

## LAB ASSIGNMENT – 2

### MacroProcessor Pass I

#### PROGRAM –

```
package MacroPass;
import java.io.*;
class arglist {
    String argname,value;
    arglist(String argument) {
        this.argname=argument;
        this.value="";
    }
}

class mnt {
    String name;
    int addr;
    int arg_cnt;
    mnt(String nm, int address)
    {
        this.name=nm;
        this.addr=address;
        this.arg_cnt=0;
    }
    mnt(String nm, int address,int total_arg)
    {
        this.name=nm;
        this.addr=address;
        this.arg_cnt=total_arg;
    }
}

class mdt {
    String stmnt;
    public mdt() {
        stmnt="";
    }
}

public class PASS_1 {

    public static void main(String[] args) throws IOException {
        BufferedReader br1=new BufferedReader(new FileReader("src\\MacroPass\\input.txt"));
        BufferedWriter bw1=new BufferedWriter(new
FileWriter("src\\MacroPass\\Output1.txt"));
        String line;
        mdt[] MDT=new mdt[20];
        mnt[] MNT=new mnt[4];
        arglist[] ARGLIST = new arglist[10];
        boolean macro_start=false,macro_end=false,fill_arglist=false,first = true,start =
false;
        int mdt_cnt=0,mnt_cnt=0,arglist_cnt=0;
        while((line = br1.readLine())!=null)
        {
            line=line.replaceAll(","," ");
            String[] words=line.split("\\s+");
            MDT[mdt_cnt] = new mdt();
            String stmnt = "";
            if(line.contains("START")) start = true;
```

```

for(int i=0;i<words.length;i++)
{
    if(line.contains("MEND"))
    {
        MDT[mdt_cnt++].stmtnt = "\t"+words[i];
        macro_end = true;
    }
    if(line.contains("MACRO"))
    {
        first = true;
        macro_start = true;
        macro_end = false;
    }
    else if(!macro_end)
    {
        if(macro_start)
        {
            MNT[mnt_cnt++]=new mnt(words[i],mdt_cnt);
            System.out.println(mnt_cnt);
            macro_start=false;
            fill_arglist=true;
        }
        if(fill_arglist)
        {
            while(i<words.length) {
                if(words[i].equals("=")) {
                    ARGLIST[arglist_cnt-1].value = words[i+1];
                }
                if (words[i].matches("&[a-zA-Z]+") || words[i].matches("&[a-zA-Z-
Z]+[0-9]+")) {

                    if (first) {
                        MDT[mdt_cnt].stmtnt += "\t" + words[i];
                        first = false;
                    } else MDT[mdt_cnt].stmtnt += "\t," + words[i];
                    ARGLIST[arglist_cnt++] = new arglist(words[i]);
                    MNT[mnt_cnt - 1].arg_cnt++;
                }

                else MDT[mdt_cnt].stmtnt = MDT[mdt_cnt].stmtnt+ "\t" + words[i];
                stmtnt += "\t"+ words[i];
                i++;
            }
            fill_arglist=false;
        }
        else {
            if(words[i].matches("[a-zA-Z]+") || words[i].matches("[a-zA-Z]+[0-
9]+") || words[i].matches("[0-9]+")) {
                MDT[mdt_cnt].stmtnt += "\t" + words[i];
                stmtnt += "\t"+ words[i];
            }
            if(words[i].matches("&[a-zA-Z]+") || words[i].matches("&[a-zA-Z]+[0-
9]+"))
            {
                for(int j=0;j<arglist_cnt;j++)
                if(words[i].equals(ARGLIST[j].argname)) {
                    if(i!=1) MDT[mdt_cnt].stmtnt += "\t,#"+(j+1);
                    else MDT[mdt_cnt].stmtnt += "\t#"+(j+1);
                    stmtnt += "\t#"+(j+1);
                }
            }
        }
    }
}

```

```

        else if(!line.contains("MEND"))
            bw1.write(words[i]+"\\t");
    }
    if(start) bw1.write("\\n");
    if(stmtnt!=" " && !macro_end)
        mdt_cnt++;
}
br1.close();
bw1.close();
BufferedWriter bw = new BufferedWriter(new FileWriter("src\\MacroPass\\MNT.txt"));
for(int i=0;i<mnt_cnt;i++)
{
    bw.write(MNT[i].name+"\\t"+MNT[i].addr+"\\t"+MNT[i].arg_cnt+"\\n");
}
bw.close();

bw1=new BufferedWriter(new FileWriter("src\\MacroPass\\ARG.txt"));
for(int i=0;i<arglist_cnt;i++)
{
    bw1.write(ARGLIST[i].argname+"\\t"+ARGLIST[i].value + "\\n");
}
bw1.close();

bw1=new BufferedWriter(new FileWriter("src\\MacroPass\\MDT.txt"));

for(int i=0;i<mdt_cnt;i++)
{
    bw1.write(MDT[i].stmtnt+"\\n");
}
bw1.close();
}
}

