Macro & Macro Preprocessor

Q.1) Define Macro. How Macro & Subroutine differ from each other. What is Lexical & Semantic Expansion?

1. A macro ids unit of specification for program generation through expansion.
   A macro name is an abbreviation, which stands for some related lines of code.

- Macros are useful for following purpose:--
  --to simplify and reduce amount of repetitive coding
  --to reduce errors caused by repetitive coding
  --to make an assembly program more readable

2. **lexical expansion:**
   Lexical Substitution
   --A model statement consist of three types of strings
   --Type 1: An ordinary string which stands for itself
   --Type 2: The name of a formal parameter which is preceded by the character '&'.
   -Type 3: The name of a pre-processor variable which is also preceded by the character '&'.
   - During lexical expansion, string of type 1 are retained without substitution.
   - Strings of type 2 and 3 are replaced by values of formal parameter or pre-processor variables.

3. **Semantic expansion:**

- Semantic expansion is the generation of instruction tailored to the requirements of a specific usage.

- It can be achieved by the combination of advanced macro facilities like AIF, AGO statements and expansion time variables.

	MACROS	PROCEDURE /SUBROUTINE
1.	The corresponding m/c code is written everytime a macro is called in a program.	1. The corresponding m/c code is written only once in memory
2.	Program takes up more memory space	2. Program takes up comparatively less memory space
3.	No transfer of program counter	3. Transfering of PC is required
4.	No overhead of using stack for using transfering control	4. Overhead of using stack for transferring control
5.	Execution is fast	5. execution is slow
6.	Assembly time is more	6. Assembly time is less
7.	More advantegous to the programs when repeated group of instruction is too short	7. More advantegous to the programs when repeated group of instruction is quite large.

Q2) DISCUSS MACRO DEFINITION & MACRO CALL IN DETAIL WITH EXAMPLE.

1. Macro definition:

Macro definition is enclosed between amacro header and macro end statement. Macro definition consist of--

- A Macro Prototype Statement : declares the name of macro and the names and kind of its parameter.
- One or more Model Statements: is a statement from which assembly language statement may be generated during macro expansion.
- Macro Preprocessor Statements: is used to perform auxiliary function during macro expansion.

2. The macro prototype statement has syntax:

<macro name> [<formal parameter specification>[,..]]

Formal parameter specification has form

&<parameter name>[<parameter kind>]

3. Macro call:

A macro is called by writing the macro name in the mnemonic field of an assembly statement.

- The macro call has syntax:
  <macro name> [<actual parameter specification>[,..]]

Ex.:

EXAMPLE OF MACRO DEFINITION

```
MACRO
INCR        &MEM_VAL, &INCR_VAL, &REG
MOVER       &REG, &MEM_VAL
ADD         &REG, &INCR_VAL
MOVEM       &REG, &MEM_VAL
MEND
```

- Example of Macro Call

```
INCR A, B, AREG
```

Q.3) EXPLAIN DIFFERENT KINDS OF PARAMETERS IN MACRO.

THE RULE OF DETERMINING THE VALUE OF FORMAL PARAMETER DEPEND ON KIND OF PARAMETER:

1. Positional Parameter
2. Keyword Parameter
3. Default Specification of Parameter
4. Macros with Mixed Parameter
5. Other uses of Parameter

1. Positional Parameter

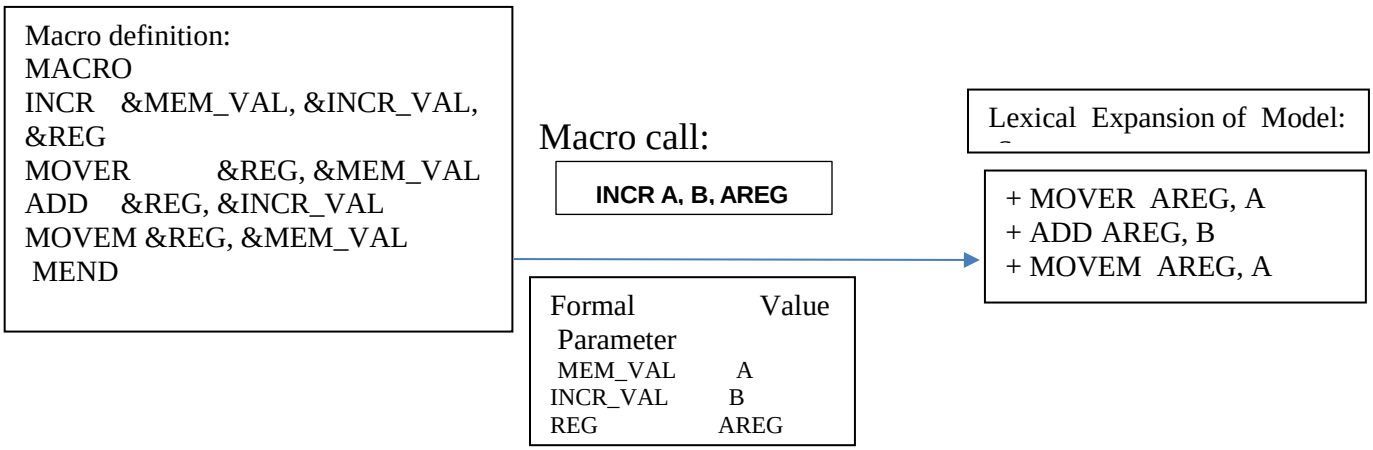
-- A positional formal parameter is written as &<parameter name>.

