```
/*
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;
    }
}
```

```java
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_in
dexed, 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).mach
ine_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).mach
ine_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]);
```

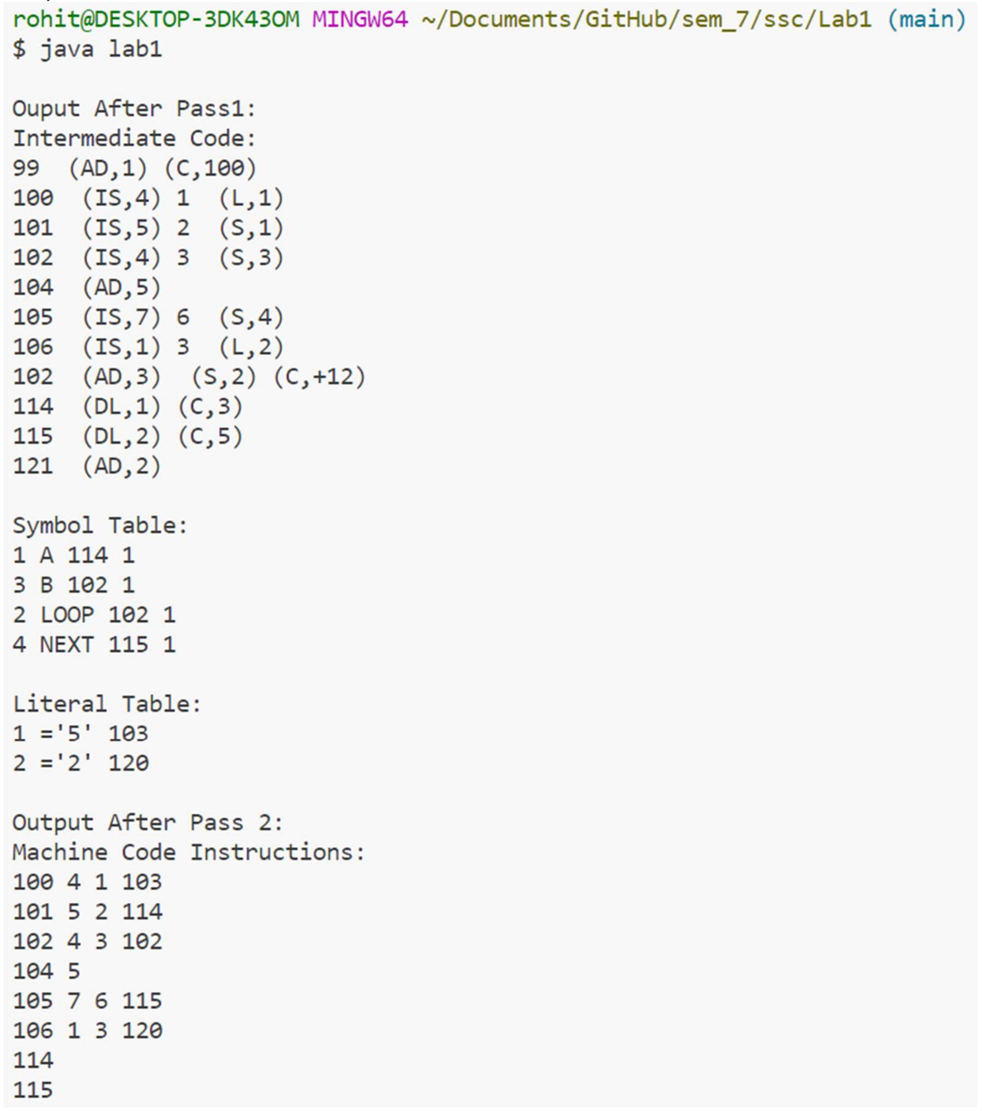
```
                                        if (k != -1) {
                                            temp[2] = " (S," + k + ")";
                                            try {
                                                Location_Counter += k;
                                            } catch (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:
```



```
rohit@DESKTOP-3DK43OM MINGW64 ~/Documents/GitHub/sem_7/ssc/Lab1 (main)
$ java lab1

Ouput After Pass1:
Intermediate Code:
99  (AD,1) (C,100)
100  (IS,4) 1  (L,1)
101  (IS,5) 2  (S,1)
102  (IS,4) 3  (S,3)
104  (AD,5)
105  (IS,7) 6  (S,4)
106  (IS,1) 3  (L,2)
102  (AD,3)  (S,2) (C,+12)
114  (DL,1) (C,3)
115  (DL,2) (C,5)
121  (AD,2)

Symbol Table:
1 A 114 1
3 B 102 1
2 LOOP 102 1
4 NEXT 115 1

Literal Table:
1 ='5' 103
2 ='2' 120

Output After Pass 2:
Machine Code Instructions:
100 4 1 103
101 5 2 114
102 4 3 102
104 5
105 7 6 115
106 1 3 120
114
115
```