```

## INPUT



input.txt ×

```
1  MACRO
2  INCR &X,&Y,&REG1 = AREG
3  MOVER &REG1,&X
4  ADD &REG1,&Y
5  MOVEM &REG1,&X
6  MEND
7  MACRO
8  DECR &A,&B,&REG2 = BREG
9  MOVER &REG2,&A
10 SUB &REG2,&B
11 MOVEM &REG2,&A
12 MEND
13 START 100
14 READ N1
15 READ N2
16 INCR N1,N2
17 DECR N1,N2
18 STOP
19 N1 DS 1
20 N2 DS 2
21 END
```

## OUTPUT

MNT.txt ×			
1	<u>INCR</u>	0	3
2	<u>DECR</u>	5	3

Output 1 – Macro Name Table.

ARG.txt		
1	&X	
2	&Y	
3	&REG1	<u>AREG</u>
4	&A	
5	&B	
6	&REG2	<u>BREG</u>

Ouptut 1 – Argurment List Array

MDT.txt			
1	INCR	&X ,&Y ,&REG1	= AREG
2	MOVER	#3 ,#1	
3	ADD #3	,#2	
4	MOVEM	#3 ,#1	
5	MEND		
6	DECR	&A ,&B ,&REG2	= BREG
7	MOVER	#6 ,#4	
8	SUB #6	,#5	
9	MOVEM	#6 ,#4	
10	MEND		

Output 3 – Macro Definition Table

Output1.txt ×			
1	START	100	
2	READ	N1	
3	READ	N2	
4	<u>INCR</u>	N1	N2
5	<u>DECR</u>	N1	N2
6	STOP		
7	N1	DS	1
8	N2	DS	2
9	END		

Output 4 – Intermediate Code.

## MacroProcessor Pass II

### PROGRAM –

```
package MacroPass;
import java.io.*;

public class MacroPass_2 {

    public static void main(String[] args) throws IOException {
        mdt[] MDT = new mdt[20];
        mnt[] MNT = new mnt[4];
        arglist[] formal_parameter = new arglist[10];
        int macro_addr = -1;

        boolean macro_start=false,macro_end=false;
        int macro_call = -1;
        int

mdt_cnt=0,mnt_cnt=0,formal_arglist_cnt=0,actual_arglist_cnt=0,temp_cnt=0,temp_cnt1=0;

        BufferedReader br1=new BufferedReader(new FileReader("src\\MacroPass\\MNT.txt"));
        String line;
        while((line = br1.readLine())!=null)
        {
            String[] parts=line.split("\\s+");
            MNT[mnt_cnt++] = new

mnt(parts[0],Integer.parseInt(parts[1]),Integer.parseInt(parts[2]));
        }
        br1.close();
        System.out.println("\n\t*****MACRO NAME TABLE*****");
        System.out.println("\n\tINDEX\tNAME\tADDRESS\tTOTAL ARGUMENTS");
        for(int i=0;i<mnt_cnt;i++)

System.out.println("\t"+i+"\t\t"+MNT[i].name+"\t\t"+MNT[i].addr+"\t\t"+MNT[i].arg_cnt);

        br1=new BufferedReader(new FileReader("src\\MacroPass\\ARG.txt"));
        while((line = br1.readLine())!=null)
        {
            String[] parameters=line.split("\\s+");
            formal_parameter[formal_arglist_cnt++]=new arglist(parameters[0]);
            if(parameters.length>1)
                formal_parameter[formal_arglist_cnt-1].value = parameters[1];
        }
        br1.close();

        System.out.println("\n\n\t*****FORMAL ARGUMENT LIST*****");
        System.out.println("\n\tINDEX\tNAME\tADDRESS");
        for(int i=0;i<formal_arglist_cnt;i++)

System.out.println("\t"+i+"\t\t"+formal_parameter[i].argname+"\t"+formal_parameter[i].value);

        br1=new BufferedReader(new FileReader("src\\MacroPass\\MDT.txt"));
        while((line = br1.readLine())!=null)
        {
            MDT[mdt_cnt]=new mdt();
            MDT[mdt_cnt++].stmnt=line;
        }
        br1.close();
    }
}
```

```

System.out.println("\n\t*****MACRO DEFINITION TABLE*****");
System.out.println("\n\tINDEX\t\tSTATEMENT");
for(int i=0;i<mdt_cnt;i++)
    System.out.println("\t"+i+"\t"+MDT[i].stmt);

brl=new BufferedReader(new FileReader("src\\MacroPass\\input.txt"));
arglist[] actual_parameter=new arglist[10];
BufferedWriter bw1 = new BufferedWriter(new
FileWriter("src\\MacroPass\\Output.txt"));
while((line = brl.readLine())!=null)
{
    line=line.replaceAll(",", " ");
    String[] tokens=line.split("\\s+");
    temp_cnt1=0;
    for(String current_token:tokens)
    {
        if(current_token.equalsIgnoreCase("macro"))
        {
            macro_start=true;
            macro_end=false;
        }
        if(macro_end && !macro_start)
        {
            if(macro_call != -1 && temp_cnt<formal_arglist_cnt-1)
            {
                if(formal_parameter[actual_arglist_cnt].value != "")
                    actual_parameter[actual_arglist_cnt++]=new
arglist(formal_parameter[actual_arglist_cnt-1].value);

                actual_parameter[actual_arglist_cnt++]=new
arglist(current_token);

                if(formal_parameter[actual_arglist_cnt].value != "")
                    actual_parameter[actual_arglist_cnt++]=new
arglist(formal_parameter[actual_arglist_cnt-1].value);

            }

            for(int i=0;i<mnt_cnt;i++)
            {
                if(current_token.equals(MNT[i].name))
                {
                    macro_call=i;
                    temp_cnt1 = temp_cnt1 +MNT[i].arg_cnt;
                    break;
                }
                temp_cnt1 = temp_cnt1 + MNT[i].arg_cnt;
            }
            if(macro_call == -1)
                bw1.write("\t" + current_token);
        }
        if(current_token.equalsIgnoreCase("mend"))
        {
            macro_end=true;
            macro_start=false;
        }
    }
    if(macro_call != -1)
    {
        macro_addr=MNT[macro_call].addr+1;
        while(true)
        {