--The <actual parameter specification> in a macro call is simply an <ordinary string>.

--The value of a positional formal parameter XYZ is determined by the rule of positional association as :

- Find the ordinal position of XYZ in the list of formal parameters in macro prototype statement.
- Find the actual parameter specification occupying the same ordinal position in the list of actual parameters in macro call statement.

Positional Parameter Example:



2. Keyword Parameter

- For keyword parameter, formal parameter is written as &<parameter name>=.
- The <actual parameter specification> in a macro call is written as <formal parameter name> = <ordinary string>.

The value of a positional formal parameter XYZ is determined by the rule of positional association as :

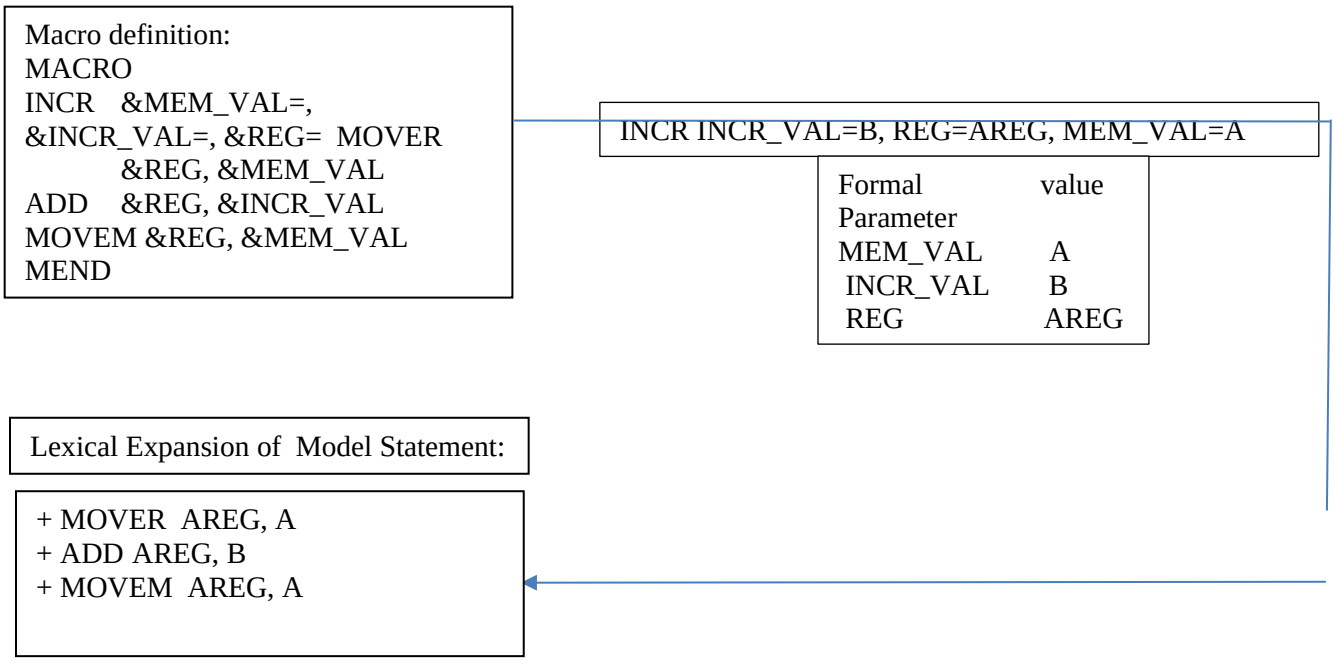
Find the actual parameter specification which has the form XYZ = <ordinary string>.

