Fokane Sakshi Anil
TE - A - 42

Assignment No : 3
Pass 1 - Macroprocessor.

- Aim : To design DataStructure for Microprocessor

- problem Statement : design Suitable data Structure & implement pass - II of two pass Macroprocessor Using OOP feature in Java.

- Theory :- ① MacroProcessor :
  A macroprocessor is a program that reads a file (or files) & scans them for Certain Keyword when a Keyword is found, it is replaced by some text. The Keyword text Combination is Called a macro.

② Basic tasks Performed by microprocessor
a) Recognize macro defination
b) Save the defination
c) Recognize call
d) Expanded calls & substitute arguments

③ Macro defination Part
It consist of
1. Macroprototype statement. This declares the Name of Macro & the types.
2. Model statement. It is a Statement for which assembly language.
3. preprocessor statement : it used to perform auxiliary function during Macro expansion.

④ Macro call and Expansion
The operation define by macro can be used by writing a macro name in the mnemonic field & find its operand field
⑤ implementation logic :-
1) Defination processing : Scan all macro-defination enter the macro name in macrotable (MNT) store them in (MDT)
. 2) Macro Expansion :-
Examine all statement in assembly Source program to detect the Macrocode
⑥ DataStructure : required for Macro defination :
1) Macro Name table (MN) : fields Name of Macro, #PP, #KP, #MDTP, KPDTP
2) parameter Name table (PNTAB) : fields parameter.
3) Keywords parameters Default Table (KPDTAB) : Feild - parameter name, def value
4) Macro Defination Table (MDT) : model stmt are stored in intermediate code format opcode & operand.
⑦ Algorithm / pseudo code :
Before processing any defination initialize KPDTAB_ptr, MDT_ptr to 0 MNT_ptr to -1

```
* Algorithm : begin {macroprocessor}
                Expanding : FALSE
                While OPCODE ≠ 'END' do
```

```
begin  GETLINE
       PROCESSLINE
       end{while}
     end{macroprocessor}
procedure PROCESSLINE
begin
    search NAMTAB for opcode
  if found then
       EXPAND
  else if opcode ='MACRO' then
      DEFINE
  else write source line do expandedfile.
  end{processline}
```

Input:

```
MACRO INCR &x & &y  &REC1
 ADD  REG  &Y
  MOVEM  &REG  &x
MEND
      START 100
      READ  N1
      READ  N2
      INCR  N1 N2
      STOP
      N1  DS1
      N2  DS2
 END.
```

```
c:\ABC> javac macro.java.
c:\ABC> java macro
```

```
MACRO INCR  &x   &Y   &REC1
      MOVER  &REG1  &x
      ADD    &REG1  &y
      MOVEM  &REG1  &x
MEND
```

```
START 100                  STOP
READ  N1                   N1    DS   1
READ  N2                   N2    DS   2
~~INCR~~ N1  N2            END
```

****

MNT:

INDEX	MACRONAME	MDT NAME
1	INCR	1

****

ALA:

INDEX	ARGUMENT
#1	&x
#2	&y
#3	&REG1

****

MDT

```
MACRO   INCR   &x  &y  &REG1
        MOVER  #3  #1
        ADD    #3  #2
        MOVEM  #3  #1
MEND
```

****

Conclusion :-
Thus, Pass 1 of macro processor is implemented & MNT, MDT & ALA file is generated.

```
//Name: Fokane Sakshi Anil
// TE-A 42
//  ASSINGNMENT:GROUP_A_3
/*
Problem Statement: Design suitable data structures and implement pass-I of a two-pass macro-processor
using
OOP features in Java
*/
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class macroPass1 {
	public static void main(String[] Args) throws IOException{
		BufferedReader b1 = new BufferedReader(new FileReader("input.txt"));
		FileWriter f1 = new FileWriter("intermediate.txt");
		FileWriter f2 = new FileWriter("mnt.txt");
		FileWriter f3 = new FileWriter("mdt.txt");
		FileWriter f4 = new FileWriter("kpdt.txt");
		HashMap<String,Integer> pntab=new HashMap<String,Integer>();
		String s;
		int paramNo=1,mdtp=1,flag=0,pp=0,kp=0,kpdtp=0;
		while((s=b1.readLine())!=null){
			String word[]=s.split("\\s");			//separate by space
			if(word[0].compareToIgnoreCase("MACRO")==0){
				flag=1;
				if(word.length<=2){

	f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1))+"\n");
					continue;
				}
				String params[]=word[2].split(",");
				for(int i=0;i<params.length;i++){
					if(params[i].contains("=")){
						kp++;
						String
keywordParam[]=params[i].split("=");

	pntab.put(keywordParam[0].substring(1,keywordParam[0].length()),paramNo++);
						if(keywordParam.length==2)

	f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+keywordParam[1]+"\n"
);
						else

	f4.write(keywordParam[0].substring(1,keywordParam[0].length())+"\t"+"-"+"\n");

					}
					else{

	pntab.put(params[i].substring(1,params[i].length()),paramNo++);
						pp++;
					}
				}
		f2.write(word[1]+"\t"+pp+"\t"+kp+"\t"+mdtp+"\t"+(kp==0?kpdtp:(kpdtp+1))+"\n");
						kpdtp+=kp;
					}
					else if(word[0].compareToIgnoreCase("MEND")==0){
						f3.write(s+'\n');
						flag=pp=kp=0;
						mdtp++;
						paramNo=1;
						pntab.clear();
					}
					else if(flag==1){
						for(int i=0;i<s.length();i++){
							if(s.charAt(i)=='&'){
								i++;
								String temp="";
								while(!(s.charAt(i)=='
'||s.charAt(i)==',')){
									temp+=s.charAt(i++);
									if(i==s.length())
										break;

								}
								i--;
								f3.write("#"+pntab.get(temp));
							}
							else
								f3.write(s.charAt(i));
						}
						f3.write("\n");
						mdtp++;
					}
					else{
						f1.write(s+'\n');
					}
				}
				b1.close();
				f1.close();
				f2.close();
				f3.close();
				f4.close();
		}
}
/*
OUTPUT:

sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ javacmacroPass1.java
sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ java macroPass1

sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ cat intermediate.txt
M1 10,20,&b=CREG
M2 100,200,&u=AREG,&v=BREG

sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ cat mnt.txt
M1	2	2	1	1
M2	2	2	7	3
M3	2	0	13	4
```

```
sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ cat mdt.txt
MOVE #3,#1
ADD #3,='1'
MOVER #3,#2
M2 69,169
ADD #3,='5'
MEND
MOVER #3,#1
MOVER #4,#2
M3 73,173
ADD #3,='15'
ADD #4,='10'
MEND
ADD #1,#2
MEND

sakshi@sakshi-1011PX:~/Desktop/sakshi_SPOS/Turn1/A3$ cat kpdt.txt
a           AREG
b           -
u           CREG
v           DREG

*/
```