3rd Assignment

Ankita Bonde
Roll No 19

Aim: To design Data structure for Macroprocessor

Problem Statement Design suitable data structures and implement pass I of a two pass Macroprocessor using OOP features in Java.

Theory

① **Macroprocessor**
- It is a program that reads a file and scans them for certain keywords.
- When a keyword is found it is replaced by some text
- The keyword / text combination is called MACRO

② **Basic tasks performed by Macroprocessor**
- Recognize MACRO definition
- Save the definition
- Recognize call
- Expanded calls and substitute arguments

③ **MACRO definition part**
Consists of
① Macro prototype statement - declares the name of macro and types of parameters

② Model statement - statement for which assembly language statement is generated during macro expansion

③ Preprocessor statement - used to perform auxillary function during macro expansion

④ MACRO CALL and Expansion
- appearance of a macro name in the mnemonic field leads to a macro call.
- Macro call replaces such statement by sequence of statement comprising the macro. This is known as macro expansion

⑤ Implementation Logic

① Definition Processing
- Scan all macro definition and for each macro definition enter a MACRO name in the Macro Name table (MNT)
- Store entire definition in the MACRO definition table (MDT) and add auxillary info in MNT such as positional parameters (#PP) no of key word parameters (#KP), macro definition table position (MDTP) etc.

② MACRO Expansion
- Examine all statement in assembly source prog to detect the macro calls
- For each Macro call locate the macro in MNT retrieve MDTP establish the

Correspondence between formal and actual parameters and expand the macro.

⑥ Data structures required for macro definition processing

① MacroName table (MNT) #PP
Fields name of macro, No of positional paramets #KP (No of keyword parametes), MDTP (Macro Definition table Pointer), KPDTP (Keywords parameters Default table Position)

② Parametes Name Table (PNT): Fields parametes name

③ Keyword Parametes Default table Position (KPDTab)
Field: parametes, name, default value

④ Macro definition table (MDT)
Model statements are stored in intermediate code form as:
Opcode & operands.

⑦ Algorithm
A one pass ~~assembles~~ macro processor that alternate between macro definition and macro expansion algorithm.

```
begin {macroprocessor}
    EXPANDING := FAISE
    while OPCODE ≠ 'END' do
```

```
begin
    GETLINE
    PROCESSLINE
    end {while}
end {macro processor}
procedure PROCESSLINE
  begin
    search NAMTAB for OPCODE
    if found then
      EXPAND
    else if OPCODE = "MACRO" then
      DEFINE
    else write source line to expanded
    file
    end {PROCESSLINE}

~~begin~~
Procedure EXPAND
    begin
     EXPANDING := TRUE
     get first line of macro definition {prototype
     from DEFTAB
     set up arguments from macro invocation
     in ARGTAB
     write macro invocation to expanded
     file as a comment
     while not end of macro definition do
        begin
          GETLINE
          PROCESSLINE
```

```
    end {while}
    EXPANDEND := FALSE
end {EXPAND}

procedure GETLINE
    begin
        if EXPANDING then
        begin get next line of macro definition
              from DEFTAB
              substitute arg from ARGTAB
              for positional notation
    end{if}
    else
        read next line from input file
end {GETLINE}
```

Conclusion

Thus pass I of Macroprocessor is implemented and MNT, MDT & ALA file is generated

```
//Name Ankita Bonde
// TE-A 19
//  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:

ankita@ankita-1011PX:~/Desktop/ankita_SPOS/Turn1/A3$ javac macroPass1.java
ankita@ankita-1011PX:~/Desktop/ankita_SPOS/Turn1/A3$ java macroPass1

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

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

```
ankita@ankita-1011PX:~/Desktop/ankita_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

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

*/
```