/\*

Name: Rohit Saini

RollNo: PC41

PRN: 1032200897

\*/

Code:

import java.io.File;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

class operator {

    String instruction;

    String statement\_class;

    int machine\_code;

    public void operator\_value(String instruction, String statement\_class, int machine\_code) {

        this.instruction = instruction;

        this.statement\_class = statement\_class;

        this.machine\_code = machine\_code;

    }

}

class register {

    String reg\_name;

    int machine\_code;

    public void register\_value(String reg\_name, int machine\_code) {

        this.reg\_name = reg\_name;

        this.machine\_code = machine\_code;

    }

}

class condition\_code {

    String condition;

    int machine\_code;

    public void condition\_code\_value(String condition, int machine\_code) {

        this.condition = condition;

        this.machine\_code = machine\_code;

    }

}

class symboltable {

    int symbol\_no;

    String symbol\_name;

    int address;

    int length;

    public void symboltable\_value(int symbol\_no,

            String symbol\_name,

            int address,

            int length) {

        this.symbol\_no = symbol\_no;

        this.symbol\_name = symbol\_name;

        this.address = address;

        this.length = length;

    }

}

class literal {

    int literal\_no;

    String literal\_name;

    int address;

    public void literal\_value(int literal\_no, String literal\_name, int address) {

        this.literal\_no = literal\_no;

        this.literal\_name = literal\_name;

        this.address = address;

    }

}

class pool\_tab {

}

public class lab1 {

    public static void main(String[] args) {

        try {

            File file = new File("C:\\Users\\rohit\\Documents\\GitHub\\sem\_7\\ssc\\Lab1\\OPTAB.txt");

            Scanner sc = new Scanner(file);

            operator[] OPTAB = new operator[18];

            String S = "";

            while (sc.hasNextLine()) {

                String temp = sc.nextLine();

                S += temp + ' ';

            }

            String[] data = S.split(" ");

            for (int j = 0; j < data.length; j += 3) {

                OPTAB[j / 3] = new operator();

                OPTAB[j / 3].operator\_value(data[j], data[j + 1], Integer.parseInt(data[j + 2]));

                // System.out.println(data[j] + " " + data[j + 1] + " " +

                // Integer.parseInt(data[j + 2]));

            }

            /\*

             \* for (int j = 0; j < data.length; j += 3) {

             \* System.out.print(OPTAB[j / 3].instruction + " " + OPTAB[j /

             \* 3].statement\_class + " "

             \* + OPTAB[j / 3].machine\_code);

             \* }

             \*/

            // code for register table

            file = new File("C:\\Users\\rohit\\Documents\\GitHub\\sem\_7\\ssc\\Lab1\\Register\_Table.txt");

            Scanner sc1 = new Scanner(file);

            register[] RTtable = new register[4];

            S = "";

            while (sc1.hasNextLine()) {

                String temp = sc1.nextLine();

                S += temp + ' ';

            }

            data = S.split(" ");

            for (int j = 0; j < data.length; j += 2) {

                RTtable[j / 2] = new register();

                RTtable[j / 2].register\_value(data[j], Integer.parseInt(data[j + 1]));

            }

            /\*

             \* for (int j = 0; j < data.length; j += 2) {

             \* System.out.print(RTtable[j / 2].reg\_name + " "

             \* + RTtable[j / 2].machine\_code);

             \* }

             \*/

            // code for condition code

            file = new File("C:\\Users\\rohit\\Documents\\GitHub\\sem\_7\\ssc\\Lab1\\Condition\_Code.txt");

            Scanner sc2 = new Scanner(file);

            condition\_code[] CCtable = new condition\_code[6];

            S = "";

            while (sc2.hasNextLine()) {

                String temp = sc2.nextLine();

                S += temp + ' ';

            }

            data = S.split(" ");

            for (int j = 0; j < data.length; j += 2) {

                CCtable[j / 2] = new condition\_code();

                CCtable[j / 2].condition\_code\_value(data[j], Integer.parseInt(data[j + 1]));

