import java.io.\*;  
public class Main {  
 public static void main(String args[]) throws NullPointerException, FileNotFoundException {  
 String REG[] = {"ax", "bx", "cx", "dx"};  
 String IS[] = {"mover", "movem", "add", "sub", "div", "mul"};  
 String DL[] = {"ds", "dc"};  
 obj[] literal\_table = new obj[10];  
 obj[] symb\_table = new obj[10];  
 obj[] optab = new obj[30];  
 pooltable[] pooltab = new pooltable[5];  
 String line;  
 try {  
 BufferedReader br = new BufferedReader(new FileReader("sample.txt"));  
 BufferedWriter bw = new BufferedWriter(new FileWriter("Output1.txt"));  
 boolean start = false;  
 boolean end = false, fill\_addr = 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) {  
 bw.write("\n" + loc);  
 ltorg = false;  
 loc++;  
 }  
 ltorg = fill\_addr = 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 + ")");  
 break;  
  
 case "origin":  
 pos = 3;  
 bw.write("\t(AD," + pos + ")");  
 pos = *search*(tokens[++k], symb\_table, total\_symb);  
 if (pos == -1) {  
 System.*out*.println("Error: Symbol not found in symbol table for origin.");  
 continue; // Skip this iteration if the symbol is not found  
 }  
 k++;  
 bw.write("\t(S," + (pos + 1) + ")");  
 loc = symb\_table[pos].addr;  
 break;  
  
 case "ltorg":  
 ltorg = true;  
 pos = 4;  
 for (temp = 0; temp < total\_ltr; temp++) {  
 if (literal\_table[temp].addr == 0) {  
 literal\_table[temp].addr = loc - 1;  
 bw.write("\t(DL,5)\t(C," + (temp + 1) + ")\n" + loc++);  
 }  
 }  
 if (pooltab\_cnt == 0) {  
 pooltab[pooltab\_cnt++] = new pooltable(0, temp);  
 } else {  
 pooltab[pooltab\_cnt] = new pooltable(pooltab[pooltab\_cnt - 1].first +  
 pooltab[pooltab\_cnt - 1].total\_literals, total\_ltr);  
 pooltab\_cnt++;  
 }  
 break;  
  
 case "equ":  
 pos = 5;  
 bw.write("\t(AD," + pos + ")");  
 String prev\_token = tokens[k - 1];  
 int pos1 = *search*(prev\_token, symb\_table, total\_symb);  
 pos = *search*(tokens[++k], symb\_table, total\_symb);  
 if (pos1 != -1 && pos != -1) {  
 symb\_table[pos1].addr = symb\_table[pos].addr;  
 bw.write("\t(S," + (pos + 1) + ")");  
 } else {  
 System.*out*.println("Error: Symbol not found for EQU.");  
 }  
 break;  
 }  
  
 if (pos == -1) {  
 pos = *search*(tokens[k], IS);  
 if (pos != -1) {  
 bw.write("\t(IS," + (pos + 1) + ")");  
 optab[optab\_cnt++] = new obj(tokens[k], pos);  
 } else {  
 pos = *search*(tokens[k], DL);  
 if (pos != -1) {  
 bw.write("\t(DL," + (pos + 1) + ")");  
 optab[optab\_cnt++] = new obj(tokens[k], pos);  
 fill\_addr = true;  
 } else if (tokens[k].matches("[a-zA-Z]+:")) {  
 pos = *search*(tokens[k], symb\_table, total\_symb);  
 if (pos == -1) {  
 symb\_table[total\_symb++] = new obj(tokens[k].substring(0, tokens[k].length() -  
 1), loc - 1);  
 bw.write("\t(S," + total\_symb + ")");  
 pos = total\_symb - 1; // Update position for new symbol  
 }  
 }  
 }  
 }  
  
 if (pos == -1) {  
 pos = *search*(tokens[k], REG);  
 if (pos != -1) {  
 bw.write("\t(RG " + (pos + 1) + ")");  
 } else {  
 if (tokens[k].matches("='\\d+'")) {  
 literal\_table[total\_ltr++] = new obj(tokens[k], 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] + ")");  
 } else {  
 pos = *search*(tokens[k], symb\_table, total\_symb);  
 if (fill\_addr && pos != -1) {  
 symb\_table[pos].addr = loc - 1;  
 fill\_addr = false;  
 } else if (pos == -1) {  
 symb\_table[total\_symb++] = new obj(tokens[k], 0);  
 bw.write("\t(S," + total\_symb + ")");  
 } else {  
 bw.write("\t(S," + pos + ")");  
 }  
 }  
 }  
 }  
 }  
 }  
 System.*out*.println("\n\*\*SYMBOL TABLE");  
 System.*out*.println("\nSYMBOL\tADDRESS");  
 for (int i = 0; i < total\_symb; i++) {  
 System.*out*.println(symb\_table[i].name + "\t" + symb\_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("\nPOOL\tTOTAL 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 + 1) + "\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 < optab\_cnt; i++) {  
 System.*out*.println(optab[i].name + "\t\t" + optab[i].addr);  
 }  
 br.close();  
 bw.close();  
 } catch (Exception e) {  
 System.*out*.println("Error while reading the file");  
 e.printStackTrace();  
 }  
 BufferedReader br = new BufferedReader(new FileReader("Output1.txt"));  
 System.*out*.println("\n\*\*Output1.txt\n");  
 try {  
 while ((line = br.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 br.close();  
 } 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  
class obj {  
 String name;  
 int addr;  
 obj(String nm, int address) {  
 this.name = nm;  
 this.addr = address;  
 }  
}  
// pooltable.java  
class pooltable {  
 int first;  
 int total\_literals;  
 pooltable(int f, int total) {  
 this.first = f;  
 this.total\_literals = total;  
 }  
}

//sample.txt

start 100  
mover ax 05  
movem a ='1'  
mul ax a  
ltorg  
movem b ='2'  
ltorg  
movem c ='3'  
ds a 02  
dc b 10  
ds c 01  
end

//Output

\*\*SYMBOL TABLE

SYMBOL ADDRESS

a 107

b 108

c 109

\*\*POOL TABLE

POOL TOTAL LITERALS

0 1

1 2

3 0

\*\*LITERAL TABLE

Index LITERAL ADDRESS

1 ='1' 103

2 ='2' 105

3 ='3' 111

\*\*OPTABLE

MNEMONIC OPCODE

mover 0

movem 1

mul 5

movem 1

movem 1

ds 0

dc 1

ds 0

\*\*Output1.txt

(AD,1) (C,100)

100 (IS,1) (RG 1) (C,05)

101 (IS,2) (S,1) (L,1)

102 (IS,6) (RG 1) (S,0)

103 (DL,5) (C,1)

104 (IS,2) (S,2) (L,2)

105 (DL,5) (C,2)

106 (IS,2) (S,3) (L,3)

107 (DL,1) (C,02)

108 (DL,2) (C,10)

109 (DL,1) (C,01)

110 (AD,2)