**1: PASS\_1**

**import** java.io.\*;

**import** java.util.\*;

**class** MnemonicTable {

**public** String mnemonic;

**public** String opcode;

**public** **int** num;

**public** MnemonicTable(String mnemonic, String opcode, **int** num) {

**this**.mnemonic = mnemonic;

**this**.opcode = opcode;

**this**.num = num;

}

}

**public** **class** Pass\_1 {

Map<String, MnemonicTable> is = **new** Hashtable<String, MnemonicTable>();

ArrayList<String> symtab = **new** ArrayList<>();

ArrayList<Integer> symaddr = **new** ArrayList<>();

ArrayList<String> littab = **new** ArrayList<>();

ArrayList<Integer> litaddr = **new** ArrayList<>();

ArrayList<Integer> pooltab = **new** ArrayList<>();

**int** LC = 0;

**public** **void** createIS() **throws** Exception {

Scanner input = **new** Scanner(System.***in***);

MnemonicTable m1 = **new** MnemonicTable("STOP", "00", 0);

is.put("STOP", m1);

MnemonicTable m2 = **new** MnemonicTable("ADD", "01", 0);

is.put("ADD", m2);

MnemonicTable m3 = **new** MnemonicTable("SUB", "02", 0);

is.put("SUB", m3);

MnemonicTable m4 = **new** MnemonicTable("MULT", "03", 0);

is.put("MULT", m4);

MnemonicTable m5 = **new** MnemonicTable("MOVER", "04", 0);

is.put("MOVER", m5);

MnemonicTable m6 = **new** MnemonicTable("MOVEM", "05", 0);

is.put("MOVEM", m6);

MnemonicTable m7 = **new** MnemonicTable("COMP", "06", 0);

is.put("COMP", m7);

MnemonicTable m8 = **new** MnemonicTable("BC", "07", 0);

is.put("BC", m8);

MnemonicTable m9 = **new** MnemonicTable("DIV", "08", 0);

is.put("DIV", m9);

MnemonicTable m10 = **new** MnemonicTable("READ", "09", 0);

is.put("READ", m10);

MnemonicTable m11 = **new** MnemonicTable("PRINT", "10", 0);

is.put("PRINT", m11);

input.close();

}

**public** **void** generateIC() **throws** Exception {

BufferedWriter wr = **new** BufferedWriter(**new** FileWriter("ic.txt"));

BufferedReader br = **new** BufferedReader(**new** FileReader("input.asm"));

String line = " ";

pooltab.add(0, 0);

wr.write("---------------------\n Intermediate Code\n---------------------\n");

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

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

**if** (split[0].length() > 0) {

// it is a label

**if** (!symtab.contains(split[0])) {

symtab.add(split[0]);

symaddr.add(LC);

} **else** {

**int** index = symtab.indexOf(split[0]);

symaddr.remove(index);

symaddr.add(index, LC);

}

}

**if** (split[1].equals("START")) {

LC = Integer.*parseInt*(split[2]);

wr.write("(AD,01)(C," + split[2] + ") \n");

} **else** **if** (split[1].equals("ORIGIN")) {

**if** (split[2].contains("+") || split[2].contains("-")) {

LC = getAddress(split[2]);

} **else** {

LC = symaddr.get(symtab.indexOf(split[2]));

}

} **else** **if** (split[1].equals("EQU")) {

**int** addr = 0;

**if** (split[2].contains("+") || split[2].contains("-")) {

addr = getAddress(split[2]);

} **else** {

addr = symaddr.get(symtab.indexOf(split[2]));

}

**if** (!symtab.contains(split[0])) {

symtab.add(split[0]);

symaddr.add(addr);

} **else** {

**int** index = symtab.indexOf(split[0]);

symaddr.remove(index);

symaddr.add(index, addr);

}

} **else** **if** (split[1].equals("LTORG") || split[1].equals("END")) {

**if** (litaddr.contains(0)) {

**for** (**int** i = pooltab.get(pooltab.size() - 1); i < littab.size(); i++) {

**if** (litaddr.get(i) == 0) {

litaddr.remove(i);

litaddr.add(i, LC);

LC++;

}

}

**if** (!split[1].equals("END")) {

pooltab.add(littab.size());

wr.write("\n(AD,05)\n");

} **else**

wr.write("(AD,04) \n");

