4rd Assignment

Ankita Bonde
Roll No 19

Aim: Design of a Macro Pass 2

Problem Statement write a java program for pass II of a II pass macro processer The o/p of assignmment-3 (MNT, MDT and file without any macro definition) should be i/p for this assignmment.

Theory

① Macroprocessor
It is a prog that reads files and scan 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 task performed by MACRO processor.
- Recognize macrodefinition
- Save the definition
- Recognize Call.
- Expand call and substitute arguments

- In two pass macroprocessor you have 2 algo to implement 1st and 2nd.
- Both the algo examine line by line over the i/p data available 2algo implement 2pass macroprocessor are

Pass1 Macro Definition
Pass2 Macro calls & Expansion

PAGE
DATE / /

Pass1 MACRO Definition

- Pass1 algo examines each line of the i/p data for macro pseudo opcode
- Following are the steps that are performed during pass1 algo.

1. Intialize MDTC and MNTC val 1 so that the previous value of MDTC and MNTC are set to 1
2. Read the 1st i/p data
3. If the Data contains MACRO pseudo opcode then
A] Read the next data i/p.
B] Enter name of MACRO and current value of MDTC and MNT.
C] Increase the counter value of MNT by one.
D] Prepare the argument list array respective to MACRO found
E] Enter the macro definition in MDT Inc val of MDT by 1
F] Read next line of i/p data
G] Substitute the index notation for dummy arguments passed in MACRO.
H] Inc counter of MDT by 1
I] If mend pseudo opcode is encountered then next source of i/p data is read.
J]. Else expand data i/p.
4. A copy of i/p data is created
A] If macro pseudo opcode not encountered
B] If end pseudo opcode is found go to Pass2

PAGE

DATE / /

c] otherwise read next source of input data

Pass2 Macro calls and Expansion

Pass 2 algo examines the op code of every i/p line to check wheter it exist in MNT or not

1 Read the i/p received from Pass1
2 Examine each opcode for finding respective entry in MNT
3. If name of macro encountered then.
A] A ptr set to MNT where name of macro foun This ptr called Macro Definition Table Pointer (MDTP)
B] Prepare arg list array containg a table of dummy arg.
c] Inc value of MDTP by 1
D] Read next line of MDT
E] Substitute values from arg list of the macr for dummy arg.
F] If mend pseudo opcode found then next source i/p data is read.
G] Else expand data i/p.
4 When macro name is not found then create expanded data file
5 If end pseudo opcode is encountered then feed the expanded source file to Assembler for processing
6 Else read next data i/p.

Conclusion Thus Pass II Macro processor implemented and ALA file generated

Scanned with CamScanner

```
//Name Ankita Bonde
// TE-A 19
//  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:
ankita@ankita-1011PX:~/Desktop/ankita_SPOS/Turn1/A4$ javac macroPass2.java
ankita@ankita-1011PX:~/Desktop/ankita_SPOS/Turn1/A4$ java macroPass2
ankita@ankita-1011PX:~/Desktop/ankita_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
*/
```