            }

            /\*

             \* for (int j = 0; j < data.length; j += 2) {

             \* System.out.print(CCtable[j / 2].reg\_name + " "

             \* + CCtable[j / 2].machine\_code);

             \* }

             \*/

            sc.close();

            sc1.close();

            sc2.close();

            // creating hashmap of all tables

            /\* Operator Table \*/

            HashMap<String, operator> OPTAB\_data = new HashMap<>();

            for (int i = 0; i < OPTAB.length; i++) {

                OPTAB\_data.put(OPTAB[i].instruction, OPTAB[i]);

            }

            /\* Register Table \*/

            HashMap<String, register> RTtable\_data = new HashMap<>();

            for (int i = 0; i < RTtable.length; i++) {

                RTtable\_data.put(RTtable[i].reg\_name, RTtable[i]);

            }

            /\* Condition Code Table \*/

            HashMap<String, condition\_code> CCtable\_data = new HashMap<>();

            for (int i = 0; i < CCtable.length; i++) {

                CCtable\_data.put(CCtable[i].condition, CCtable[i]);

            }

            /\* Symbol Table data to store \*/

            HashMap<String, symboltable> symbol\_table\_indexed = new HashMap<>();

            /\* Literal Table data \*/

            HashMap<String, literal> literal\_index = new HashMap<>();

            String input\_data = reader.read();

            int Location\_Counter = 0;

            int Base\_addr = 0;

            // System.out.println(input\_data);

            String[] output = string\_token.token(input\_data, "\n");

            for (int i = 0; i < output.length; i++) {

                String[] temp = string\_token.token(output[i], " ");

                if (temp[0].equals("LTROG")) {

                    try {

                        temp[0] = '(' + OPTAB\_data.get(temp[0]).statement\_class + ','

                                + OPTAB\_data.get(temp[0]).machine\_code

                                + ')';

                    } catch (NullPointerException e) {

                        temp[0] = temp[0];

                    }

                    output[i] = Location\_Counter++ + "  " + temp[0];

                }

                if (temp[0].equals("ORIGIN")) {

                    int k = 0;

                    try {

                        k = find\_address(symbol\_table\_indexed, temp[1]);

                        k -= Base\_addr;

                        // System.out.println("K\_s: " + k);

                        if (k == 0) {

                            throw new Exception();

                        }

                    } catch (Exception e) {

                        try {

                            k = find\_address\_l(literal\_index, temp[1]);

                            // System.out.println("K\_l: " + k);

                            k -= Base\_addr;

                            if (k == 0)

                                throw new Exception();

                        } catch (Exception e1) {

                            k = 0;

                        }

                    }

                    // System.out.println("K: " + k);

                    // System.out.println("temp: " + temp[0] + " " + temp[1] + " " + temp[2]);

                    Location\_Counter = Base\_addr + k - 1;

                    if (temp.length > 2)

                        Location\_Counter++;

                    // System.out.println("Base addr: " + Base\_addr);

                    // System.out.println("addr: " + k);

                }

                if (temp[0].equals("START")) {

                    try {

                        Location\_Counter = Integer.parseInt(temp[1]) - 1;

                        Base\_addr = Location\_Counter + 1;

                    } catch (Exception e) {

                        Location\_Counter = 0;

                    }

                }

                if (temp[0].equals("LTORG") || temp[0].equals("END")) {

                    set\_literal\_value(literal\_index, Location\_Counter);

                    Location\_Counter++;

                }

                // System.out.println(temp.length);

                if (temp[0] != "END" && temp.length == 1) {

                    // System.out.print(temp.toString());

                    try {

                        temp[0] = '(' + OPTAB\_data.get(temp[0]).statement\_class + ','

                                + OPTAB\_data.get(temp[0]).machine\_code

                                + ')';

                    } catch (NullPointerException e) {

                        temp[0] = temp[0];

                    }

                    output[i] = Location\_Counter++ + "  " + temp[0];