```



```

        if (MDT[macro_addr].stmtnt.contains("mend") ||
MDT[macro_addr].stmtnt.contains("MEND"))
        {
            macro_call = -1;
            break;
        }
        else
        {
            bw1.write("\n");
            String[] temp_tokens=MDT[macro_addr++].stmtnt.split("\\s+");

            for(String temp:temp_tokens)
            {
                if(temp.matches("#[0-9]+") || temp.matches(",#[0-9]+"))
                {
                    int num = Integer.parseInt(temp.replaceAll("[^0-9]+",
""));
                    bw1.write(actual_parameter[num-1].argname+"\t");
                }
                else
                {
                    bw1.write(temp + "\t");
                }
            }
        }
    }
    if(!macro_start )
        bw1.write("\n");
    macro_call= -1;
}
br1.close();
bw1.close();

System.out.println("\n\n\t*****ACTUAL ARGUMENT LIST*****");
System.out.println("\n\tINDEX\tNAME");
for(int i=0;i<actual_arglist_cnt;i++)
    System.out.println("\t"+i+"\t"+actual_parameter[i].argname);
}
}

```

**NOTE – Outputs of PASS 1 are pipelined as inputs to PASS II**

## OUTPUT

### \*\*\*\*\*MACRO NAME TABLE\*\*\*\*\*

INDEX	NAME	ADDRESS	TOTAL ARGUMENTS
0	INCR	0	3
1	DECR	5	3

### \*\*\*\*\*FORMAL ARGUMENT LIST\*\*\*\*\*

INDEX	NAME	ADDRESS
0	&X	
1	&Y	
2	&REG1	AREG
3	&A	
4	&B	
5	&REG2	BREG

### \*\*\*\*\*MACRO DEFINITION TABLE\*\*\*\*\*

INDEX	STATEMENT
0	INCR    &X ,&Y ,&REG1 =    AREG
1	MOVER   #3 ,#1
2	ADD #3   ,#2
3	MOVEM   #3 ,#1
4	MEND
5	DECR    &A ,&B ,&REG2 =    BREG
6	MOVER   #6 ,#4
7	SUB #6   ,#5
8	MOVEM   #6 ,#4
9	MEND

### \*\*\*\*\*ACTUAL ARGUMENT LIST\*\*\*\*\*

INDEX	NAME
0	N1
1	N2
2	AREG
3	N1
4	N2
5	BREG

Output.txt	
1	
2	
3	START 100
4	READ N1
5	READ N2
6	
7	MOVER AREG N1
8	ADD AREG N2
9	MOVEM AREG N1
10	
11	MOVER BREG N1
12	SUB BREG N2
13	MOVEM BREG N1
14	STOP
15	N1 DS 1
16	N2 DS 2
17	END

**Final Output – Expanded Code.**