Fokane Sakshi Anil
TE - A - 42
Assignment No : 4

Pass 2. Macroprocessor.

- Aim : Design of a macro Pass - 2

- Problem Statement : write java program for pass - II or a two pass Macroprocessor. The output of assignment - 3 (MNT - MDT & file without any macro defination) should be input for this assignment.

- Theory :-
① Macroprocessor :
macroprocessor is a program that reads a file (or files) & scans them for certain keywords when a keyword is found, it is replaced by some text the keyword / text combination is called a macro.
② Basic task performed by macroprocessor
a) Recognize macro defination
b) Save defination
c) Recognize call
d) Expanded cell & subtitute arguments
In two pass macropreprocessor, you have two algorithm to implement, first pass & second pass Both the algorithm examine line by line over i/p data available
• Pass 1 = Macro defination
• Pass 2 = macro calls & expansion.

- Pass 1 macro defination :

pass 1 algorithm examins each line of the i/P data for macro pseudo opcode. following are the steps that are performed using Pass 1 assembler.

1) Initialize MDTC & MNTC with value one so that previous value of MDTC & MNTC is set to value one.

2) Read the first i/P

3) if this data contains MACRO Pseudo opcode then a) Read the next i/P

b) Enter the Name of Macro & current value of MDTC in MNT

c) Increase the counter value of MNT by valu

d) prepare that argument list array.

e) Enter the macro defination in MDT

f) Read next i/P

g) Substitute the index notation for dummy argument passed in macro.

h) Increase the Counter of the MDT by value one.

i) If mend pseudo. opcode is encountered.

④ if macro pseudo opcode is not encountere in data input then

A) A Copy of input data is created.

B) if end pseudo opcode is found. then go to pass 2.

c) Otherwise read next.

- Pass - 2 macro call & expansion.

Pass two algorithm examines the operation code of every input line to check whether it exist in MNT or not

1) Read the i/p data received from Pass1
2) Examine each operation code for finding respect entery in MNT.
3) If Name of macro is encountered then.
A) A pointer is set to the MNT entry where name of the macro is found.
B) prepare argument list array containing a table of dummy arguments.
c) Increase the value of MDTP by value one.
D) Read next line.
E) Substitute the values from the argument list of the Macro for.
F) If mend pseudo opcode is found then next source of i/p data is read.
G) Else expands data input.
④ when macro name is not found then create expanded data file.
⑤ if end pseudo opcode is encountered then feed the expanded.
⑥ Else read next.

• Algorithm :-
Input:
```
    MACRO
    INCR1  &FIRST, &SECOND= DATA9
    A   1, &FIRST
    B  L2 , & SECOND
```

```
MEND MACRO
INCR2 &ARG1, &ARG2 = DATA5
 L     3, &ARG1
 ST    4, &ARG2
MEND
 PRG2  START
USING  *BASE
INCR1  DATA1
INCR2  DATA3, DATA4
FOUR   DC   F'4'
FIVE   DC   F'5'
BASE   EQU  8
TEMP   DS   1F
DROP   8
 END
```

