Macros and Macro Processor

Module 3

Topics to Cover:-

1. Introduction
2. Macro definition and call
3. Features of Macro facility : Simple, parameterized, conditional and nested.
4. Design of Two pass macro processor, data structures used.

INTRODUCTION TO MACROS

1. Macros are used to provide a program generation facility through macro expansion.
2. Many languages provide build-in facilities for writing macros like PL/I, C, Ada & C++.
3. Assembly languages also provide such facilities.

DEFINITON OF MACROS

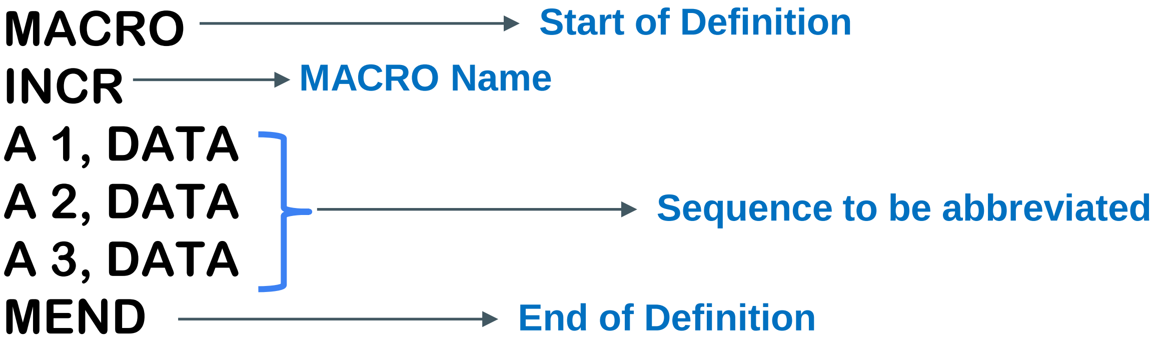
- ❑ **Def:** A macro is a unit of specification for program generation through expansion.
- ❑ A macro consists of ;
  - a) a name,
  - b) a set of formal parameters and
  - c) a body of code.

The use of a macro name with a set of actual parameters is replaced by some code generated from its body. This is called **Macro Expansion.**

Macro definition and call

MACRO DEFINITION: A macro definition is enclosed between a **macro header** statement and a **macro end** statement. They are typically located at the start of a program.

❑ Macro definitions are typically located at the start of a program.



```
MACRO
INCR
A 1, DATA
A 2, DATA
A 3, DATA
MEND
```

Start of Definition

MACRO Name

Sequence to be abbreviated

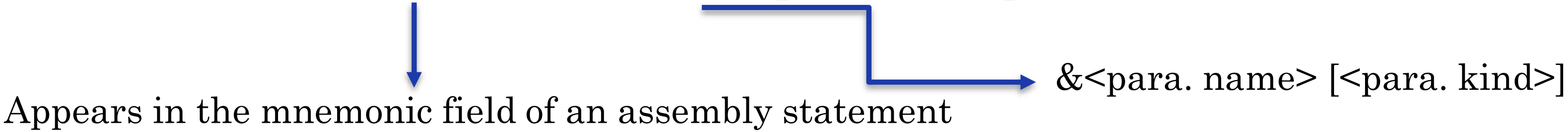
End of Definition

Macro definition and call

- Macro definitions are typically located at the start of a program.
- **A macro definition consists of.**

  a) **A macro prototype statement :** This statement declares the **name of a macro** and the **names and kinds of its parameters.** It has the following syntax

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



&<para. name> [<para. kind>]

Appears in the mnemonic field of an assembly statement

  b) **One or more model statements:-** A model statement is a statement from which an assembly language statement may be generated during Macro expansion

  c) **Macro preprocessor statements:** Is used to perform auxiliary functions during macro expansion.

