T.Y.B.Tech (CSE)

System Software and Compilers(SSC)

Lab Assignment No – 3

Name: Aniruddha Shende

Roll number: PD-05

Batch: D1

Panel: D

Name:- Aniruddha Arun Shende
Roll no:- PD05
Batch :- D1
Panel: D

MIT-WPU MIT WORLD PEACE UNIVERSITY | PUNE

Subject :- SSC

Lab Assignment No - 3

Assignment Title:- Design of Pass I of Two Pass Macroprocessor

Aim :- Design suitable data structure & implement pass I of Two Pass Macroprocessor.

Objective:- Design suitable data structure & implement pass I of Two Pass Macroprocessor Input should consist of a one macro definition & one macro call & few assembly language instructions.

Theory:-

Write about :-

1. Description about the macroprocessor.

Ans :- A Macro-processor is a program that copies a stream of text from one place to another, making a systematic set of replacements as it does so. Macro processors are often embedded in other programs, such as assemblers & compilers.

Macroprocessor designs are not directly related to computer architecture on which it runs.
Macroprocessor is generally involved with definition, invocation & expansion.

2. Data structures required for 2 pass assembler macro

Ans:2. (a) Macro Definition Table (MDT) :
- It is created by Pass I & is used to store macro definition.
- It has 80 bytes / entry
- MEND indicates end of macro definition so it is stored in MDT.

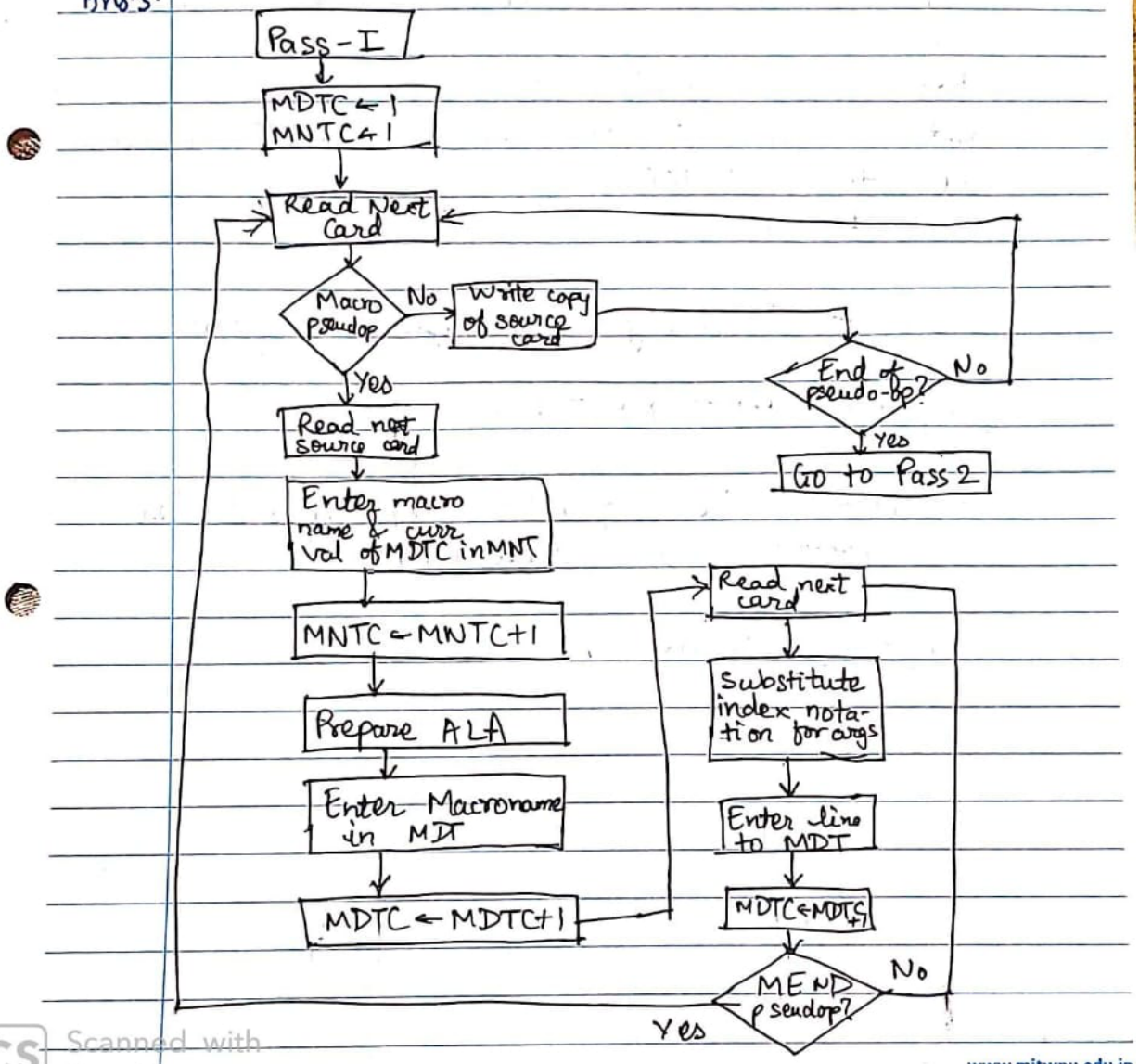
(b) Macro Name Table (MNT)
- Used to store corresponding name of Macro & its MDT index & is created in Pass I.
- MDTC stores last count from MDT table.
- MNTC stores the no. of macros defined in program.