                } else if (temp.length == 2) {

                    if (temp[0].equals("LAST")) {

                        try {

                            Integer.parseInt(temp[0]);

                            temp[0] = " (C," + temp[0] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                            if (k != -1)

                                temp[0] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[0]);

                                temp[0] = " (L," + k + ")";

                            }

                        }

                        try {

                            temp[1] = '(' + OPTAB\_data.get(temp[1]).statement\_class + ','

                                    + OPTAB\_data.get(temp[1]).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            temp[1] = temp[1];

                        }

                        output[i] = Location\_Counter++ + "  " + temp[1];

                    } else {

                        String t = temp[0];

                        try {

                            temp[0] = '(' + OPTAB\_data.get(temp[0]).statement\_class + ','

                                    + OPTAB\_data.get(temp[0]).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            temp[0] = temp[0];

                        }

                        try {

                            Integer.parseInt(temp[1]);

                            temp[1] = " (C," + temp[1] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            // System.out.println("inside origin"+t);

                            if (t.equals("ORIGIN")) {

                                int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[1]);

                                if (k != -1)

                                    temp[1] = " (S," + k + ")";

                                else {

                                    k = index\_in\_literaltable(literal\_index, -1, temp[1]);

                                    temp[1] = " (L," + k + ")";

                                }

                                temp[1] = " (S," + k + ")";

                            } else {

                                int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[1]);

                                if (k != -1) {

                                    temp[1] = " (S," + k + ")";

                                } else {

                                    k = index\_in\_literaltable(literal\_index, Location\_Counter, temp[1]);

                                    temp[1] = " (L," + k + ")";

                                }

                            }

                        }

                        output[i] = Location\_Counter++ + "  " + temp[0] + temp[1];

                    }

                } else if (temp.length == 3) {

                    if (temp[1].equals("EQU"))

                        Location\_Counter--;

                    String d\_temp = temp[2];

                    if (temp[0].equals("ORIGIN")) {

                        int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[1]);

                        if (k != -1) {

                            temp[1] = " (S," + k + ")";

                        } else {

                            k = index\_in\_literaltable(literal\_index, -1, temp[1]);

                            temp[1] = " (L," + k + ")";

                        }

                        // temp[1] = " (S," + Location\_Counter + 1 + ")";

                    }

                    if (temp[0].equals("BACK")) {

                        try {

                            Integer.parseInt(temp[0]);

                            temp[0] = " (C," + temp[0] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                            if (k != -1)

                                temp[0] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[0]);

                                temp[0] = " (L," + k + ")";

                            }

                        }

                        try {

                            temp[1] = '(' + OPTAB\_data.get(temp[1]).statement\_class + ','

                                    + OPTAB\_data.get(temp[1]).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            try {

                                temp[1] = " " + RTtable\_data.get(temp[1]).machine\_code + " ";

                            } catch (NullPointerException e1) {

                                temp[1] = " " + temp[1];

                            }

                        }

                        try {

                            Integer.parseInt(temp[2]);

                            temp[2] = " (C," + temp[2] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[2]);

                            if (k != -1)

                                temp[2] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[2]);

                                temp[2] = " (L," + k + ")";

                            }

                        }

                        output[i] = Location\_Counter + "  " + temp[1] + temp[2];

                        try {

                            Location\_Counter += Integer.parseInt(d\_temp);

                        } catch (Exception e) {

                            Location\_Counter++;

                        }

                    } else if (temp[0].length() > 1 && symbol\_table\_indexed.containsKey(temp[0])) {

                        index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                        try {

                            temp[1] = '(' + OPTAB\_data.get(temp[1]).statement\_class + ',' +

                                    OPTAB\_data.get(temp[1]).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            temp[1] = " " + temp[1];

                        }

                        // System.out.print(temp[1] + "-");

                        try {

                            Integer.parseInt(temp[2]);

                            temp[2] = " (C," + temp[2] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[2]);

                            if (k != -1)

                                temp[2] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[2]);

                                temp[2] = " (L," + k + ")";

