SYSTEM SOFTWARE

MODULE 2

1. Define macros?

   Macroinstructions (often called Macros) are single line abbriviations for group of instructions of a program.

2. How can we use macro in our program?

   To use a macro, the programmer must define a macro in his program to represent the block of code.For every occurance of this one line macro in his program the macro-processing assembler will substitute the entire block.

3. Define the syntax of a macro definition?

   A macro definition is enclosed between a macro header statement (MACRO) and a macro end (MEND) statement. Macro definitions are typically located at the start of a program.

4. What are the instructions included in a macro definition?

   A **macro definition consists of**

   • A macro prototype statement

   The macro prototype statement declares the name of a macro and the names and kinds of its parameters.The macro prototype statement has the following syntax:

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

   Where <macro name> appeares in the mnemonics field of an assembly statement and <formal parameter spec>is of the form

   &<parameter name>[<parameter kind>]

   • One or more model statements

   A model statement is a statement from which an assembly language statement may be generated during macro expansion.

   • Macro pre-processor statements

   A pre-processor statement is used to perform auxiliary functions during macro expansion.

5. Write an example of macro definition?

   The definition of the macro INCR is shown below

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

MEND

6. What is macro instruction?

   A Macro instruction is a line of computer program coding that results in one or more lines of program coding in the target programming language, sets variables for use by other statements, etc..

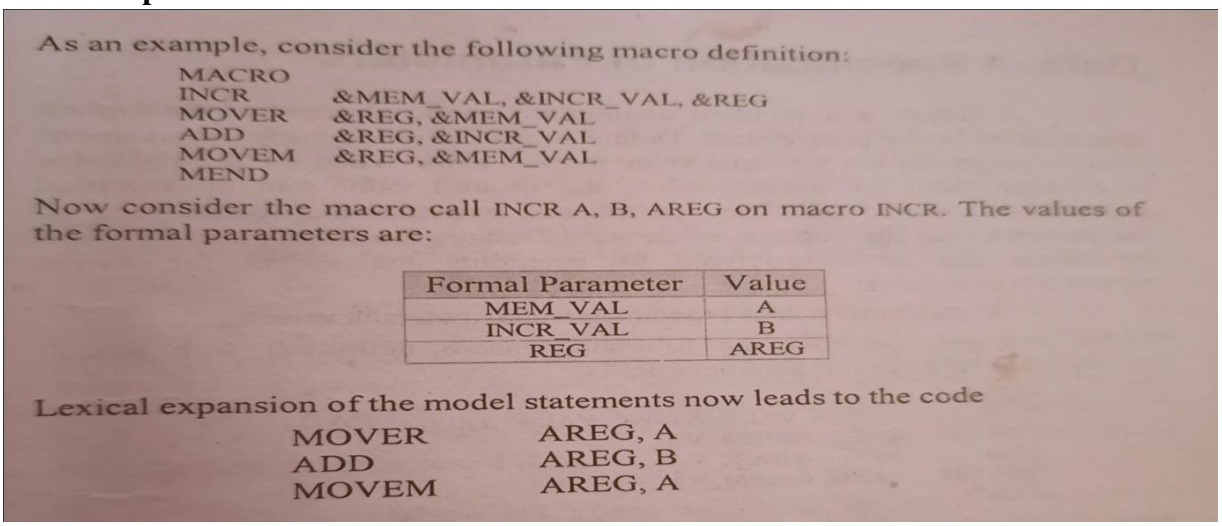
7. What you mean by macro instruction parameters?

   Macro instruction parameters provide a mechanism to pass values from a macro call to macro definition. Value is passed between the actual parameters , which is specified in the macro call statement, and the formal parameters , which is defined in the macro definition.

8. What are the different ways of specifying macro instruction parameters?

   There are generally 2 ways of specifying parameters:

   Positional parameters



As an example, consider the following macro definition:

