import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.util.ArrayList;

class TableRow {

    String symbol;

    int address;

    int index;

    public TableRow(String symbol, int address) {

        super();

        this.symbol = symbol;

        this.address = address;

    }

    public TableRow(String symbol, int address, int index) {

        super();

        this.symbol = symbol;

        this.address = address;

        this.index = index;

    }

    public int getIndex() {

        return index;

    }

    public void setIndex(int index) {

        this.index = index;

    }

    public String getSymbol() {

        return symbol;

    }

    public void setSymbol(String symbol) {

        this.symbol = symbol;

    }

    public int getAddress() {

        return address;

    }

    public void setAddress(int address) {

        this.address = address;

    }

}

public class Pass2 {

    ArrayList<TableRow> SYMTAB, LITTAB;

    public Pass2() {

        SYMTAB = new ArrayList<>();

        LITTAB = new ArrayList<>();

    }

    public static void main(String[] args) {

        Pass2 pass2 = new Pass2();

        try {

            pass2.generateCode("IC.txt");

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    public void readtables() {

        BufferedReader br;

        String line;

        try {

            br = new BufferedReader(new FileReader("SYMTAB.txt"));

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

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

                SYMTAB.add(new TableRow(parts[1], Integer.parseInt(parts[2]), Integer.parseInt(parts[0])));

            }

            br.close();

            br = new BufferedReader(new FileReader("LITTAB.txt"));

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

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

                LITTAB.add(new TableRow(parts[1], Integer.parseInt(parts[2]), Integer.parseInt(parts[0])));

            }

            br.close();

        } catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

    public void generateCode(String filename) throws Exception {

        readtables();

        BufferedReader br = new BufferedReader(new FileReader(filename));

        BufferedWriter bw = new BufferedWriter(new FileWriter("PASS2.txt"));

        String line, code;

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

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

            // Skip AD or DL,02 lines

            if (parts[0].contains("AD") || parts[0].contains("DL,02")) {

                bw.write("\n");

                continue;

            }

            if (parts.length == 2) {

                if (parts[0].contains("DL")) {

                    parts[0] = parts[0].replaceAll("[^0-9]", "");

                    if (Integer.parseInt(parts[0]) == 1) {

                        int constant = Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));

                        code = "00\t0\t" + String.format("%03d", constant) + "\n";

                        bw.write(code);

                    }

                } else if (parts[0].contains("IS")) {

                    int opcode = Integer.parseInt(parts[0].replaceAll("[^0-9]", ""));

                    if (parts[1].contains("S")) {

                        int symIndex = Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));

                        if (symIndex > 0 && symIndex <= SYMTAB.size()) {

                            code = String.format("%02d", opcode) + "\t0\t"

                                    + String.format("%03d", SYMTAB.get(symIndex - 1).getAddress()) + "\n";

                            bw.write(code);

                        } else {

                            System.out.println("Invalid symbol index: " + symIndex);

                        }

                    } else if (parts[1].contains("L")) {

                        int litIndex = Integer.parseInt(parts[1].replaceAll("[^0-9]", ""));

                        if (litIndex > 0 && litIndex <= LITTAB.size()) {

                            code = String.format("%02d", opcode) + "\t0\t"

                                    + String.format("%03d", LITTAB.get(litIndex - 1).getAddress()) + "\n";

                            bw.write(code);

                        } else {

                            System.out.println("Invalid literal index: " + litIndex);

                        }

                    }

                }

            } // Handle IS instructions with no operands

            else if (parts.length == 1 && parts[0].contains("IS")) {

                int opcode = Integer.parseInt(parts[0].replaceAll("[^0-9]", ""));

                code = String.format("%02d", opcode) + "\t0\t" + String.format("%03d", 0) + "\n";

                bw.write(code);

            } // Handle all other IS instructions with operands

            else if (parts[0].contains("IS") && parts.length == 3) {

                int opcode = Integer.parseInt(parts[0].replaceAll("[^0-9]", ""));

                int regcode = Integer.parseInt(parts[1]);

                // Handle Symbol Table (S)

                if (parts[2].contains("S")) {

                    int symIndex = Integer.parseInt(parts[2].replaceAll("[^0-9]", ""));

                    if (symIndex > 0 && symIndex <= SYMTAB.size()) {

                        code = String.format("%02d", opcode) + "\t" + regcode + "\t"

                                + String.format("%03d", SYMTAB.get(symIndex - 1).getAddress()) + "\n";

                        bw.write(code);

                    } else {

                        System.out.println("Invalid symbol index: " + symIndex);

                    }

                } // Handle Literal Table (L)

                else if (parts[2].contains("L")) {

                    int litIndex = Integer.parseInt(parts[2].replaceAll("[^0-9]", ""));

                    if (litIndex > 0 && litIndex <= LITTAB.size()) {

                        code = String.format("%02d", opcode) + "\t" + regcode + "\t"

                                + String.format("%03d", LITTAB.get(litIndex - 1).getAddress()) + "\n";

                        bw.write(code);

                    } else {

                        System.out.println("Invalid literal index: " + litIndex);

                    }

                }

            }

        }

        bw.close();

        br.close();

    }

}

/\*

input.asm -

  START    100

A    DS    1

L1    MOVEM    AREG,    B

    ADD    AREG,    C

    MOVER    AREG,    ='12'

D    EQU    A+1

    LTORG

L2    PRINT    D

    ORIGIN     A-1

    MOVER    AREG,    ='5'

C     DC     '5'

    ORIGIN    L2+1

    STOP

B    DC    '19'

    END

IC.txt -

(AD,01) (C,100)

(DL,02) (C,1)

(IS,04) 1 (S,3)

(IS,01) 1 (S,4)

(IS,05) 1 (S,5)

(AD,05)

(IS,10) (S,5)

(AD,03) (S,1)-1

(DL,01) (C,5)

(AD,03) (S,6)+1

(IS,00)

(DL,01) (C,19)

(AD,02)

SYMTAB.txt -

1 A 100

2 L1 101

3 B 106

4 C 99

5 D 101

6 L2 104

LITTAB.txt -

 \*/