Name - Vasu Kalariya
Roll - PE29
Sub - SSC

Lab Assignment - 3

Assignment Title: Design of Pass 1 of Two Pass Macroprocessor

Aim: Design suitable data structure & implement pass 1 of Two Pass Macroprocessor.

Objective:
Design suitable data structure & implement pass 1 of Two Pass Macroprocessor. Input should consist of a one macro definition and one macro call and few assembly language instructions.

Theory:

→ Description about the Macroprocessor.
A macro instruction is the notational convenience for the programmer. For every ocurrence of a macro the whole macro body or macro block of statement gets expanded in the main source code. Thus Macro instructions make writing code more convenient. Macro represents a group of commonly used statements in the source programming language. Macro Processor replaces each macro instruction with the expansion of macros. A macro consist of macro name, set of formal parameters and a body of code. Macro name with a set of actual parameters, is replaced by some code, generated from macro body. This is called macro expansion.

→ Data Structure required for 2 pass macroprocessor.
There are three main data structure involved in ~~our~~ macroprocessor

- MDT ~~TAB~~ : The macro defination themselves are stored in a defination table (MDT ~~TAB~~), which contains the macro protolype.
- MNT ~~TAB~~ : The ~~nam~~ macro names are entered into MNT ~~TAB~~ which serves as an index of MDTTAB
- ALA : It is an argument table which is used during the expansion of macro invocations

→ Flowchart for Pass 1
{At the end}.

Algorithm for Pass 1:

1 Initialization of counter for MDT & MNT
2 Read next instruction (and divide it into its various field as label. mnemonic)
3 if opcode = MACRO goto Step 5
else go to step 4
4 (a) Write copy of instruction to output of Pass 1
(b) Check whether opcode = END or not
(c) If OPCODE "" END goto Step 2
(d) if OPCODE = END goto Pass 2 i.e End of this algo. for Pass 1
5 (a) Read Next instruction
(b) Enter <macro-name, MDTC> into MNT at MNTC
MDTC is
MDTC is entered in MNT at available row
(c) MNTC - MNTC + 1
(d) Prepare Argument List Array
(e) Enter macroname instruction in MDT at MDTC
(i) MDTC - MDTC + 1.
6 (a) Read next card
(b) Substitute Index notations for dummy arguments
(c) Enter this instruction into MDT.

(d) MDTC - MDTC +1

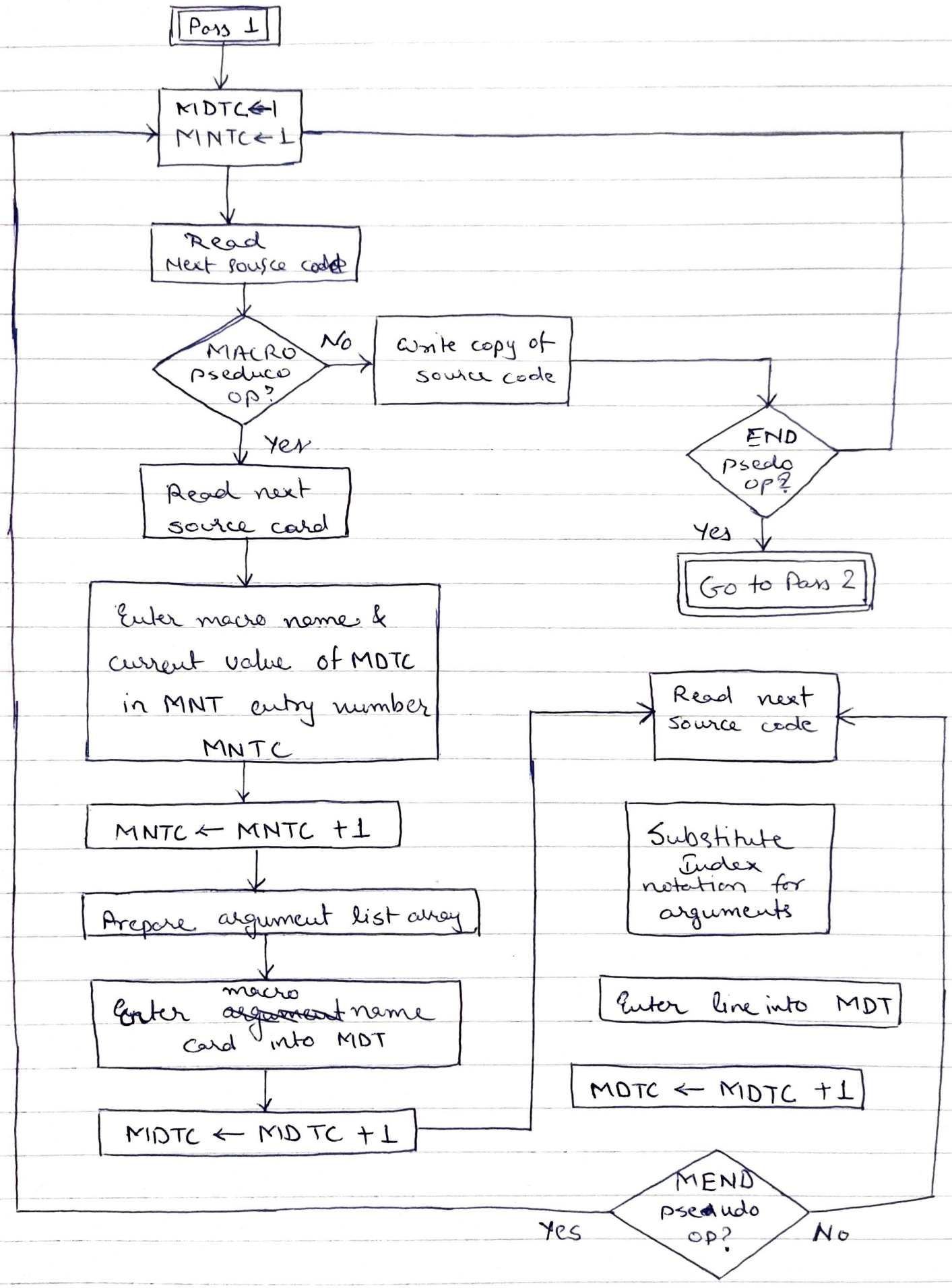
(e) If OPCODE of this instruction is MEND then goto Step 2.
else goto Step 6.a

