

System Programming and Operating Systems Lab

ASSIGNMENT 7

Name: Shrirang Mhalgi

Roll No: 322008

Batch: B1

1 Date of Completion:

20/02/2019

2 Aim:

Implement Pass-2 of two pass assembler for pseudo-machine in Java using object oriented features.

3 Objectives:

To implement Pass-2 of two pass assembler for pseudo-machine in Java.

4 Theory:

Assembly Language is a low-level programming language which is used for a computer or other programmable devices. There is a very strong correspondence between the assembly language and the architectures machine code instructions.

Every assembly language is system specific i.e, it is unique for every individual system that it operates on which is exactly opposite to the high-level languages where the language is portable from device to device. but require interpreting or compiling.

An assembler is a translator, that translates an assembler program into a conventional machine language program. Basically, the assembler goes through the program one line at a time and generates machine code for that instruction. Let us see two pass assemblers

It requires all data symbols to be defined prior to being used. A two-pass assembler solves this dilemma by devoting one pass to exclusively resolve all (data/label) forward references and then generate object code with no hassles in the next pass. If a data symbol depends on another and this another depends on yet another, the assembler resolved this recursively. If I try explaining even that in this post, the post will become too big.

Data structure required in pass 2 assembler
Intermediate file Output file MNT MDT MNT
Pointer MDT Pointer Line Pointer Actual v/s positional parameter list array

5 Algorithm:

1. Assumptions
 - a]Macro call within macro definition
 - b]Early expansion method
 - c]Any depth of nesting is permitted
 - d]Static allocation is used
2. Step1:
 - a]Set MNT pointer and MDT pointer to start of respective tables
 - b]Open source file in read mode and target file in write mode
 - c]Set macro definition flag (d) = 0 and macro expansion flag (e) = 0
 - d]Initialize formal v/s positional parameter list & actual v/s positional parameter list
 - e]Set error-flag OFF
3. Step 2:
 - a]Read a line of text from the source file till EOF() or visit a keyword END.
 - b]If EOF() or a keyword END has occurred then GOTO step 4, else GOTO step 3
4. Step 3:
 - a]Case d = 0, e = 0
 - b]Case d = 1, e = 0
 - c]Case d = 0, e = 1
 - d]Case d = 1, e = 1

A]

 - a]Read a line
 - b]If line contains MACRO DEFINITION then d=1 and goto case d=1, e=0
 - c]If line contains MACRO CALL then e=1 and goto case d=0, e=1
 - d]If line contains neither MACRO DEFINITION nor MACRO CALL then write into target file
 - e]Goto step 2

B]

 - a]Validate macro signature in MNT
 - b]If duplicate macro,
 - display err-massage
 - set error-flag = ON
 - c]Goto step 2
 - d]Otherwise
 - make entries in MNT
 - create formal v/s positional parameter list
 - make entries in MDT
 - e]If MACRO CALL,
 - set e = 1

f]goto Case d = 1, e =1
g]If MEND, set d = 0
h]Goto step 2

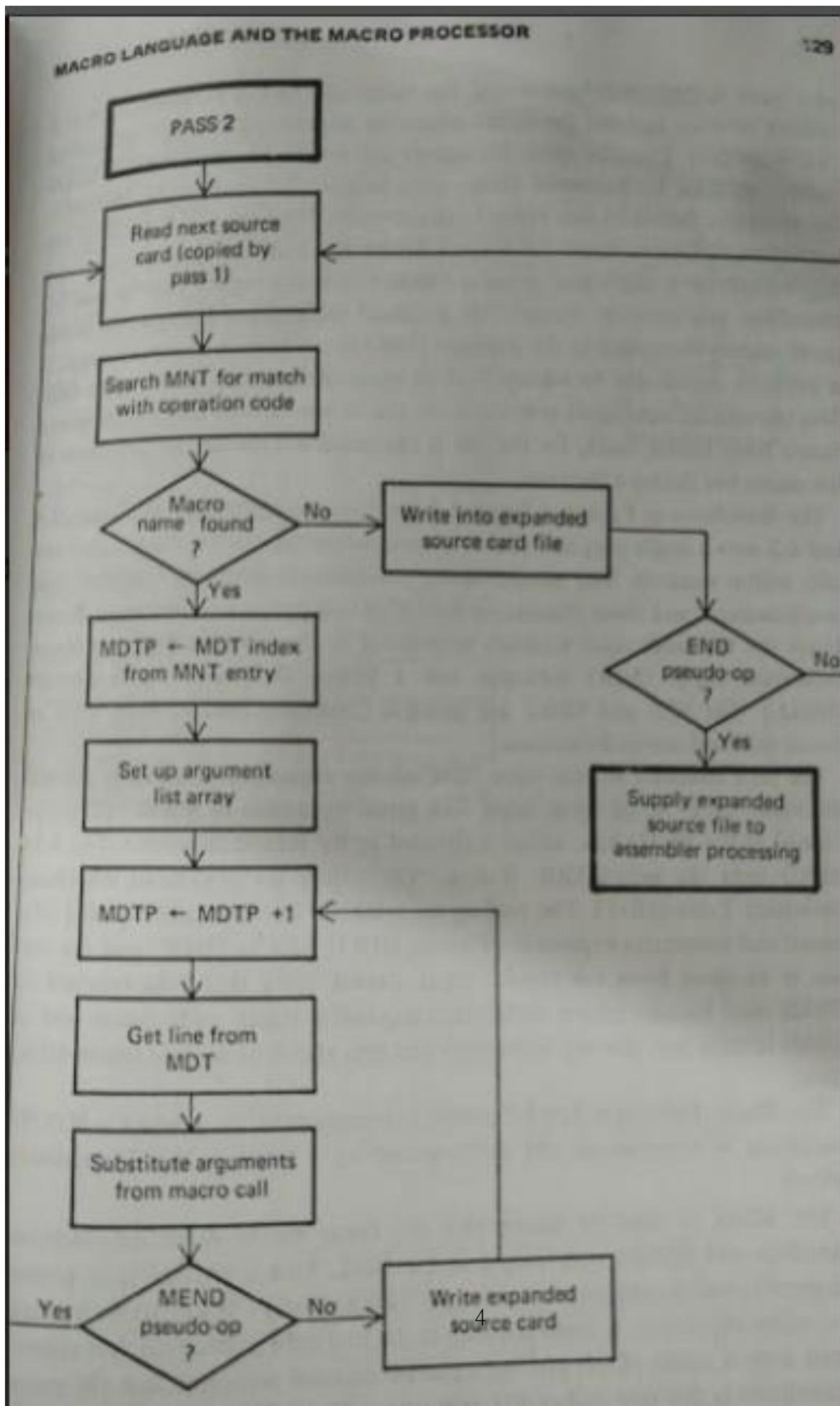
C]
a]If normal MACRO CALL, search in MNT
b]If NOT FOUND
-disp-error Call to macro without definition
-set error-flag = ON
c]Else
-Prepare actual v/s positional parameter list
-Read lines from MDT till MEND
-Replace positional parameter with actual
-Write the line into the target file
d]When MEND occurs
-Set e = 0
-Goto step 2

