question**

Write a program to simulate a multi pass assembler.

**Program**

//@author William Scott

import java.io.\*;

import java.util.\*;

public class CompilerLabExp4Assembler {

//UR12CS135 - P.William Scott - Exp 4 - Assembler

public static BufferedReader broptable, brsiccode, brsiccode2, broutputaddress, broutputaddress2, broutputaddress3, broutputsymboltable, broutputobjectcode;

public static BufferedWriter bwaddress, bwsymboltable, bwobjectcode;

public static Scanner in = new Scanner(System.in);

public static String line = "", t = "", s[] = new String[3];

public static ArrayList optableoperand = new ArrayList(), optablevalue = new ArrayList(), operands = new ArrayList(), operandaddress = new ArrayList();

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

System.out.println("UR12CS135 - Assembler");

showinputfile();

storeoptable();

showoptable();

generateaddress();

showoperands();

showoutputaddress();

generatesymboltable();

showsymboltable();

generateobjectcode();

showobjectcode();

}

public static void showoutputaddress() throws IOException {

broutputaddress = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Output Address File.txt"));

System.out.println("\n--------Output Address File----------");

String a;

while ((a = broutputaddress.readLine()) != null) {

System.out.println(a);

}

}

public static void showinputfile() throws IOException {

brsiccode2 = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Input File - SIC Code.txt"));

System.out.println("\n--------Input File----------");

String a;

while ((a = brsiccode2.readLine()) != null) {

System.out.println(a);

}

}

public static void showsymboltable() throws IOException {

broutputsymboltable = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Output SymbolTable File.txt"));

System.out.println("\n---Generated SymbolTable File----");

String a;

while ((a = broutputsymboltable.readLine()) != null) {

System.out.println(a);

}

}

public static void showoptable() {

System.out.println("\n------Stored Optable------");

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

System.out.println(optableoperand.get(i) + " - " + optablevalue.get(i));

}

}

public static void showoperands() {

System.out.println("\n------Stored Operands------");

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

System.out.println(operands.get(i));

}

}

public static void showobjectcode() throws FileNotFoundException, IOException {

broutputobjectcode = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Output ObjectCode File.txt"));

System.out.println("\n---Generated ObjectCode File----");

String a;

while ((a = broutputobjectcode.readLine()) != null) {

System.out.println(a);

}

}

public static void generateaddress() throws IOException {

brsiccode = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Input File - SIC Code.txt"));

bwaddress = new BufferedWriter(new FileWriter(new File("..\\Exp 4 - Assembler Output Address File.txt")));

bwaddress.flush();

String d = "";

System.out.println("\n----Generating Address----");

while (!(line = brsiccode.readLine()).contains("START")) {

System.out.println(line);

bwaddress.write("Address" + "\t" + line);

bwaddress.newLine();

}

do {

s = line.split("\t", 3);

System.out.println(line);

if (line.contains("START")) {

t = (s[2]);

d = "";

line = s[0] + "\t" + s[1] + "\t" + "0";

} else {

d = t;

t = addhex(t, getadditionvalue());

}

bwaddress.write(d + "\t" + line);

bwaddress.newLine();

} while (!(line = brsiccode.readLine()).contains("END"));

bwaddress.write(addhex(t, "1") + "\t" + line);

bwaddress.newLine();

bwaddress.close();

}

public static void generatesymboltable() throws IOException {

broutputaddress2 = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Output Address File.txt"));

bwsymboltable = new BufferedWriter(new FileWriter(new File("..\\Exp 4 - Assembler Output SymbolTable File.txt")));

bwsymboltable.flush();

String a, t[] = new String[2];

bwsymboltable.write("Name\tValue");

bwsymboltable.newLine();

while ((a = broutputaddress2.readLine()) != null) {

t = a.split("\t", 3);

if (operands.contains(t[1])) {

operandaddress.add(t[0]);

bwsymboltable.write(t[0] + "\t" + t[1]);

bwsymboltable.newLine();

}

}

bwsymboltable.close();

