package spos;

import java.io.\*;

import java.util.Iterator;

import java.util.LinkedHashMap;

public class passone

{

static LinkedHashMap<String, TableRow>SYMTAB;

public static void main(String[] args) throws Exception

{

SYMTAB = new LinkedHashMap<>();

INSTable lookup = new INSTable();

int LC=0,symIndex=0;

String line, code;

BufferedReader br= new BufferedReader(new FileReader("C:\\Users\\vaibh\\OneDrive\\Desktop\\spos.txt "));

BufferedWriter bw= new BufferedWriter(new FileWriter("C:\\Users\\vaibh\\OneDrive\\Desktop\\ic.txt"));

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

{

String parts[]=line.split("\\s+");

if(parts[1].equals("START"))

{

LC=Integer.parseInt(parts[2]);

code="(AD,01)(C,"+LC+")";

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

}

if(parts[1].equals("END"))

{

code="(AD,02)";

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

}

if(! parts[0].isEmpty()) {

if(SYMTAB.containsKey(parts[0]))

SYMTAB.put(parts[0],new TableRow(parts[0],LC,SYMTAB.get(parts[0]).getIndex()));

else

SYMTAB.put(parts[0], new TableRow(parts[0],LC,++symIndex));

}

if(parts[1].equals("DC")) {

parts[2]=parts[2].replace("'", "");

int constant = Integer.parseInt(parts[2]);

code = "(DC,01)(c,"+ constant+")";

LC++;

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

}

if(parts[1].equals("DS")) {

int size = Integer.parseInt(parts[2]);

code = "(DS,02)(c,"+ size+")";

LC=LC+size;

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

}

if (lookup.gettype(parts[1]).equals("IS")) {

int j = 2;

code = "(IS,"+ lookup.getcode(parts[1])+")\t";

String acode = " ";

while(j < parts.length) {

parts[j] = parts[j].replace(",", " ");

if (lookup.gettype(parts[j]).equals("RG")) {

acode = acode + lookup.getcode(parts[j])+"\t";

}

else {

if(SYMTAB.containsKey(parts[j])) {

int ind=SYMTAB.get(parts[j]).getIndex();

acode=acode+"(S,0"+ind+")\t";

}

else {

SYMTAB.put(parts[j],new TableRow(parts[j],-1,++symIndex));

int ind=SYMTAB.get(parts[j]).getIndex();

acode = acode+"(S,0"+ind+")";

}

}

j++;

}

code = code+acode;

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

LC++;

}

}

printSYMTAB();

br.close();

bw.close();

}

static void printSYMTAB() throws IOException {

BufferedWriter bw= new BufferedWriter(new FileWriter("SYMTAB"));

Iterator<String>itr=SYMTAB.keySet().iterator();

System.out.println("symbol table");

while(itr.hasNext()) {

String key = itr.next().toString();

TableRow val=SYMTAB.get(key);

System.out.println(val.getIndex()+"\t"+val.getsymbol()+"\t" + val.getAddress());

bw.write(val.getIndex()+"\t"+val.getsymbol()+"\t"+val.getAddress());

}

bw.close();

}

}

package spos;

import java.util.HashMap;

public class INSTable

{

HashMap<String, Integer>AD, IS, DL, RG;

public INSTable()

{

AD= new HashMap<>();

IS= new HashMap<>();

DL= new HashMap<>();

RG= new HashMap<>();

AD.put("START", 01);

AD.put("END", 02);

AD.put("ORIGIN", 03);

AD.put("EQU", 04);

AD.put("LTORG", 05);

IS.put("ADD",01);

IS.put("SUB",02);

IS.put("MULT",03);

IS.put("MOVER",04);

IS.put("MOVEM",05);

IS.put("COMP",06);

IS.put("BC",07);

IS.put("PRINT",10);

IS.put("STOP",00);

DL.put("DC", 01);

DL.put("Ds", 02);

RG.put("AREG", 01);

}

public String gettype(String s)

{

if(AD.containsKey(s))

return"AD";

else if (IS.containsKey(s))

return"IS";

else if (DL.containsKey(s))

return"DL";

else if (RG.containsKey(s))

return"RG";

return(" ");

}

public Integer getcode(String s)

{

if(AD.containsKey(s))

return AD.get(s);

else if (IS.containsKey(s))

return IS.get(s);

else if (DL.containsKey(s))

return DL.get(s);

else if (RG.containsKey(s))

return RG.get(s);

return(-1);

}

}

//code for instruction tablerow

package spos;

public class TableRow {

String symbol;

int add, index;

public TableRow(String s, int a, int i){

symbol = s;

add = a;

index = i;

}

public String getsymbol() {

return symbol;

}

public void setsymbol(String s) {

symbol = s;

}

public void setIndex(int i) {

index = i;

}

public int getIndex() {

return index;

}

public void setAddress(int a) {

add = a;

}

public int getAddress() {

return add;

}

}

START 200

A DS 2

L1 DC '5

ORIGIN L1+1

END

//ic

(AD,01)(C,200)

(DS,02)(c,2)

(DC,01)(c,5)

(AD,02)

//output

symbol table

1 A 200

2 L1 202