                            }

                        }

                        output[i] = Location\_Counter + "  " + temp[1] + temp[2];

                        try {

                            Location\_Counter += Integer.parseInt(d\_temp);

                        } catch (Exception e) {

                            Location\_Counter++;

                        }

                    } else if (temp[0].length() > 1) {

                        String d\_t = temp[0];

                        try {

                            temp[0] = '(' + OPTAB\_data.get(d\_t).statement\_class + ',' +

                                    OPTAB\_data.get(d\_t).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            try {

                                temp[0] = " " + RTtable\_data.get(d\_t).machine\_code + " ";

                            } catch (NullPointerException e1) {

                                int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, d\_t);

                                if (k != -1)

                                    temp[0] = " (S" + k + ")";

                                else {

                                    k = index\_in\_literaltable(literal\_index, -1, d\_t);

                                    temp[0] = " (L," + k + ")";

                                }

                            }

                        }

                        // System.out.print(temp[1] + "-");

                        try {

                            d\_t = temp[1];

                            temp[1] = '(' + OPTAB\_data.get(d\_t).statement\_class + ',' +

                                    OPTAB\_data.get(d\_t).machine\_code

                                    + ')';

                        } catch (NullPointerException e1) {

                            try {

                                temp[1] = " " + RTtable\_data.get(d\_t).machine\_code + " ";

                            } catch (Exception e) {

                                try {

                                    temp[1] = " " + CCtable\_data.get(d\_t).machine\_code + " ";

                                } catch (NullPointerException e2) {

                                    temp[1] = " " + temp[1];

                                }

                            }

                        }

                        try {

                            d\_t = temp[2];

                            Integer.parseInt(temp[2]);

                            temp[2] = " (C," + d\_t + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, d\_t);

                            if (k != -1)

                                temp[2] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, d\_t);

                                temp[2] = " (L," + k + ")";

                            }

                        }

                        output[i] = Location\_Counter + "  " + temp[0] + temp[1] + temp[2];

                        try {

                            Location\_Counter += Integer.parseInt(d\_temp);

                        } catch (Exception e) {

                            Location\_Counter++;

                        }

                    } else if (temp[0].length() == 1) {

                        try {

                            Integer.parseInt(temp[0]);

                            temp[0] = " (C," + temp[0] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                            if (k != -1)

                                temp[0] = " (S," + k + ")";

                            else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[0]);

                                temp[0] = " (L," + k + ")";

                            }

                        }

                        try {

                            temp[1] = '(' + OPTAB\_data.get(temp[1]).statement\_class + ',' +

                                    OPTAB\_data.get(temp[1]).machine\_code

                                    + ')';

                        } catch (NullPointerException e) {

                            temp[1] = temp[1];

                        }

                        // System.out.print(temp[1] + "-");

                        try {

                            Integer.parseInt(temp[2]);

                            temp[2] = " (C," + temp[2] + ")";

                        } catch (NumberFormatException e) {

                            // Error;

                            int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[2]);

                            if (k != -1) {

                                temp[2] = " (S," + k + ")";

                                try {

                                    Location\_Counter += k;

                                } catch (Exception e2) {

                                    Location\_Counter++;

                                }

                            } else {

                                k = index\_in\_literaltable(literal\_index, -1, temp[2]);

                                temp[2] = " (L," + k + ")";

                            }

                        }

                        output[i] = Location\_Counter + "  " + temp[1] + temp[2];

                        try {

                            if (temp[1].equals("(DL,1)")) {

                                throw new Exception();

                            }

                            Location\_Counter += Integer.parseInt(d\_temp);

                        } catch (Exception e) {

                            Location\_Counter++;

                        }

                    }

                } else if (temp.length == 4) {

                    // if(temp)

                    int k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                    int k1 = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[0]);

                    k = k > k1 ? k : k1;

                    if (k != -1)

                        temp[0] = " (S," + k + ")";

                    else {

                        k = index\_in\_literaltable(literal\_index, -1, temp[0]);

