Om Shete
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
TE C32
2103163

Experiment No: 8

Aim: Write a program to implement Multi Pass Macroprocessor.

Theory:

- A multi-pass compiler is a type of compiler that processes the source code or abstract syntax tree of a program multiple time.
- In multi-pass compiler, we divide phases into two passes as :-

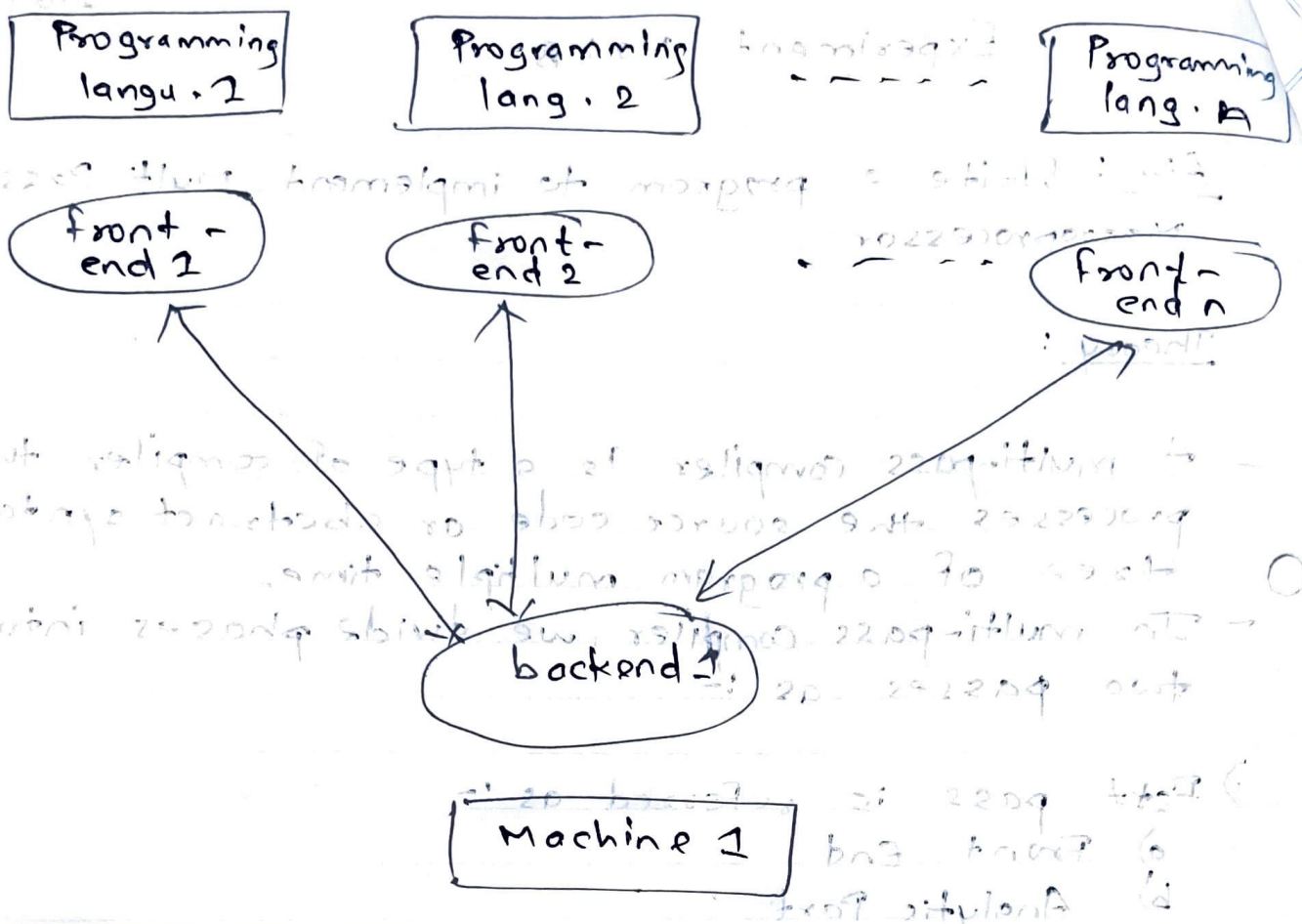
1) First pass is referred as :-
   a) Front End
   b) Analytic Part
   c) Platform Independent

2) Second pass is referred as :-
   a) Back End
   b) Synthesis Part
   c) Platform Dependent