• Output —— PASS 1 ——

ALA :
```
  [ &FIRST, &SECOND]
  [ &ARG1 , &ARG2]
```
MNT:
```
     [INCR1 , 0]
     [INCR2 , 4]
```
```
MDT :  INCR      &FIRST , &SECOND = DAT
        1         A9
        A         1, #0
        L         2, #
        MEN
        D         &ARG1 , &ARG2 = DATA5
        INCR
        2
        L         3, #0
        ST        4, #1
        D
```

```
//Name: Fokane Sakshi Anil
// TE-A 42
//  ASSINGNMENT:GROUP_A_4

/*
Problem Statement : Write a Java program for pass-II of a two-pass macro-processor. The output of
assignment-3
(MNT, MDT and file without any macro definitions) should be input for this assignment.
*/
import java.io.*;
import java.util.HashMap;
import java.util.Vector;

public class macroPass2 {
	public static void main(String[] Args) throws IOException{
		BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
		BufferedReader b2 = new BufferedReader(new FileReader("mnt.txt"));
		BufferedReader b3 = new BufferedReader(new FileReader("mdt.txt"));
		BufferedReader b4 = new BufferedReader(new FileReader("kpdt.txt"));
		FileWriter f1 = new FileWriter("Pass2.txt");
		HashMap<Integer,String> aptab=new HashMap<Integer,String>();
		HashMap<String,Integer> aptabInverse=new HashMap<String,Integer>();
		HashMap<String,Integer> mdtpHash=new HashMap<String,Integer>();
		HashMap<String,Integer> kpdtpHash=new HashMap<String,Integer>();
		HashMap<String,Integer> kpHash=new HashMap<String,Integer>();
		HashMap<String,Integer> macroNameHash=new HashMap<String,Integer>();
		Vector<String>mdt=new Vector<String>();
		Vector<String>kpdt=new Vector<String>();
		String s,s1;
		int i,pp,kp,kpdtp,mdtp,paramNo;
		while((s=b3.readLine())!=null)
			mdt.addElement(s);
		while((s=b4.readLine())!=null)
			kpdt.addElement(s);
		while((s=b2.readLine())!=null){
			String word[]=s.split("\t");
			s1=word[0]+word[1];
			macroNameHash.put(word[0],1);
			kpHash.put(s1,Integer.parseInt(word[2]));
			mdtpHash.put(s1,Integer.parseInt(word[3]));
			kpdtpHash.put(s1,Integer.parseInt(word[4]));
		}
```

```
                while((s=b1.readLine())!=null){
                        String b1Split[]=s.split("\\s");
                        if(macroNameHash.containsKey(b1Split[0])){
                                pp= b1Split[1].split(",").length-b1Split[1].split("=").length+1;
                                kp=kpHash.get(b1Split[0]+Integer.toString(pp));
                                mdtp=mdtpHash.get(b1Split[0]+Integer.toString(pp));
                                kpdtp=kpdtpHash.get(b1Split[0]+Integer.toString(pp));
                                String actualParams[]=b1Split[1].split(",");
                                paramNo=1;
                                for(int j=0;j<pp;j++){
                                        aptab.put(paramNo, actualParams[paramNo-1]);
                                        aptabInverse.put(actualParams[paramNo-1],paramNo);
                                        paramNo++;
                                }
                                i=kpdtp-1;
                                for(int j=0;j<kp;j++){
                                        String temp[]=kpdt.get(i).split("\t");
                                        aptab.put(paramNo,temp[1]);
                                        aptabInverse.put(temp[0],paramNo);
                                        i++;
                                        paramNo++;
                                }
                                i=pp+1;
                                while(i<=actualParams.length){
                                        String initializedParams[]=actualParams[i-1].split("=");

        aptab.put(aptabInverse.get(initializedParams[0].substring(1,initializedParams[0].length())),initial
izedParams[1].substring(0,initializedParams[1].length()));
                                        i++;
                                }
                                i=mdtp-1;
                                while(mdt.get(i).compareToIgnoreCase("MEND")!=0){
                                        f1.write("+ ");
                                        for(int j=0;j<mdt.get(i).length();j++){
                                                if(mdt.get(i).charAt(j)=='#')
                                                        f1.write(aptab.get(Integer.parseInt("" +
mdt.get(i).charAt(++j))));
                                                else
                                                        f1.write(mdt.get(i).charAt(j));
                                        }
                                        f1.write("\n");
                                        i++;
                        }
                        aptab.clear();
                        aptabInverse.clear();
                    }
                    else
                        f1.write("+ "+s+"\n");
                }
                b1.close();
                b2.close();
                b3.close();
                b4.close();
                f1.close();
        }
}

/*
OUTPUT:
sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A4$ javac macroPass2.java
sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A4$ java macroPass2
sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A4$ cat Pass2.txt

Intermediate - -
M1 10,20,&b=CREG
M2 100,200,&u=&AREG,&v=&BREG

Kpdt—

a       AREG
b       -
u       CREG
v       DREG

pass2—
+ MOVE AREG,10
+ ADD AREG,='1'
+ MOVER AREG,20
+ ADD AREG,='5'
+ MOVER &AREG,100
+ MOVER &BREG,200
+ ADD &AREG,='15'
+ ADD &BREG,='10'
```

```
MNT—
M1      2       2       1       1
M2      2       2       6       3


MDT --
MOVE #3,#1
ADD #3,='1'
MOVER #3,#2
ADD #3,='5'
MEND
MOVER #3,#1
MOVER #4,#2
ADD #3,='15'
ADD #4,='10'
MEND
*/
```