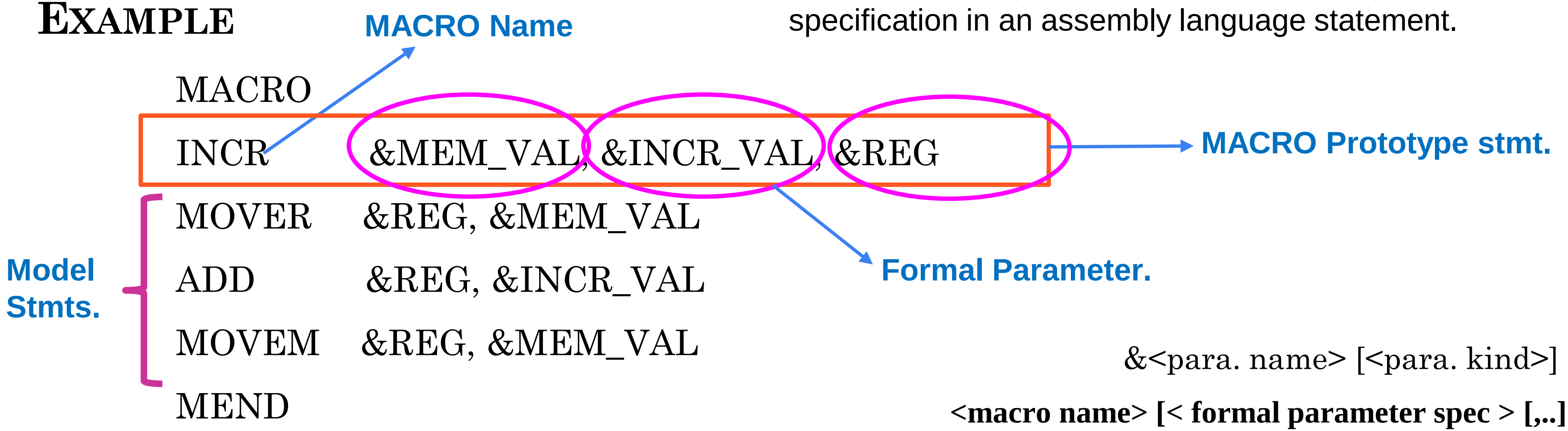
Macro definition and call

Macro Call

- ❑ A macro is called by writing the macro name in the mnemonic field.
- ❑ Macro call has the following syntax. **<macro name> [<actual parameter spec>[,..]]**

where an actual parameter resembles an operand specification in an assembly language statement.

EXAMPLE



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

&<para. name> [<para. kind>]

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

MACRO EXPANSION

- Macro call leads to macro expansion.

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

- To differentiate between the original statements of a program and the statements resulting from Macro Expansion, each expanded statement is marked with a **‘+’ preceding its label field.**

Macro definition and call

Example

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

FORMAL PARAMETERS	ACTUAL PARAMETERS
MEM_VAL	A
INCR_VAL	B
REG	AREG

Consider the following call

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

Macro definition and call

```
MACRO
INCR      &MEM_VAL, &INCR_VAL, &REG
MOVER     &REG, &MEM_VAL
ADD       &REG, &INCR_VAL
MOVEM     &REG, &MEM_VAL
MEND
.
.
.
INCR A, B, AREG
.
.
.
.

INCR A, B, AREG
```

EXPANDED SOURCE CODE

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

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

Features of Macro facility

- Macro Instruction Arguments
- Conditional Macro Expansion
- Macro calls within Macros
- Macro Instruction defining Macros

Features of Macro facility - Macro Instruction Arguments

- A macro instruction just expands the macro definition but does not allow any modification to the macro definition.

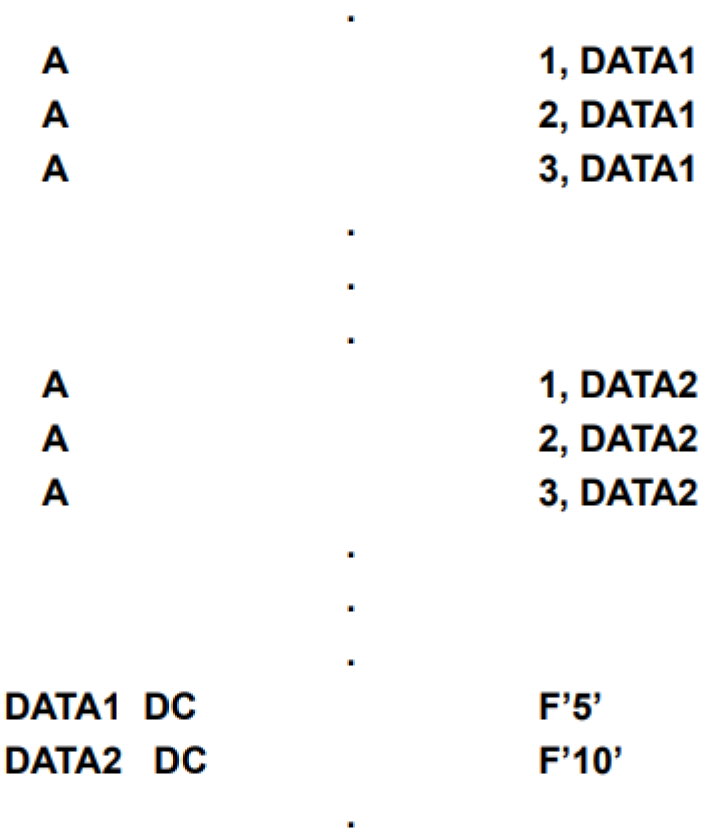
- But we can do the same with the data only by the help of arguments.