D]
a]Macro call inside a macro definition
b]Validate macro signature in MNT
c]If macro name not found,
-Display err-massage call to a macro which is not defined
-Set error-flag = ON
-Set d = 1 and e = 0
-Goto step 2
d]Else
e]Save current position of source file
f]Expand macro call with the help of MDT
g]Prepare formal v/s positional parameter list
h]Prepare actual v/s positional parameter list
i]If MEND
-Set e = 0
j]Goto case d = 1, e =0

5. Step 4:
If (d = 0 AND e = 0) AND error-flag = OFF
display Successful macro processing
Else
Display-error Unsuccessful macro processing

6. End

6 Flowchart:



7 Code:

```
package macropass2n ;

import java .io .*;
import java .lang .String ;
import java .util .Scanner ;

public class MacroPass2N {
    public static void main ( String [] args ) throws Exception
    {
        File f=new File(" program.txt " );
        Scanner s=new Scanner ( f);

        String [] tokens ,macro _args ;           // Array for storing tokens gen
        boolean is_macro=false ;
        while ( s .hasNextLine ( ))
        {
            tokens=s .nextLine ().split(" ");           //split the line on space

            for (int i=0;i<tokens .length ;i++)
            {
                is_macro=check if macro (tokens [ i ]);
                if (is_macro==true )
                {
                    macro _args=tokens [ i+1].split(" , " );
                    expand ( tokens [ i ],macro _args );
                    //System.out.println(tokens[i]+" "+macro _args[0]+" "+m

                }
                else
                {
                    if (!tokens[i].contains(" , "))
                        System .out .p r i n t l n ( tokens [ i ]);
                }
            }

            System .out .p r i n t l n ( );
        }

        s .close ();
    }
}
```

```

static void expand(String macro_name,String [] macro_args )throws File No
{
    File f1=new File(" contents.txt ");
    Scanner s1=new Scanner (f1 );

    String [] tokens_arr ;
    while(!s1.nextLine().contains("MDT:" ))
        s1.nextLine ();
    while (!s1.nextLine ().contains (macro_name))
        continue ;

    String def=" ";
    while(!def.contains("MEND" ))
    {
        def=s1.nextLine ();

        def= def.replace("#0",macro_args[0]);
        def=def.replace("#1",macro_args[1]);
        if(!def.contains("MEND" ))
            System.out.println(def );
    }
}

```

```

static boolean check-if-macro(String token) throws FileNotFoundException
{
    File f1=new File(" contents.txt ");
    Scanner s1=new Scanner (f1 );

    String [] tokens_arr ;

    while(!s1.nextLine().contains("MNT:" ))
        s1.nextLine ();
    while (s1.hasNextLine ())
    {
        tokens_arr=s1.nextLine().split(" ");
        for(int i=0;i<tokens_arr.length;i++)
        {
            if(tokens_arr[i].equals (token ))
            {
                s1.close();

                return true ;
            }
        }
    }
}