(c) Argument List Array (ALA)
- It is used for Association of Dummy Arguments & Actual Arguments.
- It keeps track of Dummy parameters,

when the macro is being defined & maintain the actual arguments when expanding the call.

3. Flowchart for Pass I

Ans:3.



MIT-WPU MIT WORLD PEACE UNIVERSITY

4. Algorithm for Pass-I.

Ans:4. Algorithm for Pass-I

1. Initialize MNTC=0 & MDTC=0
2. Scanning of all MACRO definitions one by one: If MACRO found then for each MACRO perform:
3. MNTC= MNTC+1 & enter name of Macro in MNT.
4. For every model statement until MEND is not encountered do MDTC = MDTC+1 in definition of MACRO.
5. Generated ALP.

Input :- Assembly Language Program.

Output :- MDT, MNT, ALA, Output file of Pass-I

Conclusion:- The function of Pass I in macroprocessor is studied.

Platform:- Linux, Windows (Java)

INPUT :

src > com > lab1 > input1.txt

```
1   MACRO
2   INCR &ARG1 &ARG2
3   ADD AREG &ARG1
4   ADD BREG &ARG2
5   MEND
6   MACRO
7   DECR &ARG3 &ARG4
8   SUB AREG &ARG3
9   SUB BREG &ARG4
10  MEND
11  START
12  MOVER AREG S1
13  MOVER BREG S1
14  INCR D1 D2
15  DECR D3 D4
16  S1 DC 5
17  D1 DC 2
18  D2 DC 3
19  D3 DC 4
20  D4 DC 5
21  END
```

Java Code:

1. **Main.java file**

```
package com.lab1;

public class Main {
    public static void main(String[] args) {
        Pass1_MACRO.printPass1_MACRO();
    }
}
```

2. **MDT Table file**

```
package com.lab1;

import java.util.LinkedHashMap;

public class MDTtable {
    private static int location_counter = 0;
    private static LinkedHashMap<String, String> MDT =
new LinkedHashMap<String, String>();

    public static int getLocation_counter() {
        return location_counter;
    }

    public static void add(String instructions) {
        location_counter += 1;
        MDT.put(Integer.toString(location_counter),
instructions);
    }

    public static void printMDT() {
```

```
        for (String key : MDT.keySet()) {
            System.out.println(key + " " +
MDT.get(key));
        }
    }
}
```

2. MNT Table file

```
package com.lab1;

import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.Set;

public class MNTtable {
    private static LinkedHashMap<String, String> MNT =
new LinkedHashMap<String, String>();
    private static Set<String> all_macros = new
HashSet<String>();

    public static void add_to_MNT(String macro_name,
int index) {
        all_macros.add(macro_name);
        MNT.put(macro_name,Integer.toString(index));
    }

    public static void printMNT() {
        int mnt_index = 0;
        for (String key : MNT.keySet()) {
            mnt_index++;
            System.out.println(mnt_index + " " + key +
" " + MNT.get(key));
        }
```

```
    }

    public static boolean isMacro_present(String
macro_name) {
        return all_macros.contains(macro_name);
    }
}
```