- An important extension of Macro facility consists of providing arguments or parameters.

- Macro instruction with argument (dummy arguments) are used in definition. It is specified in the macro name line and distinguished by ‘&’

Features of Macro facility - Macro Instruction Arguments

EXAMPLE



```
                    .
  A                            1, DATA1
  A                            2, DATA1
  A                            3, DATA1
                    .
                    .
                    .
  A                            1, DATA2
  A                            2, DATA2
  A                            3, DATA2
                    .
                    .
                    .
DATA1  DC                      F’5’
DATA2   DC                     F’10’
                    .
```

In this case the instruction sequences are very similar but not identical.

The first sequence performs an operation using DATA1 as operand; the second using DATA2.

They can be considered to perform the same operation with a variable parameter or argument

Such a parameter is called a macro instruction argument or dummy argument dummy argument.

It is specified on the macro name line and distinguished by the ampersand (&), which is always its first character. The preceding program could be written as :

Features of Macro facility - Macro Instruction Arguments

```
•  Source                                                      Expanded Source
   MACRO                    Macro INCR has
                            one argument
   INCR        & ARG
   A           1, &  ARG
   A           2, & ARG
   A           3, & ARG
   MEND
    .                                            .
    .                                            .
    .                                            .
   INCR       DATA 1  Use DATA1 as operand   A          1,DATA 1
    .                                        A          2,DATA1
    .                                        A          3. DATA1
    .                                           .
                                                .
    .
   INCR       DATA 2 Use DATA2 as operand    A          1,DATA 2
    .                                        A          2,DATA2
    .                                        A          3. DATA2
    .
DATA 1   DC             F’5’              DATA1    DC           F’5’
DATA 2   DC             F’10’             DATA2    DC           F’10’
   .
```

Features of Macro facility - Macro Instruction Arguments

Macro Instruction Arguments can be passed using two ways:-

1. **Positional Arguments:-** A positional formal parameter is written as &<parameter name>, e.g. &SAMPLE ; where SAMPLE is the name of parameter.

here **<parameter kind> of syntax rule is omitted.**

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

1. Find the ordinal position of ***XYZ*** in the list of formal parameters in the macro prototype statement.

2. Find the actual parameter specification occupying the same ordinal position in the list of actual parameters in the macro call statement.

Features of Macro facility - Macro Instruction Arguments

Macro Instruction Arguments can be passed using two ways:-

2. **Keyword Arguments**

For keyword parameter,

<parameter name> is an ordinary string and

<parameter kind> is the string '=' in syntax rule.

The <actual parameter spec> is written as

<formal parameter name>=<ordinary string>.

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

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

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

Features of Macro facility - Macro Instruction Arguments

Keyword Arguments -- EXAMPLE

Following is macro definition using keyword parameter:

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

Both are equivalent.

Following are macro call statement:

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

OR

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

Features of Macro facility

- Macro Instruction Arguments - DONE
- Conditional Macro Expansion
- Macro calls within Macros
- Macro Instruction defining Macros

Features of Macro facility - Conditional Macro Expansion

1. While writing a general purpose macro it is important to ensure execution efficiency of its generated code.

2. This is achieved by ensuring that a model statement is visited only under specific conditions during the expansion of a macro.

3. Two features are provided to facilitate alteration of flow of control during expansion. They are **AIF and AGO statements.**

4. **AIF is a conditional branch pseudo opcode, it performs arithmetic test and branch only if condition is TRUE**

5. **AGO is an unconditional pseudo opcode, like a GOTO statement.**

6. These are macro processor directives and they do not appear in expanded source code.

Features of Macro facility - Conditional Macro Expansion

1. An **AIF statement** has the syntax:

AIF (<expression>) <sequencing symbol>

If the **relational expression evaluates to true**, expansion time control is transferred to the statement containing <sequencing symbol> in its label field.

A sequencing symbol (SS) has the syntax; **.**<ordinary string>
As SS is defined by putting it in the label field of statement in the macro body.

