# import java.io.\*; import java.util.\*;

**import java.util.HashMap**;

*//passone.java*

# public class passone {

**static** LinkedHashMap<String, TableRow> SYMTAB;

**static** int symIndex = 0;

**public static** void main(String args[]) **throws** IOException { INSTable lookup = **new** INSTable();

SYMTAB = **new** LinkedHashMap<>(); int lc = 0;

String line, code;

BufferedReader br = **new** BufferedReader(**new** FileReader(*"C:/Users/DELL/OneDrive/Desktop/input.txt"*)); BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(*"C:/Users/DELL/OneDrive/Desktop/Intermediate"*)); **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], -1, ++symIndex));

}

}

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

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

int constant = Integer.parseInt(parts[2]); code = *"(DL,01) (C,"*+constant+*")"*;

lc++;

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

}

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

int size = Integer.parseInt(parts[2]); code = *"(DL,02) (C,"*+size+*")"*;

lc = lc + size; bw.write(code + *"\n"*);

}

**if**(lookup.gettype(parts[1]).equals(*"IS"*)) { int j = 2;

code = *"(IS,0"*+lookup.getcode(parts[1])+*") "*; String acode = *""*;

**while**(j < parts.length) {

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

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

acode = acode + *"(RG,0"* + lookup.getcode(parts[j])+ *") "*;

}

# 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; lc++;

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

}

}

br.close();

bw.close();

printSYMTAB();

}

**static** void printSYMTAB() **throws** IOException {

BufferedWriter bw = **new** BufferedWriter(**new** FileWriter(*"C:/Users/DELL/OneDrive/Desktop/symtab"*)); Set<String> allKeys = SYMTAB.keySet();

Iterator<String> itr = allKeys.iterator(); System.out.println(*"Symbol Table"*); **while**(itr.hasNext()) {

String key = itr.next().toString(); TableRow val = SYMTAB.get(key);

String s = val.getIndex() + *"\t "* + val.getsymbol()+ *"\t\t"* + val.getAddress() + *"\n"*; System.out.println(s);

bw.write(s);

}

bw.close();

}

}

*// TableRow.java*

**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** int getIndex() {

**return** index;

}

**public** void setIndex(int i) { index = i;

}

**public** void setAddress(int a) { add = a;

}

**public** int getAddress() {

**return** add;

}

}

*// INSTable.java*

# 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"*, 1);

IS.put(*"SUB"*, 2);

IS.put(*"MULT"*, 3);

IS.put(*"MOVER"*, 4);

IS.put(*"MOVEM"*, 5);

IS.put(*"COMP"*, 6);

IS.put(*"BC"*, 7);

IS.put(*"DIV"*, 8);

IS.put(*"READ"*, 9);

IS.put(*"PRINT"*, 10);

IS.put(*"STOP"*, 00);

DL.put(*"DC"*, 1);

DL.put(*"DS"*, 2);

RG.put(*"AREG"*, 1);

RG.put(*"BREG"*, 2);

RG.put(*"CREG"*, 3);

RG.put(*"DREG"*, 4);

}

**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;

}

}