3. ALA Table file

```
package com.lab1;

import java.util.LinkedHashMap;

public class ALAtable {
    private static LinkedHashMap<String, String> ALA =
new LinkedHashMap<String, String>();
    private static int index = 0;

    public static void add(String arguments) {
        index++;
        ALA.put(Integer.toString(index), arguments);
    }

    public static void printALA() {
        for (String key : ALA.keySet()) {
            System.out.println(key + " " +
ALA.get(key));
        }
    }

    public static LinkedHashMap<String, String>
getALA() {
        return ALA;
```

```
    }

}
```

3. Pass1_MACRO.java file

```
package com.lab1;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Pass1_MACRO {
    private static LinkedHashMap<String,
ArrayList<String>> macro_with_their_params = new
LinkedHashMap<String, ArrayList<String>>();

    public static void printPass1_MACRO() {
        System.out.println("Pass1 MACRO : \n");
        File file = new
File("src/com/lab1/input1.txt");
        Scanner sc = null;
        try {
            sc = new Scanner(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        while (sc.hasNextLine()) {
```

```
            String line = sc.nextLine();
            StringTokenizer st = new
StringTokenizer(line, " ");
            String opcode = st.nextToken();
            if (opcode.equals("MACRO")) {
                String str = sc.nextLine();
                MDTtable.add(str);
                int counter = 0;
                while (!str.equals("MEND")) {
                    if (counter == 0) {
                        String[] list1 = str.split("
");
                        MNTtable.add_to_MNT(list1[0],
MDTtable.getLocation_counter());
                        ArrayList<String> list2 = new
ArrayList<String>();
                        for (int i = 1; i <
list1.length; i++) {
                            list2.add(list1[i]);
                            ALAtable.add(list1[i]);
                        }

macro_with_their_params.put(list1[0], list2);
                    }
                    counter++;
                    str = sc.nextLine();
                    MDTtable.add(str);
                }
            } else {
                FileWriter abc;
                try {
                    abc = new
FileWriter("src/com/lab1/output_file.txt");
```

```
                    BufferedWriter writer = new
BufferedWriter(abc);
                    writer.write(line);
                    writer.newLine();
                    while (sc.hasNextLine()) {
                        String line1 = sc.nextLine();
                        writer.write(line1);
                        writer.newLine();
                    }
                    writer.close();
                } catch (IOException except) {
                    except.printStackTrace();
                }
            }
        }
        System.out.println("\n\nMDT Table\n");
        MDTtable.printMDT();

        System.out.println("\n\nMNT Table\n");
        MNTtable.printMNT();

        System.out.println("\n\nALA Table\n");
        ALAtable.printALA();

        sc.close();

    }

    public static ArrayList<String>
getMacro_with_their_params(String macro_name) {
        return
macro_with_their_params.get(macro_name);
    }
```

}

Output of the program:

```
Pass1 MACRO :


MDT Table :

Index   Instructions
-----------------------------------------
1       INCR &ARG1 &ARG2
2       ADD AREG &ARG1
3       ADD BREG &ARG2
4       MEND
5       DECR &ARG3 &ARG4
6       SUB AREG &ARG3
7       SUB BREG &ARG4
8       MEND


MNT Table :

MNT Index       Macro Name      MDT Index
-----------------------------------------
1               INCR            1
2               DECR            5


ALA Table :

Index   Formal Arguments
-----------------------------------------
1       &ARG1
2       &ARG2
3       &ARG3
4       &ARG4
```

Output File generated as a result of Pass 1 :

```
 1   START
 2   MOVER AREG S1
 3   MOVER BREG S1
 4   INCR D1 D2
 5   DECR D3 D4
 6   S1 DC 5
 7   D1 DC 2
 8   D2 DC 3
 9   D3 DC 4
10   D4 DC 5
11   END
```