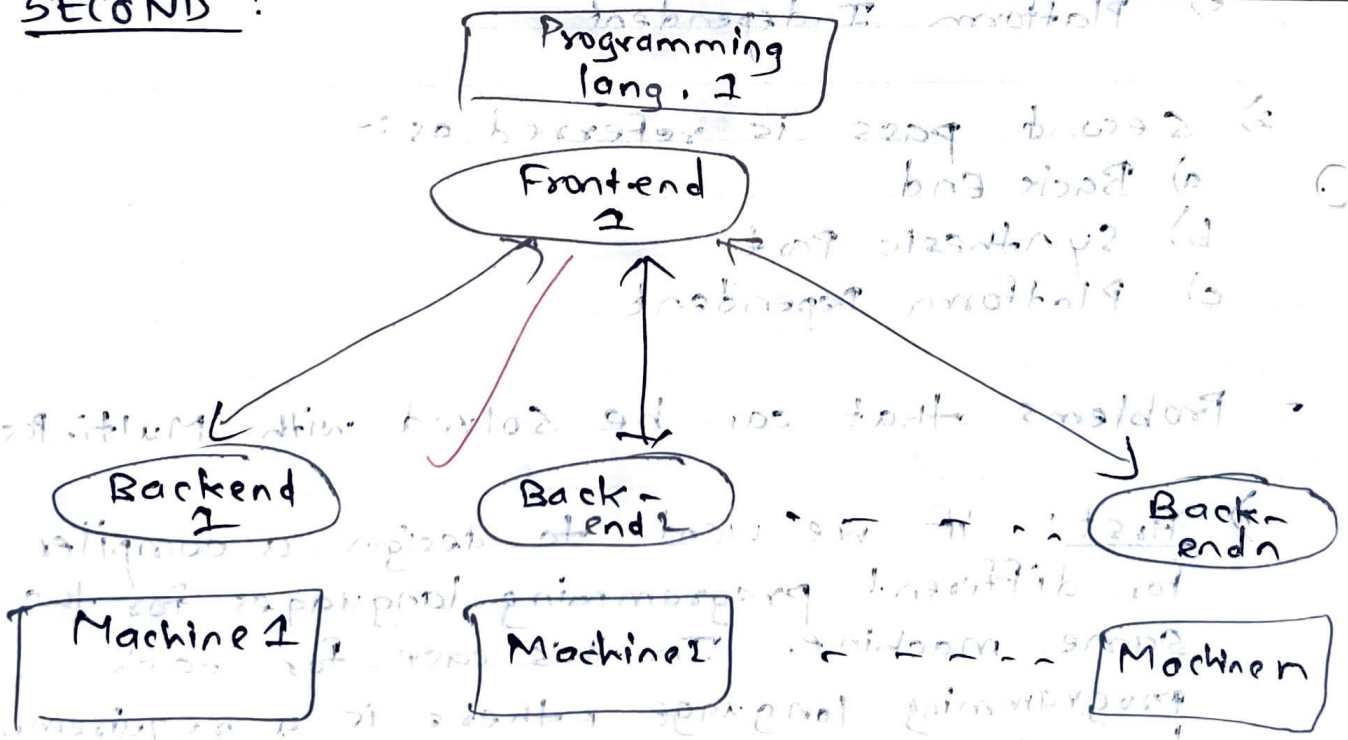
- Problems that can be solved with Multi-Pass:

1) **First:** If we want to design a compiler for different programming languages for the same machine. In this case for each programming language, there is a requirement to make the Front end for each of them and only Backend pass

FIRST :



SECOND :



2) Second : If we want to design a compiler for the same programming language for different machines / system. In this case, we make different Back End for different Machines and make only one Front End for same programming language.

- Conclusion : In conclusion, the choice between a single pass and a two-pass compiler depends on specific requirements and trade-offs. Multi-pass compilers offers greater flexibility for different programming languages and machines systems but come at cost of additional processing.

10/04/2024.