                        temp[0] = " (L," + k + ")";

                    }

                    try {

                        temp[1] = '(' + OPTAB\_data.get(temp[1]).statement\_class + ','

                                + OPTAB\_data.get(temp[1]).machine\_code

                                + ')';

                        temp[2] = " " + RTtable\_data.get(temp[2]).machine\_code + " ";

                    } catch (NullPointerException e) {

                    }

                    try {

                        Integer.parseInt(temp[3]);

                        temp[3] = " (C," + temp[3] + ")";

                    } catch (NumberFormatException e) {

                        // Error;

                        k = index\_in\_symboltable(symbol\_table\_indexed, Location\_Counter, temp[3]);

                        if (k != -1)

                            temp[3] = " (S," + k + ")";

                        else {

                            k = index\_in\_literaltable(literal\_index, -1, temp[3]);

                            temp[3] = " (L," + k + ")";

                        }

                    }

                    output[i] = Location\_Counter++ + "  " + temp[1] + temp[2] + temp[3];

                }

                // System.out.println(output[i]);

            }

            /\* Printing the pass-1 \*/

            System.out.println("\nOuput After Pass1: ");

            System.out.println("Intermediate Code: ");

            for (int i = 0; i < output.length; i++) {

                System.out.println(output[i]);

            }

            System.out.println("\nSymbol Table: ");

            for (String key : symbol\_table\_indexed.keySet()) {

                symboltable temp = new symboltable();

                temp = symbol\_table\_indexed.get(key);

                System.out

                        .println(temp.symbol\_no + " " + temp.symbol\_name + " " + temp.address + " " + temp.length);

            }

            System.out.println("\nLiteral Table: ");

            for (String key : literal\_index.keySet()) {

                literal temp = new literal();

                temp = literal\_index.get(key);

                System.out

                        .println(temp.literal\_no + " " + temp.literal\_name + " " + temp.address);

            }

            Pass2(output, OPTAB\_data, RTtable\_data, CCtable\_data, symbol\_table\_indexed, literal\_index);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    public static String get\_symbol\_name(HashMap<String, symboltable> symbol\_table\_indexed, String s) {

        String name = "Not-Found";

        for (String key : symbol\_table\_indexed.keySet()) {

            symboltable symbol = symbol\_table\_indexed.get(key);

            if (symbol.symbol\_no == Integer.parseInt(s)) {

                return "" + symbol.address;

            }

        }

        return name;

    }

    public static String get\_literal\_name(HashMap<String, literal> literal\_index, String s) {

        String name = "Not-Found";

        for (String key : literal\_index.keySet()) {

            literal l = literal\_index.get(key);

            if (l.literal\_no == Integer.parseInt(s)) {

                return "" + l.address;

            }

        }

        return name;

    }

    public static void Pass2(String[] intermediateCode, HashMap<String, operator> OPTAB\_data,

            HashMap<String, register> RTtable\_data, HashMap<String, condition\_code> CCtable\_data,

            HashMap<String, symboltable> symbol\_table\_indexed, HashMap<String, literal> literal\_index) {

        System.out.println("\nOutput After Pass 2: ");

        System.out.println("Machine Code Instructions: ");

        String machine\_code = "";

        for (int i = 1; i < intermediateCode.length; i++) {

            // check if string in intermediateCode[i] contain AD if yes the i++

            if (intermediateCode[i].contains("AD,3") || intermediateCode[i].contains("AD,2")) {

                continue;

            }

            String line = intermediateCode[i];

            String temp = "";

            String[] tokens = string\_token.token(line, " ");

            for (String l : tokens) {

                // System.out.println(l);

                if(l.contains("DL,1") || l.contains("DL,2")) {

                    break;

                }

                try {

                    int k = Integer.parseInt(l);

                    temp = "" + k;

                } catch (NumberFormatException e) {

                    l = l.substring(1, l.length() - 1);

                    String[] l\_set = l.split(",");

                    if (l\_set[0].equals("S")) {

                        l = "" + get\_symbol\_name(symbol\_table\_indexed, l\_set[1]);