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

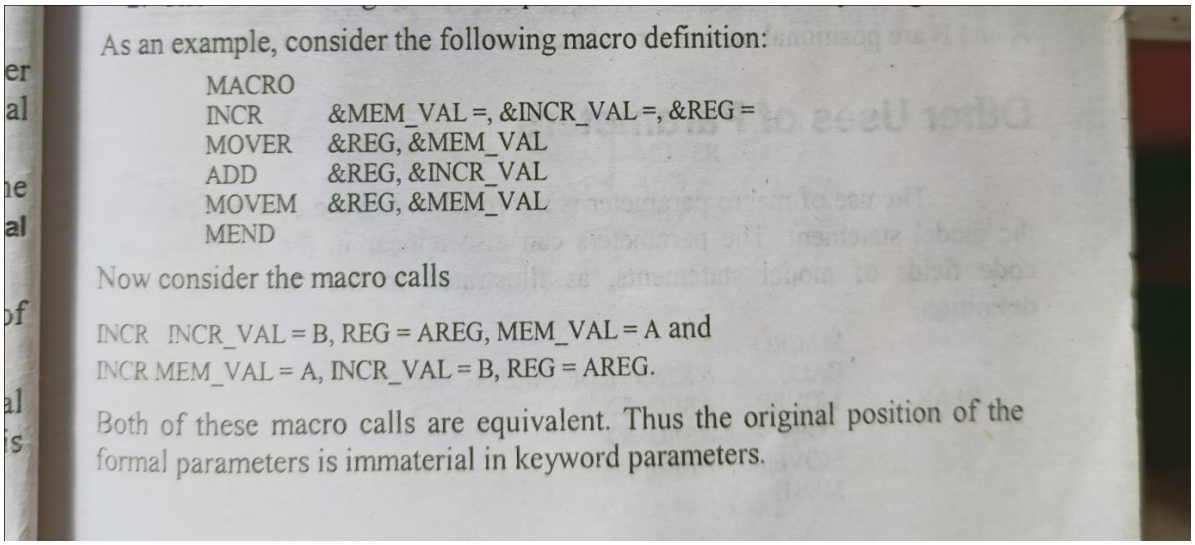
Now consider the macro call INCR A, B, AREG on macro INCR. The values of the formal parameters are:

Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	AREG

Lexical expansion of the model statements now leads to the code

```
MOVER   AREG, A
ADD     AREG, B
MOVEM   AREG, A
```

   Keyword parameters



As an example, consider the following macro definition:

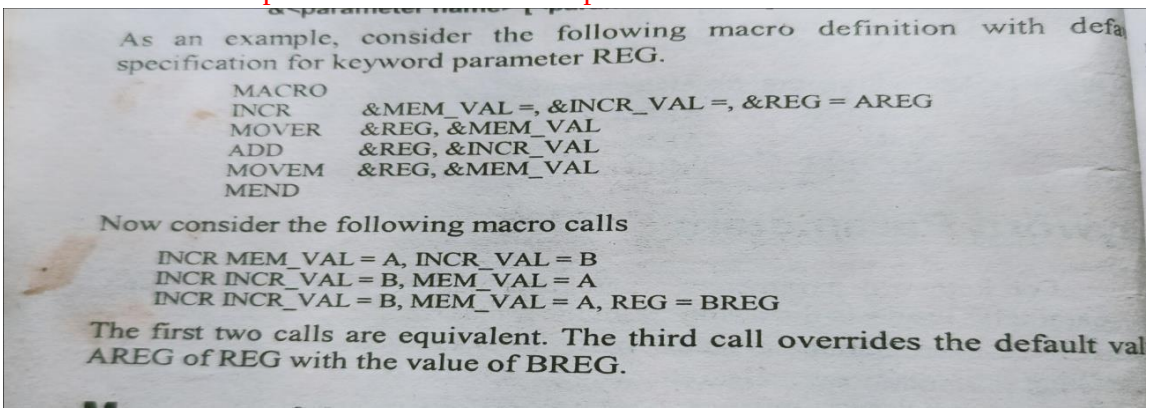
```
MACRO
INCR    &MEM_VAL =, &INCR_VAL =, &REG =
MOVER   &REG, &MEM_VAL
ADD     &REG, &INCR_VAL
MOVEM   &REG, &MEM_VAL
MEND
```

Now consider the macro calls

INCR INCR_VAL = B, REG = AREG, MEM_VAL = A and
INCR MEM_VAL = A, INCR_VAL = B, REG = AREG.

Both of these macro calls are equivalent. Thus the original position of the formal parameters is immaterial in keyword parameters.

9. Illustrate the default parameters with an example?



As an example, consider the following macro definition with defa specification for keyword parameter REG.

