import java.io.\*;

//import java.util.\*;

class Pass1 {

public static void main(String[] args) {

String REF[] = {"ax", "bx", "cx", "dx"};

String IS[] = {"stop", "add", "sub", "mult", "mover", "movem", "comp", "bc", "div", "read"};

String DL[] = {"ds", "dc"};

int temp1 = 0;

int f = 0;

Obj[] literal\_table = new Obj[50];

Obj[] symbol\_table = new Obj[50];

Obj[] optab = new Obj[100];

Pooltable[] pooltab = new Pooltable[5];

String line;

try (BufferedReader br = new BufferedReader(new FileReader("D:\\TCOB22\\Assembler\\Sample.txt"));

BufferedWriter bw = new BufferedWriter(new FileWriter("D:\\TCOB22\\Assembler\\Output.txt"))) {

boolean start = false;

boolean end = false, filladdr = false, ltorg = false;

int total\_symb = 0, total\_ltr = 0, optab\_cnt = 0, pooltab\_cnt = 0, loc = 0, temp, pos;

while ((line = br.readLine()) != null && !end) {

String tokens[] = line.split(" ", 4);

if (loc != 0 && !ltorg) {

if (f == 1) {

ltorg = false;

loc = loc + temp1 - 1;

bw.write("\n" + loc);

f = 0;

loc++;

} else {

bw.write("\n" + loc);

ltorg = false;

loc++;

}

}

ltorg = filladdr = false;

for (int k = 0; k < tokens.length; k++) {

pos = -1;

if (start) {

loc = Integer.*parseInt*(tokens[k]);

start = false;

}

switch (tokens[k]) {

case "start":

start = true;

pos = 1;

bw.write("\t(AD," + pos + ")");

break;

case "end":

end = true;

pos = 2;

bw.write("\t(AD," + pos + ")\n");

for (temp = 0; temp < total\_ltr; temp++) {

if (literal\_table[temp].addr == 0) {

literal\_table[temp].addr = loc - 1;

bw.write("\t(DL,2)\t(C," + literal\_table[temp].name + ")\n" + loc++);

}

}

break;

case "origin":

pos = 3;

bw.write("\t(AD," + pos + ")");

pos = *search*(tokens[++k], symbol\_table, total\_symb);

k++;

bw.write("\t(C," + symbol\_table[pos].addr + ")");

loc = symbol\_table[pos].addr;

break;

case "ltorg":

ltorg = true;

pos = 5;

bw.write("\t(AD," + pos + ")\n");

for (temp = 0; temp < total\_ltr; temp++) {

if (literal\_table[temp].addr == 0) {

literal\_table[temp].addr = loc - 1;

bw.write("\t(DL,2)\t(C," + literal\_table[temp].name + ")\n" + loc++);

}

}

if (pooltab\_cnt == 0) {

pooltab[pooltab\_cnt++] = new Pooltable(0, total\_ltr);

} else {

pooltab[pooltab\_cnt] = new Pooltable(pooltab[pooltab\_cnt - 1].first + pooltab[pooltab\_cnt - 1].total\_literals, total\_ltr - pooltab[pooltab\_cnt - 1].first - 1);

pooltab\_cnt++;

}

break;

case "equ":

pos = 4;

bw.write("\t(AD," + pos + ")");

String prev\_token = tokens[k - 1];

int pos1 = *search*(prev\_token, symbol\_table, total\_symb);

pos = *search*(tokens[++k], symbol\_table, total\_symb);

symbol\_table[pos1].addr = symbol\_table[pos].addr;

bw.write("\t(S," + (pos + 1) + ")");

break;

default:

if (pos == -1) {

pos = *search*(tokens[k], IS);

if (pos != -1) {

bw.write("\t(IS," + pos + ")");

optab[optab\_cnt++] = new Obj(tokens[k], pos);

} else {

pos = *search*(tokens[k], DL);

if (pos != -1) {

if (pos == 0) f = 1;

bw.write("\t(DL," + (pos + 1) + ")");

optab[optab\_cnt++] = new Obj(tokens[k], pos);

filladdr = true;