Let <ordinary string> in the specification be the string ABC. Then the value of formal parameter XYZ is ABC.

Keyword Parameter Example:

Macro call:

```
INCR MEM_VAL=A, INCR_VAL=B, REG=AREG
```



3.Default Specification of Parameter:

- A default is a standard specification in the absence of an explicit specification by the programmer.
- The syntax of formal parameter specification is

  &<parameter name>[<parameter kind>[<default value>]]

Example on page no 22&23 from notes

4. Macros with mixed parameter list:

- A macro may be defined to use both positional and keyword parameter.
- In such a case, all positional parameter must precede all keywords parameter.

  Example on page no.25

5. Other uses of parameters:

– Formal parameter can also appear in the label and opcode fields of model statement.

Example on page no.27

Q.4) Write a note on-Nested Macro calls

- A model statement in a macro may constitute a call on another macro. Such calls are known as nested macro calls.
- Macro containing the nested call is referred to as the outer macro and the called macro as the inner macro.
- Expansion of nested macro calls follows the last- in-first-out (LIFO) rule.

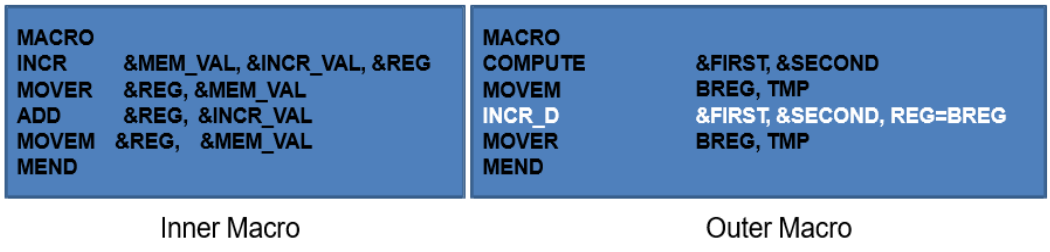
- Example:

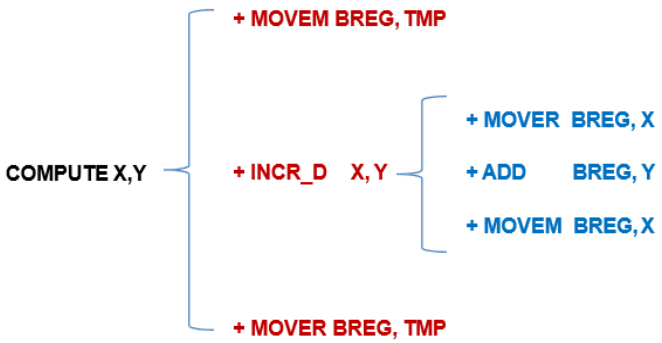
```
MACRO
INCR      &MEM_VAL, &INCR_VAL, &REG
MOVER     &REG, &MEM_VAL
ADD       &REG, &INCR_VAL
MOVEM     &REG,  &MEM_VAL
MEND
```

Inner Macro

```
MACRO
COMPUTE   &FIRST, &SECOND
MOVEM     BREG, TMP
INCR_D    &FIRST, &SECOND, REG=BREG
MOVER     BREG, TMP
MEND
```