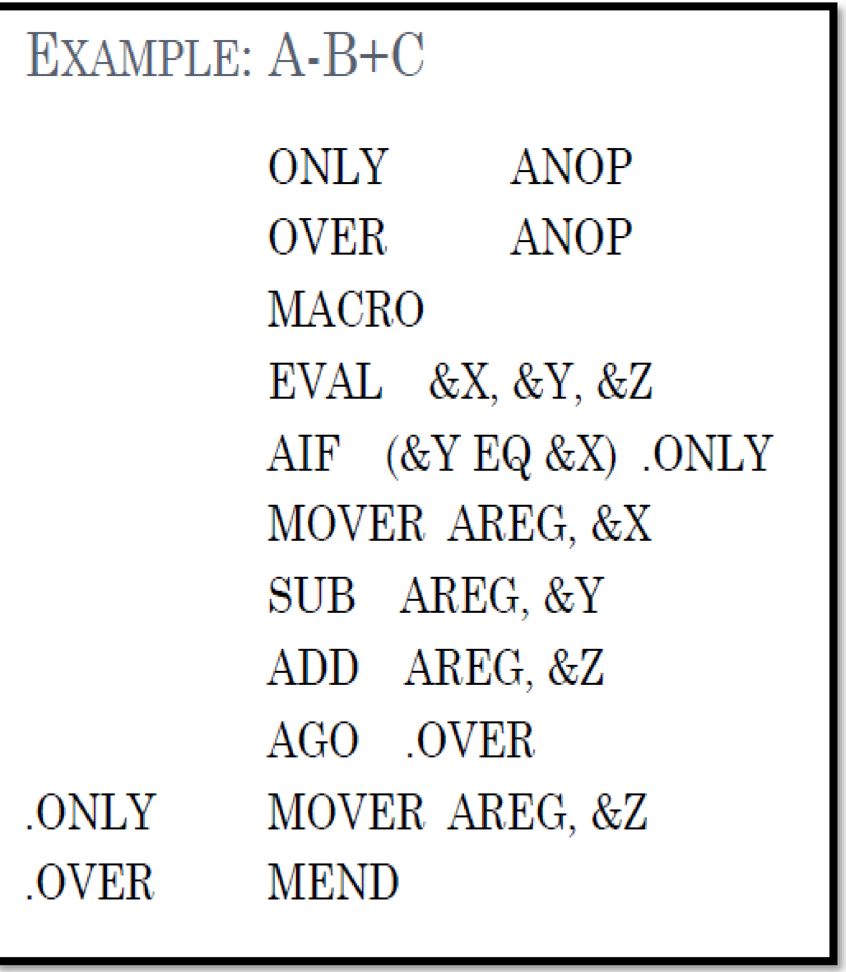
2. An **AGO statement** has the syntax:

AGO <sequencing symbol>

Unconditionally transfers expansion time control to the statement containing <sequencing symbol> in its label field.

3. No operation is carried out by an **ANOP instruction.** It is used to define the sequencing symbol. An ANOP statement is written as

<sequencing symbol> ANOP



EXAMPLE: A-B+C

```
            ONLY      ANOP
            OVER      ANOP
            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
```

A macro **EVAL** is developed such that a **call EVAL** A,B,C generates efficient code to evaluate A-B+C in AREG.

When the first two parameters of a call are identical, EVAL should generate single MOVER instruction to load 3rdparameter into AREG.

AIF statement effectively compares names of first two actual parameters. If condition is true, expansion time control is transferred to model statement MOVER AREG, &Z.

If false, MOVE-SUB-ADD sequence is generated and expansion time control is transferred to statement .OVER MEND which terminates expansion. Thus, efficient code is generated under all conditions.

Features of Macro facility

- Macro Instruction Arguments - DONE
- Conditional Macro Expansion - DONE
- Macro calls within Macros
- Macro Instruction defining Macros

Features of Macro facility - Macro calls within Macros

- Also Known as NESTED MACRO CALLS

- A model statement in macro may constitute a call on another macro, such calls are known as Nested Macro Calls.

