**Assembler pass2**

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** pass2 {

**static** Obj[] *symb\_table* = **new** Obj[10];

**static** Obj[] *literal\_table* = **new** Obj[10];

**static** **int** *symb\_found* = 0;

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

// **TODO** Auto-generated method stub

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("ENTER TOTAL NUMBER OF SYMBOLS: ");

**int** total\_symb = sc.nextInt();

**int** pos, num;

**for** (**int** i = 0; i < total\_symb; i++) {

*symb\_table*[i] = **new** Obj("", 0);

System.***out***.println("ENTER SYMBOL NAME: ");

*symb\_table*[i].name = sc.next();

System.***out***.println("ENTER SYMBOL ADDRESS: ");

*symb\_table*[i].addr = sc.nextInt();

}

System.***out***.println("ENTER TOTAL NUMBER OF LITERALS: ");

**int** total\_ltr = sc.nextInt();

**for** (**int** i = 0; i < total\_ltr; i++) {

*literal\_table*[i] = **new** Obj("", 0);

System.***out***.println("ENTER LITERAL NAME: ");

*literal\_table*[i].name = sc.next();

System.***out***.println("ENTER LITERAL ADDRESS: ");

*literal\_table*[i].addr = sc.nextInt();

}

System.***out***.println("\n\*\* SYMBOL TABLE \*\*");

System.***out***.println("\nSYMBOL\tADDRESS");

**for** (**int** i = 0; i < total\_symb; i++) {

System.***out***.println(*symb\_table*[i].name + "\t" + *symb\_table*[i].addr);

}

System.***out***.println("\n\*\* LITERAL TABLE \*\*");

System.***out***.println("\nIndex\tLITERAL\tADDRESS");

**for** (**int** i = 0; i < total\_ltr; i++) {

System.***out***.println((i + 1) + "\t" + *literal\_table*[i].name + "\t" + *literal\_table*[i].addr);

}

BufferedReader br2 = **new** BufferedReader(**new** FileReader("K:\Drive\OneDrive\Desktop\LP1 SPOS\Code\Output2.txt"));

String line;

**boolean** symbol\_error = **false**, undef\_mnemonic = **false**;

System.***out***.println("\n\*\* OUTPUT FILE \*\*\n\n");

lab:

**while** ((line = br2.readLine()) != **null**) {

String[] token\_list = line.split("\\s+", 5);

symbol\_error = **false**;

undef\_mnemonic = **false**;

labl:

**for** (String token : token\_list) {

**if** (token.length() > 0) {

pos = -1;

**if** (token.matches("---")) {

System.***out***.print("\t---");

undef\_mnemonic = **true**;

} **else** **if** (token.matches("[0-9]+")) { // LOCATION COUNTER

System.***out***.print("\n\n" + token);

} **else** {

String letters = token.replaceAll("[^A-Za-z]+", "");

num = Integer.*parseInt*(token.replaceAll("[^0-9]+", ""));

**if** (token.matches("\\([0-9]+\\)")) {

System.***out***.print("\t" + num);

} **else** {

**switch** (letters.toUpperCase()) {

**case** "S":

**if** (*symb\_table*[num - 1].addr == 0) {

System.***out***.print("\t---");

symbol\_error = **true**;

} **else** {

System.***out***.print("\t" + *symb\_table*[num - 1].addr);

}

**break**;

**case** "L":

System.***out***.print("\t" + *literal\_table*[num - 1].addr);

**break**;

**case** "AD":

System.***out***.print("\n");

**continue** lab;

**case** "DL":

**switch** (num) {

**case** 1:

System.***out***.print("\n");

**continue** lab;

**case** 2:

System.***out***.print("\t 00 \t 00");

}

**continue** labl;

**case** "C":

System.***out***.print("\t" + num);

**break**;

**default**:

System.***out***.print("\t" + "00" + num);

}

**if** (symbol\_error) {

System.***out***.print("\n\n\*\* SYMBOL IS NOT DEFINED \*\*");

}

**if** (undef\_mnemonic) {

System.***out***.print("\n\n\*\* INVALID MNEMONIC \*\*");

}

}

}

}

}

}

**int**[] flag = **new** **int**[total\_symb];

**for** (**int** i = 0; i < total\_symb; i++) {

*symb\_found* = 0;

**for** (**int** j = 0; j < total\_symb; j++) {

**if** (*symb\_table*[i].name.equalsIgnoreCase(*symb\_table*[j].name) && flag[j] == 0) {

*symb\_found*++;

flag[i] = flag[j] = 1;

}

}

**if** (*symb\_found* > 1) {

System.***out***.print("\n\n\*\* '" + *symb\_table*[i].name + "' IS A DUPLICATE SYMBOL \*\*");

}

}

br2.close();

sc.close();

}

}

**Output.txt**

(AD,1) (C,100)

100 (IS,4) (RG,1) (C,05)

101 (IS,4) (RG,2) (C,10)

102 (S,1) (IS,1) (RG,1) (RG,2)

103 (IS,5) (S,2) (L,1)

104 (IS,3) (RG,1) (S,1)

105 (AD,3) (C,102)

102 (AD,5) (DL,2) (C,5)

103 (IS,5) (S,3) (L,2)

104 (IS,5) (S,4) (L,3)

105 (AD,5) (DL,2) (C,8)

106 (DL,2) (C,8)

107 (IS,5) (S,2) (L,4)

108 (IS,5) (S,3) (L,5)

109 (DL,1) (C,02)

111 (DL,2) (C,10)

112 (DL,1) (C,09)

121 (S,5) (AD,4) (S,1)

122 (AD,2)

(DL,2) (C,7)

123 (DL,2) (C,8)

124

**Obj.java**

**public** **class** Obj {

String name;

**int** addr;

Obj(String nm, **int** address)

{

**this**.name=nm;

**this**.addr=address;

}

}

**OUTPUT**