Code:

```
import java.util.*;
import java.io.*;

class twopassmacro {
    static String mnt[][] = new String[5][3];

    public static void main(String args[]) {
        pass1();

        System.out.println("Macro Table(MNT)");
        display(mnt, mntc, 3);

        System.out.println("Argument Array(ALA) for Pass1");
        display(ala, alac, 2);

        System.out.println("Macronition Table(MDT)");
        display(mdt, mdtc, 1);

        pass2();

        System.out.println("Argument Array(ALA) for Pass2");
        display(ala, alac, 2);

        System.out.println("Note: All are displayed here whereas
intermediate pass1 output & expanded pass2 output is stored in files");
    }

    static void pass1() {
        int index = 0, i;
        String s, prev = "", substring;
        try {
            BufferedReader inp = new BufferedReader(new
FileReader("input.txt"));
            File op = new File("pass1_output.txt");
            if (!op.exists())
                op.createNewFile();

            BufferedWriter output = new BufferedWriter(new
FileWriter(op.getAbsoluteFile()));
            while ((s = inp.readLine()) != null) {
                if (s.equalsIgnoreCase("MACRO")) {
```

```
                    prev = s;
                    for (; !(s =
inp.readLine()).equalsIgnoreCase("MEND"); mdtc++, prev = s) {
                        if (prev.equalsIgnoreCase("MACRO")) {
                            StringTokenizer st = new
StringTokenizer(s);
                            String str[] = new
String[st.countTokens()];
                            for (i = 0; i < str.length; i++)
                                str[i] = st.nextToken();

                            mnt[mntc][0] = (mntc + 1) + "";
                            mnt[mntc++][2] = (++mdtc) + "";

                            st = new StringTokenizer(str[1], ",");
                            for (i = 0; i < string.length; i++) {
                                string[i] = st.nextToken();
                                ala[alac][0] = alac + "";
                                if (index != -1)
                                    ala[alac++][1] =
string[i].substring(0, index);
                                else
                                    ala[alac++][1] = string[i];
                            }
                        } else {
                            index = s.indexOf("&");
                            substring = s.substring(index);
                            for (i = 0; i < alac; i++)
                                if (ala[i][1].equals(substring))
                                    s = s.replaceAll(substring, "#" +
ala[i][0]);
                        }

                        mdt[mdtc - 1][0] = s;
                    }

                    mdt[mdtc - 1][0] = s;
                } else {
                    output.write(s);
                    output.newLine();
                }
            }
```

```
            output.close();
        } catch (FileNotFoundException ex) {
            System.out.println("Unable to find file ");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    static void pass2() {
        int alap = 0, index, mdtp, flag = 0, i, j;
        String s, temp;
        try {
            BufferedReader inp = new BufferedReader(new
FileReader("pass1_output.txt"));
            File op = new File("pass2_output.txt");
            if (!op.exists())
                op.createNewFile();

            BufferedWriter output = new BufferedWriter(new
FileWriter(op.getAbsoluteFile()));
            for (; (s = inp.readLine()) != null; flag = 0) {
                StringTokenizer st = new StringTokenizer(s);
                String str[] = new String[st.countTokens()];
                for (i = 0; i < str.length; i++)
                    str[i] = st.nextToken();
                for (j = 0; j < mntc; j++) {
                    if (str[0].equals(mnt[j][1])) {
                        mdtp = Integer.parseInt(mnt[j][2]);
                        st = new StringTokenizer(str[1], ",");
                        String arg[] = new String[st.countTokens()];
                        for (i = 0; i < arg.length; i++) {
                            arg[i] = st.nextToken();
                            ala[alap++][1] = arg[i];
                        }

                        for (i = mdtp;
!(mdt[i][0].equalsIgnoreCase("MEND")); i++) {
                            index = mdt[i][0].indexOf("#");
                            temp = mdt[i][0].substring(0, index);
                            temp += ala[Integer.parseInt("" +
mdt[i][0].charAt(index + 1))][1];
                            output.write(temp);
                            output.newLine();
```

```
                        }

                        flag = 1;
                    }
                }

                if (flag == 0) {
                    output.write(s);
                    output.newLine();
                }
            }

            output.close();
        } catch (FileNotFoundException ex) {
            System.out.println("Unable to find file ");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    static void display(String a[][], int n, int m) {
        int i, j;
        for (i = 0; i < n; i++) {
            for (j = 0; j < m; j++)
                System.out.print(a[i][j] + " ");
            System.out.println();
        }
    }
}
```

Output:

```
C:\Program Files (x86)\Java\jdk1.6.0_2\bin>java twopassmacro
Macro Name Table(MNT)
1  INCR1  1
2  INCR2  5
Argument List Array(ALA) for Pass1
0  &FIRST
1  &SECOND
2  &ARG1
3  &ARG2
Macro Definition Table(MDT)
INCR1        &FIRST,&SECOND=DATA9
A            1,#0
L            2,#1
MEND
INCR2        &ARG1,&ARG2=DATA5
L            3,#2
ST           4,#3
MEND
Argument List Array(ALA) for Pass2
0  DATA1
1  DATA2
2  DATA3
3  DATA4
```