Outer Macro

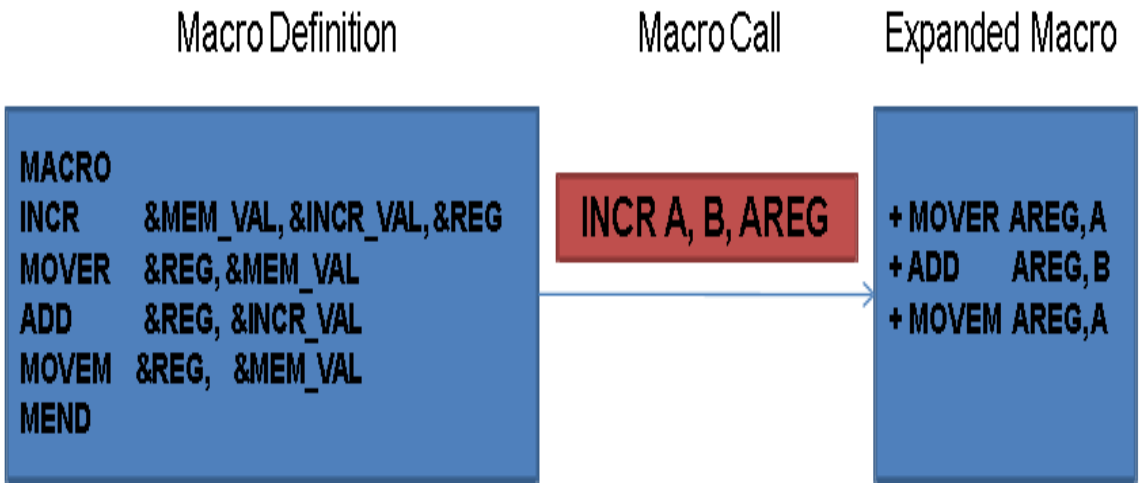




a) Macro Expansion:-

-Macro calls leads to macro expansion

-During macro expansion, the macro call is replaced by a sequence of assembly statements.



- Two key notions concerning macro expansion are:
  - Expansion Time Control Flow: determines the order in which model statements are visited during macro expansion.
  - Lexical substitution: Is used to generate an assembly statement from a model statement.

- Flow of control during expansion:
  - The default flow of control during macro expansion is sequential.
  - A preprocessor statement can alter the flow of control during the expansion such that
    - Some model statement are never visited – Conditional Expansion
    - Some model statements are repeatedly visited – Expansion Time Loop
    - The flow of control during macro expansion is implemented using a Macro Expansion Counter (MEC).

Algorithm (Outline of macro expansion)

1. MEC := Statement number of first statement following the prototype statement
2. While statement pointed by MEC is not a MEND statement
   (a) If a model statement then
      (i) Expand the statement
      (ii) MEC := MEC + 1
   (b) Else (i.e. a preprocessor statement)
      (i) MEC := new value specified in the statement
3. Exit from macro expansion

- **Lexical Substitution**
  - A model statement consist of three types of strings
    - Type 1: An ordinary string which stands for itself
    - Type 2: The name of a formal parameter which is preceded by the character '&'.
    - Type 3: The name of a preprocessor variable which is also preceded by the character '&'.
  - During lexical expansion, strings of type 1 are retained without substitution.
  - Strings of type 2 and 3 are replaced by values of formal parameter or preprocessor variables.
  - The rule for determining the value of formal parameter depend on kind of parameter.
    - Positional Parameter
    - Keyword Parameter
    - Default Specification of Parameter
    - Macros with Mixed Parameter
    - Other uses of Parameter

Q.5) EXPLAIN ADVANCED MACRO FACILITIES WITH EXAMPLE.

THESE FACILITIES SUPPORT SEMANTIC EXPANSION.

THESE FACILITIES CAN BE GROUPED INTO:

1. FACILITIES FOR ALTERATION OF FLOW OF CONTROL DURING EXPANSION
2. EXPANSION TIME VARIABLES
3. ATTRIBUTES OF PARAMETERS

1. A Facilities for alteration of flow of control during expansion

- Two features are provided to facilitate alteration of flow of control during expansion

1. Expansion time sequencing symbol
2. Expansion time statements AIF, AGO and ANOP

1.Expansion time sequencing symbol:

- A sequencing symbol (SS) has the syntax.<ordinary string>
- SS is defined by putting it in the label field of a statement in macro body.
- It is used as an operand in AIF or AGO to designate the destination of an expansion time control transfer.

2. **Expansion time statements AIF, AGO and ANOP:**
    - An AIF statement has the syntax AIF (<expression>) <sequencing symbol>
    - An AGO statement has the syntax AGO <sequencing symbol>
    - An ANOP statement is written as <sequencing symbol> ANOP

- Example

```
        MACRO
        EVAL    &X, &Y, &Z
        AIF             (&Y EQ &X)  .ONLY
        MOVER           AREG, &X
        SUB             AREG, &Y
        ADD             AREG, &Z
        AGO             .OVER
.ONLY   MOVER           AREG, &Z
.OVER   MEND
```

b) Expansion time variables (EV's)

- EV's are variables which can only be used during the expansion of macro calls.
- A local EV is created for use only during a particular macro call.
- A global EV exists across all macro calls situated in a program and can be used in macro.