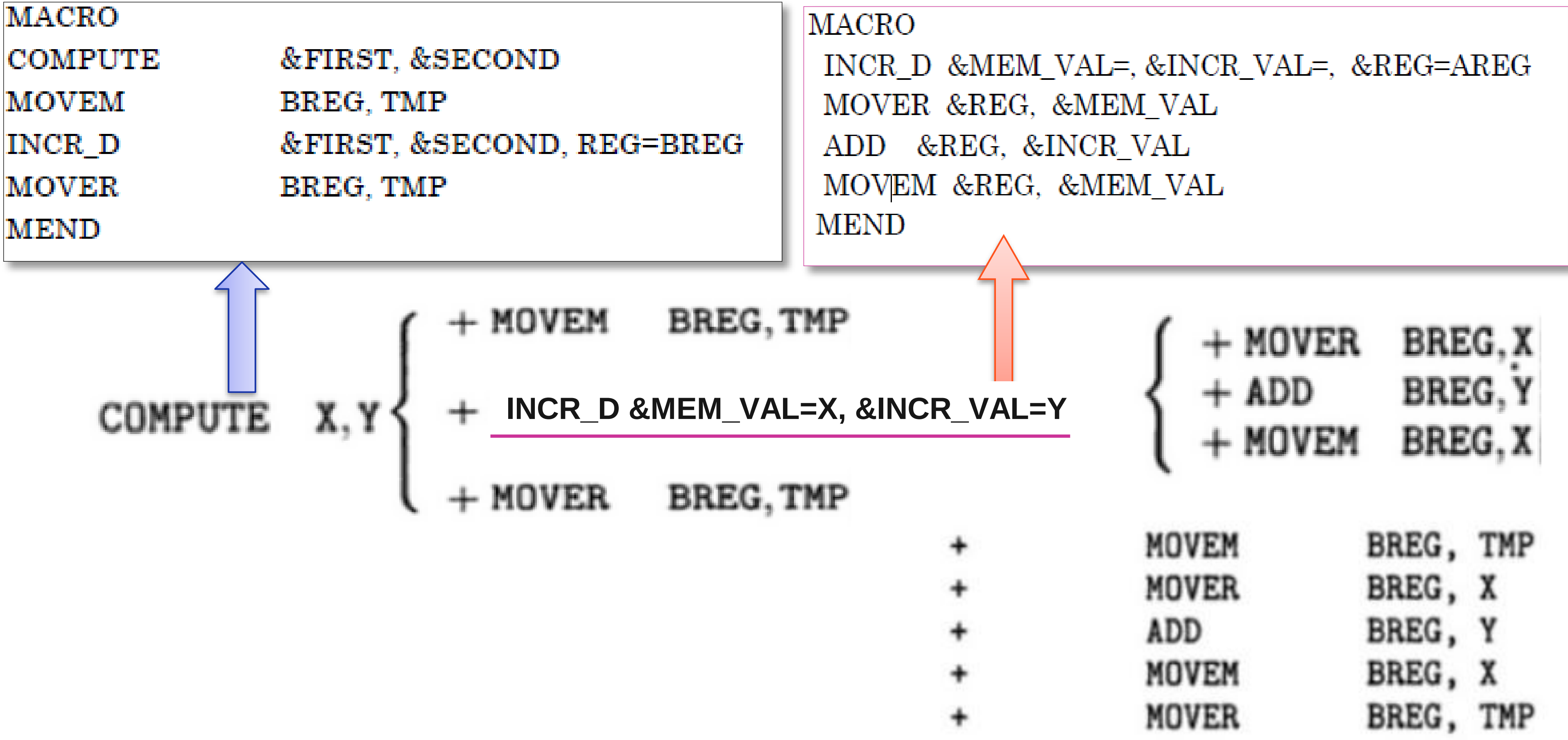
- The macro containing the nested call is called Outer Macro.

- The called macro called Inner Macro.

- Expansion of nested macro calls follows the Last-In-First-Out(LIFO) rule.

Features of Macro facility - Macro calls within Macros

EXAMPLE



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

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

```
                 + MOVEM     BREG, TMP
COMPUTE   X, Y { + INCR_D &MEM_VAL=X, &INCR_VAL=Y      { + MOVER   BREG, X
                                                         + ADD     BREG, Y
                                                         + MOVEM   BREG, X
                 + MOVER     BREG, TMP
```

```
+          MOVEM        BREG,  TMP
+          MOVER        BREG,  X
+          ADD          BREG,  Y
+          MOVEM        BREG,  X
+          MOVER        BREG,  TMP
```

Features of Macro facility

- Macro Instruction Arguments - DONE
- Conditional Macro Expansion - DONE
- Macro calls within Macros - DONE
- Macro Instruction defining Macros

Features of Macro facility - Macro Instruction defining Macros

- Macros can be defined within a macro.
- Inner macro definition is not defined until after the outer macro has been called.
- Group of macros can be defined for subroutine calls with some standardized calling sequence.
- Individual macros have names of the associated subroutines (as given by the argument &SUB).