```

```

    }

}

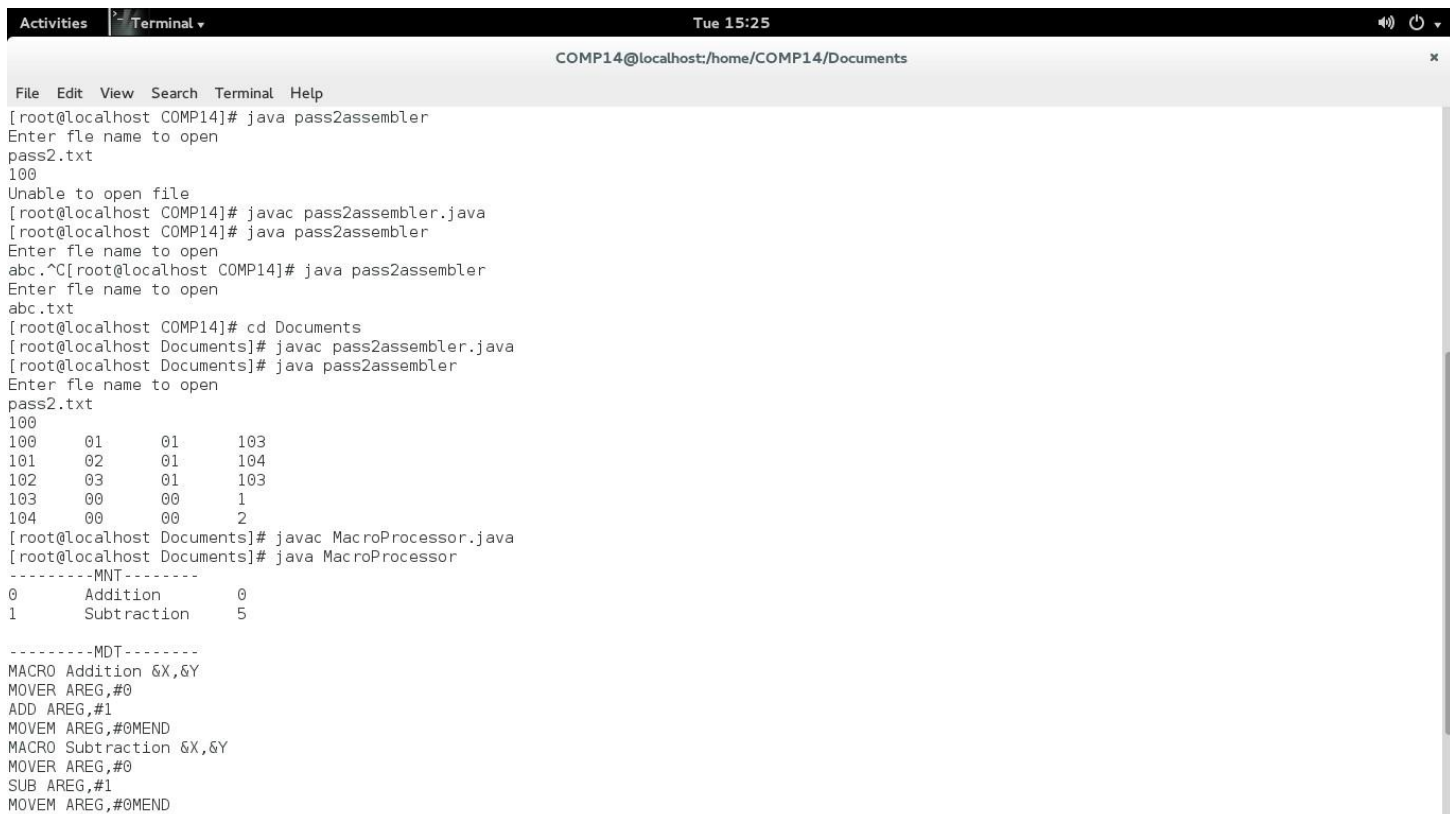
    sl.close();
    return false;

}

}

```

8 Output:



```

[root@localhost COMP14]# java pass2assembler
Enter file name to open
pass2.txt
100
Unable to open file
[root@localhost COMP14]# javac pass2assembler.java
[root@localhost COMP14]# java pass2assembler
Enter file name to open
abc.^C[root@localhost COMP14]# java pass2assembler
Enter file name to open
abc.txt
[root@localhost COMP14]# cd Documents
[root@localhost Documents]# javac pass2assembler.java
[root@localhost Documents]# java pass2assembler
Enter file name to open
pass2.txt
100
100      01      01      103
101      02      01      104
102      03      01      103
103      00      00      1
104      00      00      2
[root@localhost Documents]# javac MacroProcessor.java
[root@localhost Documents]# java MacroProcessor
-----MNT-----
0      Addition      0
1      Subtraction   5

-----MDT-----
MACRO Addition &X,&Y
MOVER AREG,#0
ADD AREG,#1
MOVEM AREG,#0MEND
MACRO Subtraction &X,&Y
MOVER AREG,#0
SUB AREG,#1
MOVEM AREG,#0MEND

```

9 Conclusion:

In this assignment we learn in detail the concept and working of Pass-2 of two pass assembler.