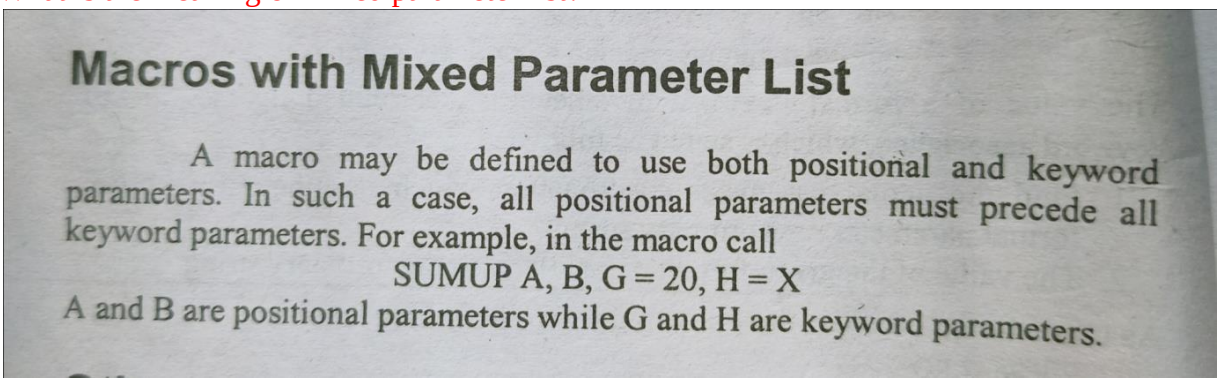
```
MACRO
INCR    &MEM_VAL =, &INCR_VAL =, &REG = AREG
MOVER   &REG, &MEM_VAL
ADD     &REG, &INCR_VAL
MOVEM   &REG, &MEM_VAL
MEND
```

Now consider the following macro calls

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

The first two calls are equivalent. The third call overrides the default val AREG of REG with the value of BREG.

10. What is the meaning of mixed parameter list?



Macros with Mixed Parameter List

A macro may be defined to use both positional and keyword parameters. In such a case, all positional parameters must precede all keyword parameters. For example, in the macro call

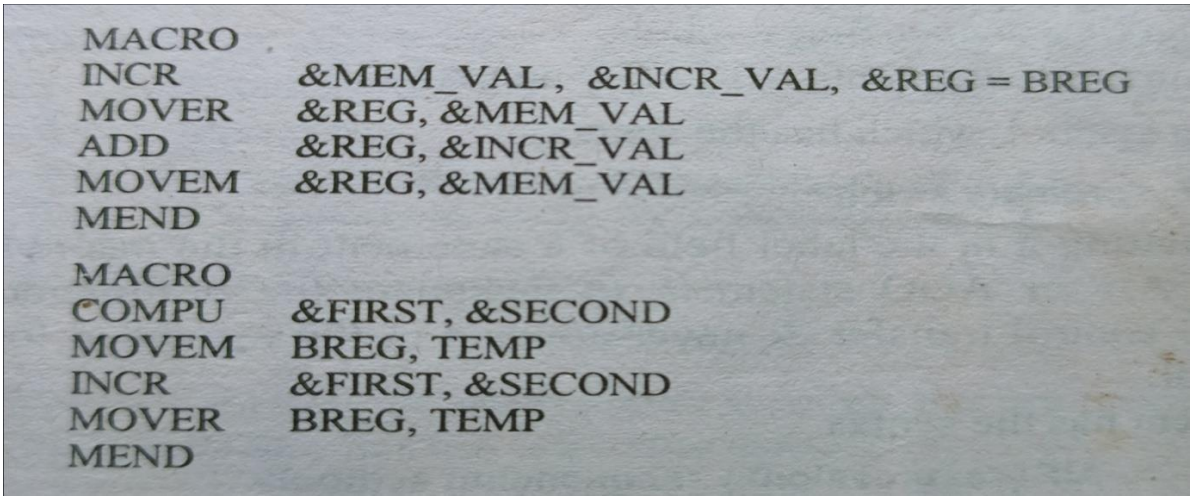
SUMUP A, B, G = 20, H = X

A and B are positional parameters while G and H are keyword parameters.

11. What is nested and recursive macro call?

If a macro call is seen throughout the expansion of a macro, the assembler starts immediately with the expansion of the called macro. For this, its its expanded body lines are simply inserted into the expanded macro body of the calling macro, until the called macro is completely expanded. Then the expansion of the calling macro is continued with the body line following the nested macro call.

