```
import java.io.*;
import java.util.*;

/*
Pass2.java
Usage:
javac Pass2.java
java Pass2 IC.txt

Reads:
IC.txt -> intermediate code produced by Pass1
SYMTAB.txt -> symbol table (index SYMBOL ADDRESS LENGTH)
LITTAB.txt -> literal table (index LITERAL ADDRESS)

Produces:
MACHINECODE.txt and console printed machine code.
*/

public class pass2 {
static Map<Integer, Integer> symAddr = new HashMap<>(); // index -> address
static Map<Integer, Integer> litAddr = new HashMap<>(); // index -> address

public static void main(String[] args) throws Exception {
if(args.length==0){
System.out.println("Usage: java Pass2 <IC_file>");
return;
}
// Load SYMTAB
try(BufferedReader br = new BufferedReader(new FileReader("SYMTAB.txt"))){
String line;
while((line = br.readLine()) != null){
line = line.trim();
if(line.isEmpty()) continue;
String[] p = line.split("\\s+");
// format: index SYMBOL ADDRESS LENGTH
int idx = Integer.parseInt(p[0]);
int addr = Integer.parseInt(p[2]);
symAddr.put(idx, addr);
}
} catch(Exception e){
System.out.println("Warning: could not read SYMTAB.txt: " + e.getMessage());
}

// Load LITTAB
try(BufferedReader br = new BufferedReader(new FileReader("LITTAB.txt"))){
String line;
while((line = br.readLine()) != null){
line = line.trim();
if(line.isEmpty()) continue;
String[] p = line.split("\\s+");
// format: index LITERAL ADDRESS
int idx = Integer.parseInt(p[0]);
int addr = Integer.parseInt(p[2]);
litAddr.put(idx, addr);
}
} catch(Exception e){
System.out.println("Warning: could not read LITTAB.txt: " + e.getMessage());
}

// Read intermediate code
List<String> machine = new ArrayList<>();
try(BufferedReader br = new BufferedReader(new FileReader(args[0]))){
String ic;
while((ic = br.readLine()) != null){
```

```
ic = ic.trim();
if(ic.isEmpty()) continue;

// Examples of IC lines:
// (AD,01)(C,200)
// (IS,04)(1)(L,1)
// (DL,01)(C,5)
// (IS,00)
// We'll parse tokens like (XX,YY) and (S,n) or (L,n) or (C,n) or register
numbers

List<String> fields = new ArrayList<>();
int i=0;
while(i<ic.length()){
if(ic.charAt(i)=='('){
int j = ic.indexOf(')', i);
if(j==-1) break;
fields.add(ic.substring(i+1,j));
i = j+1;
} else i++;
}

if(fields.size()==0) continue;
String head = fields.get(0); // like "AD,01" or "IS,04" or "DL,01"
String[] headParts = head.split(",");
String cls = headParts[0];
String code = headParts.length>1 ? headParts[1] : "00";

if(cls.equals("AD")){
// No Machine code usually
// except some pseudo directives handled earlier
// For readability produce + 00 0 000 line if needed - we'll skip
} else if(cls.equals("DL")){
// (DL,01)(C,5) => DC produce data constant
String constVal = "0";
for(int k=1;k<fields.size();k++){
if(fields.get(k).startsWith("C,")){
constVal = fields.get(k).split(",")[1];
}
}
String mc = "+ 00 0 " + String.format("%03d", Integer.parseInt(constVal));
machine.add(mc);
} else if(cls.equals("IS")){
// fields may be: (IS,04) (reg) (L,1) OR (IS,04) (reg) (S,1)
String opcode = code;
String reg = "0";
String addrStr = "000";

if(fields.size()>=2){
String f1 = fields.get(1);
// sometimes register is encoded as number alone (like "1")
if(f1.matches("\\d+")){
reg = f1;
} else if(f1.startsWith("R") && f1.length()>1 &&
Character.isDigit(f1.charAt(1))){
reg = f1.substring(1);
} else {
// if f1 is something like "1" still accept
if(f1.matches("\\d+")) reg = f1;
}
}

// look for S,n or L,n or C,n
for(int k=1;k<fields.size();k++){
```

```
String f = fields.get(k);
if(f.startsWith("S,")){
int idx = Integer.parseInt(f.split(",")[1]);
int addr = symAddr.getOrDefault(idx, 0);
addrStr = String.format("%03d", addr);
} else if(f.startsWith("L,")){
int idx = Integer.parseInt(f.split(",")[1]);
int addr = litAddr.getOrDefault(idx, 0);
addrStr = String.format("%03d", addr);
} else if(f.startsWith("C,")){
int c = Integer.parseInt(f.split(",")[1]);
addrStr = String.format("%03d", c);
} else if(f.matches("\\d+")) {
// reg
reg = f;
}
}

String mc = "+ " + String.format("%02d", Integer.parseInt(opcode)) + " " + reg +
" " + addrStr;
machine.add(mc);
} else {
// unknown class
}
}
}

// write MACHINECODE.txt and print
try(PrintWriter pw = new PrintWriter(new FileWriter("MACHINECODE.txt"))){
for(String s: machine){
pw.println(s);
System.out.println(s);
}
}
System.out.println("\nMachine code written to MACHINECODE.txt");
}
}
```