}

} **else** **if** (split[1].contains("DS")) {

LC += Integer.*parseInt*(split[2]);

wr.write("(DL,01) (C," + split[2] + ") \n");

} **else** **if** (split[1].equals("DC")) {

LC++;

wr.write("\n(DL,02) (C," + split[2].replace("'", "").replace("'", "") + ") \n");

} **else** **if** (is.containsKey(split[1])) {

wr.write("(IS," + is.get(split[1]).opcode + ") ");

**if** (split.length > 2 && split[2] != **null**) {

String reg = split[2].replace(",", "");

**if** (reg.equals("AREG")) {

wr.write("(1) ");

} **else** **if** (reg.equals("BREG")) {

wr.write("(2) ");

} **else** **if** (reg.equals("CREG")) {

wr.write("(3) ");

} **else** **if** (reg.equals("DREG")) {

wr.write("(4) ");

} **else** {

**if** (symtab.contains(reg)) {

wr.write("(S," + symtab.indexOf(reg) + ")\n");

} **else** {

symtab.add(reg);

symaddr.add(0);

wr.write("(S," + symtab.indexOf(reg) + ") \n");

}

}

}

**if** (split.length > 3 && split[3] != **null**) {

**if** (split[3].contains("=")) {

String norm = split[3].replace("=", "").replace("'", "").replace("'", "");

**if** (!littab.contains(norm)) {

littab.add(norm);

litaddr.add(0);

wr.write("(L," + littab.indexOf(norm) + ")");

} **else** {

wr.write("L," + littab.indexOf(norm) + ")");

}

} **else** **if** (symtab.contains(split[3])) {

wr.write("(S," + symtab.indexOf(split[3]) + ") \n");

} **else** {

symtab.add(split[3]);

symaddr.add(0);

wr.write("(S," + symtab.indexOf(split[3]) + ") \n");

}

}

LC++;

}

}

wr.flush();

BufferedReader icr = **new** BufferedReader(**new** FileReader("ic.txt"));

**while** (icr.ready()) {

System.***out***.print((**char**) icr.read());

}

icr.close();

wr.close();

br.close();

BufferedWriter br1 = **new** BufferedWriter(**new** FileWriter("sym.txt"));

br1.write("-------------------\n Symbol Table\n-------------------\nSymbol Address\n");

**for** (**int** i = 0; i < symtab.size(); i++) {

br1.write(" " + symtab.get(i) + " " + symaddr.get(i) + "\n");

}

br1.flush();

BufferedReader br1r = **new** BufferedReader(**new** FileReader("sym.txt"));

**while** (br1r.ready()) {

System.***out***.print((**char**) br1r.read());

}

br1r.close();

br1.close();

BufferedWriter br2 = **new** BufferedWriter(**new** FileWriter("lit.txt"));

br2.write("-----------------------\n Literal Table\n-----------------------\nLiteral Address\n");

**for** (**int** i = 0; i < littab.size(); i++) {

br2.write("='" + littab.get(i) + "' " + litaddr.get(i) + "\n");

}

br2.flush();

BufferedReader br2r = **new** BufferedReader(**new** FileReader("lit.txt"));

**while** (br2r.ready()) {

System.***out***.print((**char**) br2r.read());

}

br2r.close();

br2.close();

BufferedWriter br3 = **new** BufferedWriter(**new** FileWriter("pool.txt"));

BufferedReader br3r = **new** BufferedReader(**new** FileReader("pool.txt"));

br3.write(

"-----------------------------\n Pool Table\n-----------------------------\nPool Index Literal Index\n");

**for** (**int** i = 0; i < pooltab.size(); i++) {

br3.write(" " + i + " " + pooltab.get(i) + "\n");

}

br3.flush();

**while** (br3r.ready()) {

System.***out***.print((**char**) br3r.read());

}

br3r.close();

br3.close();

}

**private** **int** getAddress(String string) {

**int** temp = 0;

**if** (string.contains("+")) {

String sp[] = string.split("\\+");

**int** ad = symaddr.get(symtab.indexOf(sp[0]));

temp = ad + Integer.*parseInt*(sp[1]);

} **else** **if** (string.contains("-")) {

String sp[] = string.split("\\-");

**int** ad = symaddr.get(symtab.indexOf(sp[0]));

temp = ad - Integer.*parseInt*(sp[1]);

}

**return** temp;

}

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

Pass\_1 p = **new** Pass\_1();

p.createIS();

p.generateIC();

}

}