Example:



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

MACRO
COMPU   &FIRST, &SECOND
MOVEM   BREG, TEMP
INCR    &FIRST, &SECOND
MOVER   BREG, TEMP
MEND
```

12. What are the advanced macro facilities?

The advanced macro facilities can be grouped into

1. Facilities to alter the flow of control during expansion

2. Expansion time variables
3. Attributes of parameters

1. Altering flow control during expansion(conditional expansion)

It can be achieved through the use of

1.Expansion time sequencing symbols

2. Expansion time statements AIF,AGO and ANOP

A sequencing symbol which has the syntax

.<ordinary string>

is defined by putting it in the label field of statement in the macro body

- An AIF ststement has the syntax

  AIF(<expression>)<sequencing symbol>

  where <expression> is a relational expression

  It is used to specify the branching condition.

  This statements provides conditional branching facility.

- An AGO ststement has the syntax

  AGO<sequencing symbol>

  This statement provides the unconditional branching facilities.

  We do not specify the condition

- ANOP statement is written as

  <sequencing symbol>ANOP

  This statement performs no operations.

2. Expansion time variable

Expansion time variables (EVs) are variables, which can only be used during the expansion of the macro calls.An EV can be local in scop(in which case is created for the use during a particular macro call) or global in scop(in which case it exists across all the macro calls situated in a program)

- Local and global Evs are created through declaration statements with the following syntax:
  - **LCL <EV specification>[,<EV specification>...]**
  - **GBL <EV specification>[,<EV specification>...]**

Where <EV specification> has the syntax

&<EV name>

3. Attributes of formal parameters

- An attribute is written using the syntax

  <attribute name>' <formal parameter spec>

  And represents information about the value of the formal parameter. The type,size and length attributes have the names T,S and L. Example:

```
MACRO
DC L CONST &A
AIF (L'&A EQ 1) .NEXT
.NEXT     ..... .....
          ......
MEND
```

Here expansion time control is transferred to the ststement having .NEXT in the label field only actual parameter corresponding to the formal parameter A has lenth of '1'.

13. Explain about macro processor?

The macroprocessor accepts an assembly language program containing macro definitions and macro calls and translate it into assembly program, which does not contain any macro definition or calls. The program output from the macro processor can now be handed over to an assembler to obtain the target language form of the program. Thus the macro processor separates macro expansion from the process of program assembly.

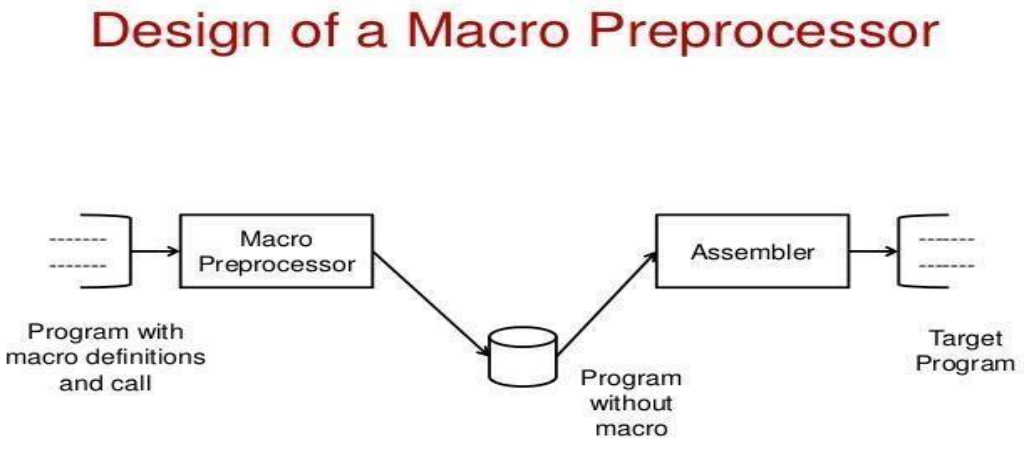


Figure: A Schematic of a macro preprocessor

14. Compare between macro processor and macro assembler?

Macro processor is not efficient as a macro assembler. Because a macro assembler that performs macro expansions as well as assembly. In a macro processor the number of passes over the source program is large and many function get duplicated.Similar functions can be merged if macros are handled by macro assembler, which performs macro expansion and program assembly simultaniously. This may also reduce the number of passes.

15. Explain about the pass structure of a macro assembler?

A macro assembler is an assembler, which performs macro expansion and program assembly simultaniously.The use of a macro processor followed by a conventional assembler is an expensive way of handling macros.The number of passees over the sourse program is large and many functions get duplicated. To design the pass structure of a macro assembler is required to identify the function of a macro processor and the conventional assembler which can be merged to advantage.After merging, the functions can be structured in to passes of the macro assembler. This process leades to the following pass structure:-

- Pass 1: 1.Macro definition processing.
  2.Collect names of declared symbols and information concerning their attributes.
- Pass 2: 1.Macro expansion.
  2. Memmory allocation and LC processing.
  3.Processing of literals.
  4.Intermediate code generation.
- Pass 3: 1.Target code generation.