}

public static void generateobjectcode() throws FileNotFoundException, IOException {

broutputaddress3 = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Output Address File.txt"));

bwaddress = new BufferedWriter(new FileWriter(new File("..\\Exp 4 - Assembler Output ObjectCode File.txt")));

bwaddress.flush();

String a, t[] = new String[4], r;

int e;

while ((a = broutputaddress3.readLine()) != null) {

t = a.split("\t", 4);

if (optableoperand.contains(t[2])) {

a += "\t" + getopvalue(t[2]) + getoperandaddress(t[3]);

} else if (t[2].contains("BYTE")) {

r = t[3].substring(2, 3);

a += "\t" + Integer.toHexString((int) r.charAt(0)).toUpperCase();

} else if (t[2].contains("WORD")) {

a += "\t" + Integer.parseInt(t[3], 16);

} else if (a.contains("operand")) {

a += "\tObjectCode";

}

bwaddress.write(a);

bwaddress.newLine();

}

bwaddress.close();

}

public static String getopvalue(String a) {

int v;

String val;

v = optableoperand.indexOf(a);

return (String) optablevalue.get(v);

}

public static String getoperandaddress(String a) {

int v;

String val;

v = operands.indexOf(a);

return (String) operandaddress.get(v);

}

public static void storeoptable() throws IOException {

broptable = new BufferedReader(new FileReader("..\\Exp 4 - Assembler Input File - OpTable.txt"));

String a, t[] = new String[2];

while ((a = broptable.readLine()) != null) {

t = a.split("\t", 2);

optableoperand.add(t[0]);

optablevalue.add(t[1]);

}

}

public static String getadditionvalue() {

String val = "0", tr;

int p;

if (optableoperand.contains(s[1])) {

val = "3";

operands.add(s[2]);

} else if (s[1].contains("WORD")) {

val = "3";

} else if (s[1].contains("RESW")) {

p = 3 \* Integer.parseInt(s[2]);

val = String.valueOf(p);

} else if (s[1].contains("BYTE") || s[1].contains("END")) {

val = "1";

}

return val;

}

public static String addhex(String a, String b) {

String c = "";

int t;

t = Integer.parseInt(a, 16) + Integer.parseInt(b, 16);

c = Integer.toHexString(t);

return c;

}

}

**Input**

*Exp 4 - Assembler Input File – OpTable.txt*

LDA 00

STA 0C

LDCH 50

STCH 54

*Exp 4 - Assembler Input File - SIC Code.txt*

Label opcode operand

\*\* START 1000

\*\* LDA FIVE

\*\* STA ALPHA

\*\* LDCH CHARZ

\*\* STCH C1

ALPHA RESW 1

FIVE WORD 5

CHARZ BYTE C'Z'

C1 RESB 1

\*\* END \*\*

**Output**

*Exp 4 - Assembler Output Address File.txt*

Address Label opcode operand

\*\* START 0

1000 \*\* LDA FIVE

1003 \*\* STA ALPHA

1006 \*\* LDCH CHARZ

1009 \*\* STCH C1

100c ALPHA RESW 1

100f FIVE WORD 5

1012 CHARZ BYTE C'Z'

1013 C1 RESB 1

1014 \*\* END \*\*

*Exp 4 - Assembler Output ObjectCode File.txt*

Address Label opcode operand ObjectCode

\*\* START 0

1000 \*\* LDA FIVE 00100c

1003 \*\* STA ALPHA 0C100f

1006 \*\* LDCH CHARZ 501012

1009 \*\* STCH C1 541013

100c ALPHA RESW 1

100f FIVE WORD 5 5

1012 CHARZ BYTE C'Z' 5A

1013 C1 RESB 1

1014 \*\* END \*\*

*Exp 4 - Assembler Output SymbolTable File.txt*

Name Value

100c ALPHA

100f FIVE

1012 CHARZ

1013 C1

**Result**

Two pass assembler is successfully implemented in java.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Jeban Chandir Moses

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_