  LCL <EV specification>[<EV specification>..]
  GBL <EV specification>[<EV specification>..]
- <EV specification> has syntax &<EV name> where

<EV name> is an ordinary string.

- Value of EV's can be manipulated through the preprocessor statement SET which is written as

  <EV specification> SET <SET-expression>

Example:

```
        MACRO
        CONTANTS
        LCL     &A
&A      SET     1
        DB      &A
&A      SET     &A+1
        DB      &A
        MEND
```

c) Attributes of parameters

- An attribute is written using the syntax <attribute name>'<formal parameter specification>
- The type, length and size attributes have the names T, L and S.

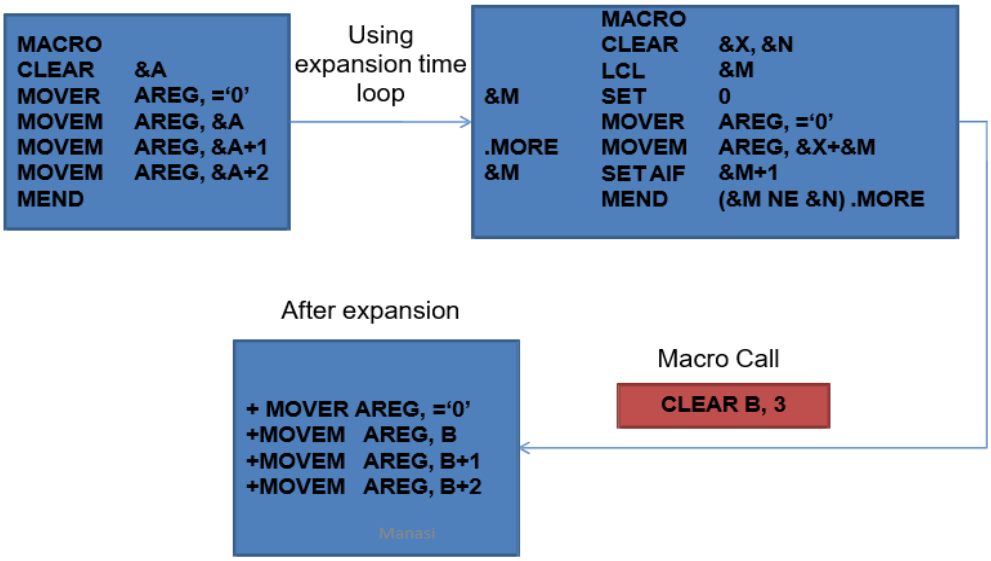
```
        MACRO
        DCL_CONST   &A
        AIF                 (L'&A EQ 1)  .NEXT
        ----
.NEXT   ----
        ----
        MEND
```

Conditional Expansion

- It helps in generating assembly code specifically suited to the parameters in a macro call.
- The AIF and AGO statements are used for this purpose.
  Example:

```
        MACRO
        EVAL    &X, &Y, &Z
        AIF             (&Y EQ &X)  .ONLY
        MOVER           AREG, &X
        SUB             AREG, &Y
        ADD             AREG, &Z
        AGO             .OVER
.ONLY   MOVER           AREG, &Z
.OVER   MEND
```

- **Expansion time loops**
  It is necessary to generate many similar statements during the expansion of a macro.
  Expansion time loops can be written using expansion time variables and expansion time control transfer statements AIF and AGO.



- Other facilities for Expansion Time Loops
  Many assembler provide other facilities for conditional expansion such as
  1. The RPET statement
  2. The IRP statement

1.The RPET statement:

- The RPET statement has the syntax RPET <expression>
- The statement between REPT and ENDM statement would be processed for expression <expression> number of times.

- Example:

```
        MACRO
        CONST10
        LCL         &M
&M      SET         1
        REPT  10
        DC          '&M'
&M      SETA        &M+1
        ENDM
        MEND
```

2. The IRP statement

a. The IRP statement has the syntax IRP <formal parameter>, <argument-list>
b. The formal parameter mentioned in the statement takes successive values from the argument list.
c. For each value, the statement between the IRP and ENDM statements are expanded once.
   Example:

```
MACRO
CONSTS      &M, &N, &Z
IRP         &Z, &M, 7, &N
DC          '&Z'
ENDM
MEND
```

- **Semantic Expansion**
  i. Semantic expansion is the generation of instruction tailored to the requirements of a specific usage.
  ii. It can be achieved by the combination of advanced macro facilities like AIF, AGO statements and expansion time variables.
  Example:

```
            MACRO
            CREATE_CONST    &X, &Y
            AIF             (T'&X EQ B) .BYTE
&Y          DW              25
            AGO             .OVER
.BYTE       ANOP
&Y          DB              25
.OVER       MEND
```

Q.6) Explain the following facilities for expansion

time loops: REPT & IRP.