Input: Assembly Language Program

Output:
1) Program without Macro Defination (Pass - 1)
2) Macro Defination Table (MDT)
Index MDT - Instruction
3) Macro Name Table (MNT)
Index Macro Name MDT - index
4) Argument List Array (ALA)
Index Dummy Argument.

Platform: Linux (Java)

Conclusion: ~~This~~ The function of Pass 1 in assembler is studied along errors comming in each pass.


Pass 1
MDTC ← 1
MNTC ← 1
Read Next source code
MACRO pseduo op?
No
Write copy of source code
END pseudo op?
Yes
Go to Pass 2
Yes
Read next source card
Enter macro name & current value of MDTC in MNT entry number MNTC
MNTC ← MNTC + 1
Prepare argument list array
Enter macro name card into MDT
MDTC ← MDTC + 1
Read next source code
Substitute Index notation for arguments
Enter line into MDT
MDTC ← MDTC + 1
MEND pseudo op?
Yes
No

```java
/*
Name : Vasu Kalariya
Roll : PE29
Sub  : SSC ( Macro Pass 1)
*/

import java.io.*;
import java.util.Hashtable;

public class MacroPass1 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = null;
        FileReader fr = null;

        Hashtable<String, String> ala = new Hashtable<String, String>();

        boolean macrodef = false;                // flag for macro definition
        boolean alaParameter = false;            // flag for macro inserting ala parameters name
        boolean endMacro = false;                // flag for end of macro definition
        boolean macroName = false;               // flag for name of macro definition

        int mdtptr = 1, mntptr = 1, alaptr = 1; // pointers for MDT, MNT, ALA

        String inputfilename = "F:\\T9\\SSC\\Assi 3\\src\\Input.txt";       //Input File
        fr = new FileReader(inputfilename);
        br = new BufferedReader(fr);

        String f1 = "F:\\T9\\SSC\\Assi 3\\src\\MDT.txt";                // MDT file
        FileWriter fw1 = new FileWriter(f1);
        BufferedWriter bw1 = new BufferedWriter(fw1);

        String f2 = "F:\\T9\\SSC\\Assi 3\\src\\MNT.txt";                // MNT file
        FileWriter fw2 = new FileWriter(f2);
        BufferedWriter bw2 = new BufferedWriter(fw2);
```

```java

        String f3 = "F:\\T9\\SSC\\Assi 3\\src\\ALA.txt";                // ALA file
        FileWriter fw3 = new FileWriter(f3);
        BufferedWriter bw3 = new BufferedWriter(fw3);

        String f4 = "F:\\T9\\SSC\\Assi 3\\src\\Output.txt";            // OUTPUT file without macro expansion
        FileWriter fw4 = new FileWriter(f4);
        BufferedWriter bw4 = new BufferedWriter(fw4);

        try {


            String sCurrentLine, s0, s1, s2;
            while ((sCurrentLine = br.readLine()) != null) {
                s0 = sCurrentLine.split(" |\\,")[0];

                if (s0.equals("MACRO")) {               // check for Macro definition
                    macrodef = true;
                    alaParameter = true;
                    endMacro = true;
                }
                else if (macrodef == true || alaParameter == true || endMacro == true) {

                    if (macrodef == true) {             //  writing in MNT
                        bw2.write(mntptr + "\t" + s0 + "\t" + mdtptr + "\n");
                        macrodef = false;
                        macroName = true;
                        mntptr++;
                    }
