```
package ASSEM_PASS_2;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.StringTokenizer;
import ASSEM_PASS_2.Tuple;
import ASSEM_PASS_2.SymTuple;
import ASSEM_PASS_2.LitTuple;
class Tuple {
//m_class specifies class of the mnemonic such as IS, DL, or AD
String mnemonic, m_class, opcode;
int length;
Tuple() {}
Tuple(String s1, String s2, String s3, String s4) {
mnemonic = s1;
m_class = s2;
opcode = s3;
length = Integer.parseInt(s4);
}
}
class SymTuple {
String symbol, address, length;
SymTuple(String s1, String s2, String i1) {
symbol = s1;
address = s2;
length = i1;
}
}
class LitTuple {
```

```
String literal, address, length;
LitTuple() {}
LitTuple(String s1, String s2, String i1) {
literal = s1;
address = s2;
length = i1;
}
}
public class pass2 {
static int lc,iSymTabPtr=0, iLitTabPtr=0, iPoolTabPtr=0;
static int poolTable[] = new int[10];
static Map<String,Tuple> MOT;
static ArrayList<SymTuple> symtable;
static ArrayList<LitTuple> littable;
static Map<String, String> regAddressTable;
static PrintWriter out_pass2;
static void initiallizeTables() throws Exception{
symtable = new ArrayList<>();
littable = new ArrayList<>();
regAddressTable = new HashMap<>();
//MOT = new HashMap<>();
String s;
BufferedReader br;
br = new BufferedReader(new InputStreamReader(new
FileInputStream("C:\\Users\\Store\\Desktop\\LP1\\LP1\\symtable.txt")));
while((s = br.readLine()) != null) {
StringTokenizer st = new StringTokenizer(s, "\t", false);
symtable.add(new SymTuple(st.nextToken(), st.nextToken(), ""));
}
br.close();
br = new BufferedReader(new InputStreamReader(new
FileInputStream("C:\\Users\\Store\\Desktop\\LP1\\LP1\\littable.txt")));
while((s = br.readLine()) != null) {
StringTokenizer st = new StringTokenizer(s, "\t", false);
littable.add(new LitTuple(st.nextToken(), st.nextToken(), ""));
}
br.close();
```

```
//Initiallize register address table
regAddressTable.put("AREG", "1");
regAddressTable.put("BREG", "2");
regAddressTable.put("CREG", "3");
regAddressTable.put("DREG", "4");
}
static void pass2() throws Exception{
BufferedReader input = new BufferedReader(new InputStreamReader(new
FileInputStream("C:\\Users\\Store\\Desktop\\LP1\\LP1\\output_pass1.txt")));
out_pass2 = new PrintWriter(new
FileWriter("C:\\Users\\Store\\Desktop\\LP1\\LP1\\ASSEM_PASS_2\\output_pass2.txt"), true);
String s;
//Read from intermediate file one line at a time
while((s = input.readLine()) != null) {
//Replace all ( and ) characters by a blank string
s=s.replaceAll("(\\()", " ");
s=s.replaceAll("(\\))", " ");
//For each line, separate out the tokens
String ic_tokens[] = tokenizeString(s, " ");
if(ic_tokens == null || ic_tokens.length==0){
continue;
}
String output_str = "";
//Second token contains mnemonic class and opcode
String mnemonic_class = ic_tokens[1];
//Separate the mnemonic and its opcode which are separated by a comma
String m_tokens[] = tokenizeString(mnemonic_class, ",");
//Write the second token as is in the output file
if(m_tokens[0].equalsIgnoreCase("IS")){
//First token is location counter which will be output as it is
output_str += ic_tokens[0] + " ";
//Output the opcode of the instruction
output_str += m_tokens[1] + " ";
String opr_tokens[];
for(int i = 2; i <ic_tokens.length; i++){
opr_tokens = tokenizeString(ic_tokens[i], ",");
if(opr_tokens[0].equalsIgnoreCase("RG")){
```

```
output_str += opr_tokens[1] + " ";
}
else if(opr_tokens[0].equalsIgnoreCase("S")){
int index = Integer.parseInt(opr_tokens[1]);
output_str += symtable.get(index).address + " ";
}
else if(opr_tokens[0].equalsIgnoreCase("L")){
int index = Integer.parseInt(opr_tokens[1]);
output_str += littable.get(index).address + " ";
}
}
}
else if(m_tokens[0].equalsIgnoreCase("DL")){
//First token is location counter which will be output as it is
output_str += ic_tokens[0] + " ";
if(m_tokens[1].equalsIgnoreCase("02")){
//Process for operands of mnemonic DC
String opr_tokens[] = tokenizeString(ic_tokens[2], ",");
output_str += "00 00 " + opr_tokens[1] + " ";
}
}
System.out.println(output_str);
out_pass2.println(output_str);
}
}
static String[] tokenizeString(String str, String separator){
StringTokenizer st = new StringTokenizer(str, separator, false);
//Construct an array of the separated tokens
String s_arr[] = new String[st.countTokens()];
for(int i=0 ; i < s_arr.length ; i++) {
s_arr[i] = st.nextToken();
}
return s_arr;
}
public static void main(String[] args) throws Exception {
System.out.println("Name: Bhavika Patil \nRoll No. TBCO22172\n");
```

```
initiallizeTables();
pass2();
}
}
```