Refer above answer

Q.7) Explain the data structures required for

design of macro preprocessor.

Data Structures

- Macro Name Table (MNT)-1
  - Macro Name
  - Number of positional parameter (#PP)
  - Number of keyword parameter (#KP)
  - Number of expansion time variables (#EV)
  - MDT Pointer (MDTP)
  - KPDTAB Pointer (KPDTP)
  - SSTAB Pointer (SSTP)

MNT

NAME	#PP	#KP	#EV	MDTP	KPDTP	SSTP
CLEARMEM	2	1	1	25	10	5

- Parameter Name Table (PNTAB)-2
  - Parameter name

PNTAB

PNTAB_ptr	Parameter Name
1	X
2	N
3	REG

- EV Name Table (EVNTAB)-3
  - EV Name

EVNTAB

EV Name
M

- SS Name Table (SSNTAB)-4
  - SS name

SSNTAB

SSNTAB_ptr	SS Name
1	MORE

- Keyword Parameter Default Table (KPDTAB)-5
  - Parameter Name
  - Default value

KPDTAB

KPDTAB_ptr	Parameter Name	Default value
.		
10	REG	AREG

- Macro Definition Table (MDT)-6
  - Label
  - Opcode
  - Operand

MDT

MDT_ptr	LABEL	OPCODE	OPERAND
	.	.	.
25		LCL	(E, 1)
26	(E, 1)	SET	0
27		MOVER	(P, 3), ='0'
28		MOVEM	(P, 3), (P, 1)+(E, 1)
29	(E, 1)	SET	(E, 1)+1
30		AIF	((E, 1) NE (P, 2)) (S, 1)
31		MEND	
	.	.	.

- Actual Parameter Table (APTAB)-7
  - Value

APTAB

APTAB_ptr	Value
1	AREA
2	10
3	AREG

- EV Table (EVTAB)-8
  - Value

EVTAB

EVTAB_ptr	Value
1	0

- SS Table (SSTAB)-9
  - MDT entry#

SSTAB

SSTAB_ptr	MDT Entry #
.	
5	28

Q.8) Write the algorithm for processing of macro definition & macro expansion.

Algorithm – Processing of a Macro Definition

1. SSNTAB_ptr := 1; PNTAB_ptr := 1; KPDTAB_ptr :=1;

   SSTAB_ptr := 1; MDT_ptr := 1;

2. Process the macro prototype statement and form the MNT entry

(a) name := macro name;

(b) for each positional parameter

(i) Enter parameter name in PNTAB[PNTAB_ptr].

(ii) PNTAB_ptr := PNTAB_ptr +1;

(iii) #PP := #PP + 1;

(c) KPDTP := KPDTAB_ptr

(d) For each keyword parameter

(i) Enter parameter name and default value in KPDTAB[KPDTAB_ptr].

(ii) Enter parameter name in PNTAB[PNTAB_ptr].

(iii) KPDTAB_ptr := KPDTAB_ptr + 1;

(iv) PNTAB_ptr := PNTAB_ptr + 1;

(v) #KP := #KP + 1;

(e) MDTP := MDT_ptr;

(f) #EV := 0;

(g) SSTP := SSTAB_ptr;

3. While not a MEND statement

(a) If an LCL statement then

(i) Enter expansion time variable name in EVNTAB.

(ii) #EV := #EV +1;

(b) If a model statement then

(i) If a label field contains a sequencing symbol then If symbol is present in SSNTAB then

q := entry number in SSNTAB

else

Enter symbol in SSNTAB[SSNTAB_ptr].

q := SSNTAB_ptr;

SSNTAB_ptr := SSNTAB_ptr +1;

SSTAB[SSTP + q - 1] := MDT_ptr

(i) For a parameter, generate the specification (P, #n).

(ii) For an expansion variable, generate the specification (E, #m).