} else if (tokens[k].matches("[a-zA-Z]+:")) {

pos = *search*(tokens[k], symbol\_table, total\_symb);

if (pos == -1) {

symbol\_table[total\_symb++] = new Obj(tokens[k].substring(0, tokens[k].length() - 1), loc - 1);

bw.write("\t(S," + total\_symb + ")");

pos = total\_symb;

}

}

}

}

if (pos == -1) {

pos = *search*(tokens[k], REF);

if (pos != -1) {

bw.write("\t(RG," + (pos + 1) + ")");

} else {

if (tokens[k].matches("='\\d+'")) {

String s = tokens[k].substring(2, 3);

literal\_table[total\_ltr++] = new Obj(s, 0);

bw.write("\t(L," + total\_ltr + ")");

} else if (tokens[k].matches("\\d+") || tokens[k].matches("\\d+H") || tokens[k].matches("\\d+h") ||

tokens[k].matches("\\d+D") || tokens[k].matches("\\d+d")) {

bw.write("\t(C," + tokens[k] + ")");

temp1 = Integer.*parseInt*(tokens[k]);

} else {

pos = *search*(tokens[k], symbol\_table, total\_symb);

if (filladdr && pos != -1) {

symbol\_table[pos].addr = loc - 1;

filladdr = false;

} else if (pos == -1) {

symbol\_table[total\_symb++] = new Obj(tokens[k], 0);

bw.write("\t(S," + total\_symb + ")");

} else {

bw.write("\t(S," + pos + ")");

}

}

}

}

break;

}

}

}

System.***out***.println("\n\*\*symbol table");

System.***out***.println("\nsymbol\taddress");

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

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

pooltab[pooltab\_cnt] = new Pooltable(pooltab[pooltab\_cnt - 1].first + pooltab[pooltab\_cnt - 1].total\_literals, total\_ltr - pooltab[pooltab\_cnt - 1].first - 2);

pooltab\_cnt++;

System.***out***.println("\n\*\*pool table\*");

System.***out***.println("\npooltable literals");

for (int i = 0; i < pooltab\_cnt; i++)

System.***out***.println(pooltab[i].first + "\t" + pooltab[i].total\_literals);

System.***out***.println("\n\*literal table\*");

System.***out***.println("\nindex\tliteral\taddress");

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

if (literal\_table[i].addr == 0) {

literal\_table[i].addr = loc++;

}

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

}

System.***out***.println("\n\*optable");

System.***out***.println("\nmnemonic\topcode");

for (int i = 0; i < IS.length; i++)

System.***out***.println(IS[i] + "\t\t" + i);

} catch (Exception e) {

System.***out***.println("error while reading file");

e.printStackTrace();

}

try (BufferedReader br = new BufferedReader(new FileReader("D:\\TCOB22\\Assembler\\Output.txt"))) {

System.***out***.println("\n\*output1.txt\*\n");

while ((line = br.readLine()) != null) {

System.***out***.println(line);

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static int search(String token, String[] list) {

for (int i = 0; i < list.length; i++) {

if (token.equalsIgnoreCase(list[i])) {

return i;

}

}

return -1;

}

public static int search(String token, Obj[] list, int cnt) {

for (int i = 0; i < cnt; i++) {

if (token.equalsIgnoreCase(list[i].name)) {

return i;

}

}

return -1;

}

}

**Obj .java**

public class Obj {

String name;

int addr;

Obj(String nm,int address)

{

this.name=nm;

this.addr=address;

}

}

**Pooltable.java**

public class Pooltable {

int first,total\_literals;

public Pooltable(int f,int l)

{

this.first=f;

this.total\_literals=l;

}

}

Sample.txt

start 100

mover ax 05

mover bx 10

up: add ax bx

movem a='5'

mult ax a

origin up

ltorg

movem b='8'

movem c='8'

ltorg

movem b='7'

movem c='8'

ds a 02

dc b 10

ds c 09

next equ up

end