                    } else if (l\_set[0].equals("L")) {

                        l = "" + get\_literal\_name(literal\_index, l\_set[1]);

                    } else {

                        temp = "0";

                        try {

                            l = l\_set[1];

                        } catch (Exception e1) {

                        }

                    }

                    temp = "" + l;

                }

                machine\_code += temp + " ";

            }

            machine\_code += "\n";

        }

        System.out.println(machine\_code);

    }

    public static int find\_address\_l(HashMap<String, literal> table\_indexed, String S) {

        for (String str : table\_indexed.keySet()) {

            if (str.equals(S)) {

                return table\_indexed.get(S).address;

                // If the string is found, return true

            }

        }

        int ans = 0;

        try {

            ans = Integer.parseInt(S);

        } catch (NumberFormatException e) {

            ans = 0;

        }

        return ans;

    }

    public static int find\_address(HashMap<String, symboltable> symbol\_table\_indexed, String S) {

        for (String str : symbol\_table\_indexed.keySet()) {

            if (str.equals(S)) {

                return symbol\_table\_indexed.get(S).address;

                // If the string is found, return true

            }

        }

        int ans = 0;

        try {

            ans = Integer.parseInt(S);

        } catch (NumberFormatException e) {

            ans = 0;

        }

        return ans;

    }

    public static int index\_in\_symboltable(HashMap<String, symboltable> symbol\_table\_indexed, int Location\_Counter,

            String S) {

        int index = 1;

        if (S.charAt(0) == '=') {

            return -1;

        }

        Map<String, symboltable> linkedHashMap = new LinkedHashMap<>(symbol\_table\_indexed);

        symboltable st = new symboltable();

        // Print the data in serial order

        for (Map.Entry<String, symboltable> entry : linkedHashMap.entrySet()) {

            if (entry.getKey().equals(S)) {

                st = symbol\_table\_indexed.get(S);

                if (Location\_Counter != -1 && !S.equals("AGAIN"))

                    st.address = Location\_Counter;

                symbol\_table\_indexed.put(S, st);

                // System.out

                // .println(st.symbol\_no + " " + st.symbol\_name + " " + st.address + " " +

                // st.length);

                return st.symbol\_no;

            }

            index++;

            /\* insert data into the symbol table \*/

        }

        st.symboltable\_value(index, S, Location\_Counter, 1);

        symbol\_table\_indexed.put(S, st);

        // System.out

        // .println(st.symbol\_no + " " + st.symbol\_name + " " + st.address + " " +

        // st.length);

        return symbol\_table\_indexed.size();

    }

    public static void set\_literal\_value(HashMap<String, literal> literal\_indexed, int Location\_Count) {

        int index = 0;

        for (String t : literal\_indexed.keySet()) {

            literal l = literal\_indexed.get(t);

            if (l.address == -1)

                l.address = Location\_Count+index++;

        }

    }

    public static int index\_in\_literaltable(HashMap<String, literal> literal\_indexed, int Location\_Counter, String S) {

        int index = 1;

        Map<String, literal> linkedHashMap = new LinkedHashMap<>(literal\_indexed);

        literal lt = new literal();

        // Print the data in serial order

        for (Map.Entry<String, literal> entry : linkedHashMap.entrySet()) {

            if (entry.getKey().equals(S)) {

                lt = literal\_indexed.get(S);

                lt.address = Location\_Counter;

                literal\_indexed.put("" + index, lt);

                return lt.literal\_no;

            }

            index++;

            /\* insert data into the symbol table \*/

        }

        lt.literal\_value(index, S, Location\_Counter);

        literal\_indexed.put("" + index, lt);

        return literal\_indexed.size();

    }

}

Input:

START 100

MOVER AREG ='5'

MOVEM BREG A

LOOP MOVER CREG B

LTORG

BC ANY NEXT

ADD CREG ='2'

ORIGIN LOOP +12

A DC 3

NEXT DS 5

END

Output:

