

# T. Y. B. Tech Computer Engineering

Student Name	Pranav Dambe (Nikam)
SRN No	202201704
Roll No	68
PRN	2280030506
Division	D(D3)
Subject	System Programming
Year	Third Year

# Assignment - 4

#### QUE 1:

Design suitable data structures and implement simple Macro definition processing for the hypothetical ALP. Generate different Parameter Tables and MDT, MNT. Detect any one error. Input file contains multiple macro definitions.

Output: Submit a single .doc / .pdf file containing input ALP, MNT, MDT, PNTAB , KPDTAB , EVNTAB , SSNTAB , SSTAB in that sequence.

### **OUTPUT:**

### Input.asm:

```
[#] input.asm
      MACRO
      SAMPLE &X, &N, &REG=AREG, &REG1=BRAG
    LCL &M
  4 &M SET 0
    MOVER &REG, ="0"
     .MORE MOVEM &REG, &X + &M
     .AGAIN &M SET &M + 1
     AIF (&M NE &N) .MORE
      MEND
      MACRO
     CALC &Z, &W, &REG2=CREG, &REG3=DREG
     LCL &T
     .LABEL &T SET &Z
      ADD &REG2, &W
      .LOOP MOVEM &REG3, &Z
      MEND
      MACRO
     INCR &A, &B, &Q=10
    GBL &NUM, &ALPHA
     .NEXT ADD &A, &B
      SUB &Q, &A
      .STEP MUL &Q, 15
      MEND
```

### MACRO Name Table (MNT):

MNT:							
Index	MACRO	#PP	#KP	#EV	MDTP	KPDTP	SSTP
1:	SAMPLE	2	2	1	1	1	1
2:	CALC	2	2	1	8	3	3
3:	INCR	2	1	2	13	5	5

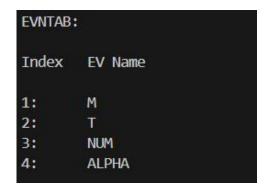
### MACRO Definition Table (MDT):

```
MDT:
       MACRO Definition
Index
                (E, 1)
1:
        LCL
2:
        (E,1)
                SET
                        0
                        ="0"
       MOVER
                (P,3)
        (S, 1)
               MOVEM
4:
                        (P,3)
                                (P,1)
                                                (E,1)
        (S, 2)
5:
                                (E,1)
               (E,1)
                        SET
                                                1
6:
        AIF
                (E,1)
                        NE
                                (P,2)
                                       (S, 1)
        MEND
7:
        LCL
                (E, 2)
8:
        (S, 3)
                (E,2)
9:
                        SET
                                (P,5)
10:
        ADD
                (P,7)
                        (P,6)
11:
        (S, 4)
               MOVEM
                        (P,8)
                                (P,5)
       MEND
12:
13:
       GBL
                (E, 3) (E, 4)
        (S, 5)
               ADD
                        (P,9)
                                (P,10)
14:
15:
        SUB
                (P,11)
                        (P,9)
        (S, 6) MUL
                        (P,11) 15
16:
17:
        MEND
```

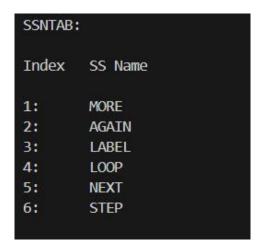
## Parameter Name Table (PNTAB):

```
PNTAB:
Index
         Parameter Name
1:
        X
2:
         N
3:
         REG
4:
         REG1
5:
         Z
6:
         W
7:
         REG2
8:
         REG3
9:
         A
10:
         В
         Q
11:
```

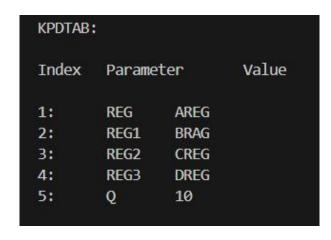
### Expansion Time Variable Name Table (EVNTAB):



## Sequencing Symbol Name Table (SSNTAB):



# Keyword Parameter Default Table (KPDTAB):



#### Sequencing Symbol Table (SSNTAB):

```
SSTAB:

Index MDT_ENTRY

1: 4
2: 5
3: 9
4: 11
5: 14
6: 16
```

#### CODE:

```
import java.util.*;
import java.io.*;
class MacroPass1
 ArrayList<String> MNT = new ArrayList<>();
 ArrayList<String> MDT = new ArrayList<>();
 ArrayList<String> PNTAB = new ArrayList<>();
 ArrayList<String> EVNTAB = new ArrayList<>();
 ArrayList<String> SSNTAB = new ArrayList<>();
 ArrayList<String> KPDTAB = new ArrayList<>();
 ArrayList<Integer> SSTAB = new ArrayList<>();
 ArrayList<String> trackSSN = new ArrayList<>();
 void pass1(String fileName) throws IOException
    String macroName = null;
    Integer PP = 0, KP = 0, EV = 0, tempEV=0;
    Integer MDTP = 1, KPDTP = 0, SSTP = 1;
    boolean flag = false;
    try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
      String line;
      while ((line = br.readLine()) != null)
        if(line.isEmpty()){
        String[] words = line.split("\\s+");
        if (words.length == 1 && words[0].equalsIgnoreCase("MACRO")) {
           flag = true;
```

```
line = br.readLine();
          words = line.split("\\s+");
          macroName = words[0];
          if (words.length <= 1) {</pre>
            MNT.add(macroName + "\t" + PP + "\t" + KP + "\t" + EV + "\t" + MDTP + "\t" + (KP == 1)
  0 ? KPDTP : (KPDTP + 1)) + "\t" + SSTP);
          for (int i = 1; i < words.length; i++) {
            words[i] = words[i].replaceAll("[&,]", "");
            if (words[i].contains("=")) {
               String param_value[] = words[i].split("=");
               KP++:
               PNTAB.add(param_value[0]);
               KPDTAB.add(String.join("\t",param_value));
            } else {
               PP++:
               PNTAB.add(words[i]);
       else if (words[0].equalsIgnoreCase("LCL") || words[0].equalsIgnoreCase("GBL"))
          flag = true;
          ArrayList<String> EVname = new ArrayList<>();
          for (int i = 1; i < words.length; i++) {
             String cleanedWord = words[i].replaceAll("[&,]", "");
             EVNTAB.add(cleanedWord);
            EV++:
            tempEV++;
            EVname.add("(E, " + tempEV + ")");
          String mdtEntry = words[0] + "\t" + String.join("\t", EVname);
          MDT.add(mdtEntry);
          EVname.clear();
       else if (words.length == 1 && words[0].equalsIgnoreCase("MEND"))
          flag = false;
          MDT.add("MEND");
          createSSTAB();
          SSTP = updateSSTP();
          MNT.add(macroName + "\t" + PP + "\t" + KP + "\t" + EV + "\t" + MDTP + "\t" + (KP == 0?
KPDTP: (KPDTP + 1)) + "\t" + SSTP);
          MDTP = MDT.size() + 1;
          KPDTP += KP:
          PP = KP = EV = 0;
       else if (flag)
          if (words[0].startsWith(".")) {
            String cleanedWord = words[0].replaceAll("[.]", "");
            if (!SSNTAB.contains(cleanedWord)) {
               SSNTAB.add(cleanedWord);
```

```
trackSSN.add(cleanedWord);
          else if (words[words.length - 1].startsWith(".")) {
            String cleanedWord = words[words.length - 1].replaceAll("[.]", "");
            if (!SSNTAB.contains(cleanedWord)) {
               SSNTAB.add(cleanedWord);
               trackSSN.add(cleanedWord);
          ArrayList<String> MDT_parts = new ArrayList<>();