                    if (alaParameter == true) {         //  writing in ALA
                        for (int i = 1; i < (sCurrentLine.split(" |\\,").length); i++) {
                            s1 = sCurrentLine.split(" |\\,")[i];
                            String temp;
```

```
                                bw3.write("#" + alaptr + "\t" + s1 + "\n");
                                temp = "#" + alaptr;
                                ala.put(s1, temp);
                                alaptr++;
                            }
                            alaParameter = false;
                        }
                        if (endMacro == true){              //  writing in MDT
                            bw1.write(mdtptr+" ");
                            for (int i = 0; i < (sCurrentLine.split(" |\\,").length); i++){

                                s2 = sCurrentLine.split(" |\\,")[i];

                                if (ala.containsKey(s2) && macroName == false){     // replacing Parameters with # number
                                    bw1.write(ala.get(s2)+" ");
                                }
                                else {
                                    bw1.write(s2+" ");
                                }
                            }
                            bw1.write("\n");
                            mdtptr++;
                            macroName = false;
                        }
                    }

                    else {
                        bw4.write(sCurrentLine+"\n");
                    }

                    if (s0.equals("MEND")){
                        endMacro = false;
                    }
                }

            } catch (FileNotFoundException fileNotFoundException) {
```

```
            fileNotFoundException.printStackTrace();
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }

        bw1.close();
        bw2.close();
        bw3.close();
        bw4.close();

    }
}

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Assi 3 [F:\T9\SSC\Assi 3] - Input.txt

Assi 3 > src > Input.txt

MacroPass1

MacroPass1.java | Input.txt | MNT.txt | MDT.txt | ALA.txt | Output.txt

```
MACRO
ADDS &L1 &L2
ADD AREG &L1
ADD BREG &L2
MEND
MACRO
SUBS &L3
SUB CREG &L3
MEND
START
MOVER AREG S1
MOVER BREG S1
ADDS D1 D2
SUBS D1
S1 DC 3
D1 DC 3
D2 DC 4
END
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Assi 3 [F:\T9\SSC\Assi 3] - MNT.txt

Assi 3 〉 src 〉 MNT.txt

MacroPass1.java | Input.txt | MNT.txt | MDT.txt | ALA.txt | Output.txt

```
1    ADDS    1
2    SUBS    5
3
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Assi 3 [F:\T9\SSC\Assi 3] - MDT.txt

Assi 3 〉 src 〉 MDT.txt

MacroPass1.java | Input.txt | MNT.txt | MDT.txt | ALA.txt | Output.txt

```
1 ADDS &L1 &L2
2 ADD AREG #1
3 ADD BREG #2
4 MEND
5 SUBS &L3
6 SUB CREG #3
7 MEND
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Assi 3 [F:\T9\SSC\Assi 3] - ALA.txt

Assi 3 〉 src 〉 ALA.txt

MacroPass1.java | Input.txt | MNT.txt | MDT.txt | ALA.txt | Output.txt

Project

```
#1  &L1
#2  &L2
#3  &L3
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Assi 3 [F:\T9\SSC\Assi 3] - Output.txt

Assi 3 〉 src 〉 Output.txt

MacroPass1.java | Input.txt | MNT.txt | MDT.txt | ALA.txt | Output.txt

```
START
MOVER AREG S1
MOVER BREG S1
ADDS D1 D2
SUBS D1
S1 DC 3
D1 DC 3
D2 DC 4
END
```