(iii) Record the IC in MDT[MDT_ptr];

(iv) MDT_ptr := MDT_ptr + 1;

(c) If a preprocessor statement then

(i) If a SET statement

Search each expansion time variable name used in the statement in EVNTAB and generate the specification(E,#m)

(ii) If an AIF or AGO statement

If a sequencing symbol used in the statement is present in SSNTAB then

q:= entry number in SSNTAB

else

Enter symbol in SSNTAB[SSNTAB_ptr].

q := SSNTAB_ptr

SSNTAB_ptr := SSNTAB_ptr + 1;

Replace the symbol by (S, SSTP + q - 1)

(iii) Record the IC in MDT[MDT_ptr]

(iv) MDT_ptr := MDT_ptr + 1;

4. (MEND statement)

If SSNTAB_ptr = 1 (SSNTAB is empty) then

SSTP := 0;

else

SSTAB_ptr := SSTAB_ptr + SSNTAB_ptr – 1;

If #KP = 0 then KPDTP = 0;

Algorithm of Macro Expansion:-

1. Perform initialization for the expansion of macro.

(a) MEC := MDTP field of the MNT entry;

(b) Create EVTAB with #EV entries and set EVTAB_ptr.

(c) Create APTAB with #PP + #KP entries and set

APTAB_ptr.

(d) Copy keyword parameter defaults from the entries

KPDTAB[KPDTP]… KPDTAB[KPDTP + #KP - 1]

into APTAB[#PP + 1]… APTAB[#PP+#KP]

(e) Process positional parameter in the actual parameter list and copy them into APTAB[1]…APTAB[#PP]

(f) For keyword parameters in the actual parameter list Search the keyword name in parameter name field of KPDTAB[KPDTP]… KPDTAB[KPDTP + #KP - 1]. Let KPDTP[q] contain a matching entry. Enter value of the keyword parameter in the call in APTAB[#PP + q – KPDTP + 1]

2. <u>While statement pointed by MEC is not MEND statement</u>

(a) If a model statement then

1. Replace the operands of the form (P, #n) and (E, #m) by values in APTAB[n] and EVTAB[m] respectively.
2. Output the generated statement.
3. MEC := MEC + 1;

(b) If a SET statement with the specification (E, #m) in the label field then

1. Evaluate the expression in the operand field and set an appropriate value in EVTAB[m].
2. MEC := MEC + 1;
3. If an AGO statement with (S, #s) in operand field then

MEC := SSTAB[SSTP + s -1];

4. If an AIF statement with (S, #s) in operand field

   then If condition in the AIF statement is true then

   MEC := SSTAB[SSTP + s -1];

Q.9) Explain Macro Assembler.

10) Given is **Macro Definition and its Call**

```
MACRO
INCR    &MEM_VAL, &INCR_VAL, &REG
MOVER   &REG, &MEM_VAL
ADD     &REG, &INCR_VAL
MOVEM   &REG, &MEM_VAL
MEND
```

Macro Call is INCR A, B, AREG

Expand the Macro Definition using Macro Expansion?

11) Given is the Inner and Outer Macro (Nested Macro) Expand the Macro Definition using Macro Expansion?

Inner Macro

```
MACRO
INCR  &MEM_VAL, &INCR_VAL, &REG
MOVER     &REG, &MEM_VAL
ADD  &REG, &INCR_VAL
MOVEM &REG, &MEM_VAL
MEND
```

Outer Macro

```
MACRO
COMPUTE  &FIRST, &SECOND
MOVEM    BREG, TMP
INCR_D   &FIRST, &SECOND, REG=BREG
MOVER    BREG, TMP
MEND
```

12) Write the macro definition for following macro call

EVAL A,B,C to generate efficient code for A-B+C.

13) Following is the macro definition:

```
        MACRO
        CLEARMEM    &X, &N, &REG = AREG
        LCL         &M
&M      SET         0
        MOVEREG     &REG, = '0'
.MORE   MOVEMEM     &REG, &X + &M
&M      SET         &M+1
        AIF         (&M NE N)           .MORE
        MEND
```

Show the contents of the data structures like MDT, MNT, PNTAB, EVNTAB, SSNTAB etc. for the call **CLEARMEM AREA, 10**

14) Following is the macro definition:

```
        MACRO
        CLEARMEM    &X, &N, &REG = AREG
        LCL         &M
&M      SET         0
        MOVEREG     &REG, = '0'
.MORE   MOVEMEM     &REG, &X + &M
&M      SET         &M+1
        AIF         (&M NE N)           .MORE
        MEND
```

Expand the above using expansion of the call CLEARMEM AREA, 10