          for (int i = 0; i < words.length; i++)</pre>
            if (words[i].contains("&") || words[i].startsWith("."))
               words[i] = words[i].replaceAll("[&,.()]", "");
               if (PNTAB.contains(words[i])) {
                 MDT_parts.add("(P," + (PNTAB.indexOf(words[i]) + 1) + ")");
               } else if (EVNTAB.contains(words[i])) {
                 MDT_parts.add("(E," + (EVNTAB.indexOf(words[i]) + 1) + ")");
               } else if (SSNTAB.contains(words[i])) {
                 MDT_parts.add("(S, " + (SSNTAB.indexOf(words[i]) + 1) + ")");
            else {
               MDT_parts.add(words[i]);
          String mdtEntry = String.join("\t", MDT_parts);
          MDT.add(mdtEntry);
          MDT_parts.clear();
  } catch (Exception e) {
     System.out.println(e);
void createSSTAB(){
  for(int j=0; j<trackSSN.size(); j++){</pre>
     String str = trackSSN.get(j);
     Integer indexinSS = SSNTAB.indexOf(str)+1;
     Integer indexinMDT=0;
    for (int i = 0; i < MDT.size(); i++) {</pre>
       if (MDT.get(i).startsWith("(S, "+indexinSS +")")) {
          indexinMDT = i + 1;
          SSTAB.add(indexinMDT);
          break;
```

```
Integer updateSSTP() {
  Integer temp = 0;
  String str = trackSSN.get(0);
  Integer indexinSS = SSNTAB.indexOf(str)+1;
  Integer indexinMDT=0;
  for (int i = 0; i < MDT.size(); i++) {</pre>
    if (MDT.get(i).startsWith("(S, "+indexinSS +")")) {
      indexinMDT = i + 1;
      temp = SSTAB.indexOf(indexinMDT)+1;
      break;
  trackSSN.clear();
  return temp;
void Print_Tables()
  System.out.println("\n----\nMNT:\n");
  System.out.println("Index\tMACRO\t#PP\t#KP\t#EV\tMDTP\tKPDTP\tSSTP");
  for(int i=0; i<MNT.size(); i++){</pre>
    System.out.println((i + 1) + ": \t^" + (MNT.get(i)));
  System.out.println("\n----\nMDT:\n");
  System.out.println("Index" + "\t" + "MACRO Definition\n");
  for (int i = 0; i < MDT.size(); i++) {</pre>
    System.out.println((i + 1) + ":\t" + (MDT.get(i)));
  System.out.println("\n-----\nPNTAB:\n");
  System.out.println("Index" + "\t" + "Parameter Name\n");
  for (int i = 0; i < PNTAB.size(); i++) {
    System.out.println((i + 1) + ":\t" + PNTAB.get(i));
  System.out.println("\n-----\nEVNTAB:\n");
  System.out.println("Index" + "\t" + "EV Name\n");
  for (int i = 0; i < EVNTAB.size(); i++) {</pre>
    System.out.println((i + 1) + ":\t" + EVNTAB.get(i));
  System.out.println("\n-----\nSSNTAB:\n");
  System.out.println("Index" + "\t" + "SS Name\n");
  for (int i = 0; i < SSNTAB.size(); i++) {</pre>
    System.out.println((i + 1) + ":\t" + SSNTAB.get(i));
  System.out.println("\n-----\nKPDTAB:\n");
  System.out.println("Index" + "\t" + "Parameter"+ "\t" + "Value\n");
  for (int i = 0; i < KPDTAB.size(); i++) {</pre>
    System.out.println((i + 1) + ":\t" +(KPDTAB.get(i)));
```

```
System.out.println("\n----\nSSTAB:\n");
System.out.println("Index" + "\t" + "MDT_ENTRY\n");
for(int i=0; i<SSTAB.size(); i++){
    System.out.println((i + 1) + ":\t" + (SSTAB.get(i)));
}

public class Assignment_4 extends MacroPass1 {
    public static void main(String[] arg) {
        Assignment_4 MACRO = new Assignment_4();
        try {
            MACRO.pass1("input.asm");
            MACRO.Print_Tables();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```