Macro Instruction defining Macros

```
                          MACRO
                          DEFINE &SUB              Macro name: DEFINE
                               MACRO
                               &SUB      &Y        Dummy macro name
Definition                     CNOP      0,4       Align boundary
of macro       Definition      BAL       1,*+8          Set reg 1 to parameter list pointer
DEFINE         of macro        DC        A(&Y)          Parameter list pointer
               &SUB            L         15,=V(&SUB)    Address of subroutine
                               BALR      1    4,15      Transfer control to subroutine
                               MEND
                          MEND
```

```
DEFINE   COS
COS  AR
BAL  1,*+8
DC   A(AR)          Address of AR
L    15,=V(COS)     V denotes Address of external symbol
BALR      14,15
```

Design of Two pass macro processor, data structures used.

General Design Steps

1. Specification of Problem
2. Specification of databases
3. Specification of database formats
4. Algorithm

Implementation of Macro Processor

There are 4 basic tasks that a macro processor must perform;

1. **Recognize Macro Definition:** A macro processor must recognize macro definition identified by MACRO & MEND pseudo-operations.

2. **Save the Definition:** The processor must store the macro-definitions which will be needed at the time of expansion of calls

3. **Recognize Macro Call:** The processor must recognize macro calls that appear as mnemonics.

4. **Expand calls & Substitute arguments:** The macro call is replaced by macro definition and dummy / formal arguments are replaced by actual data.

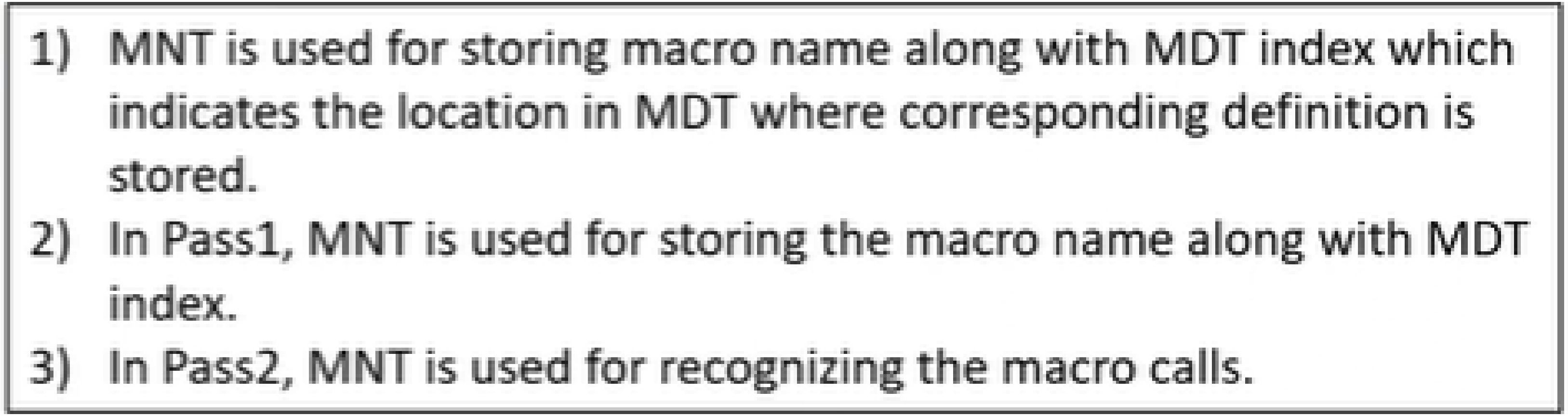
Specification of databases used in PASS 1

1. Input macro source program
2. Output macro source program to be used for pass 2.
3. **Macro Definition Table (MDT)** used to store body of macro definition.
4. **Macro Name Table (MNT)** used to store names of macros
5. **Macro Definition Table Counter (MDTC)** used to mark next available entry in the MDT.
6. **Macro Name Table Counter (MNTC)** used stores next available entry in the MNT.
7. **Argument List Array (ALA)** used to substitute index markers for dummy arguments before storing macro definition

FORMAT OF DATABASES

MNTC

MACRO NAME TABLE (MNT)		
INDEX	MACRO NAME	MDT INDEX



1) MNT is used for storing macro name along with MDT index which indicates the location in MDT where corresponding definition is stored.
2) In Pass1, MNT is used for storing the macro name along with MDT index.
3) In Pass2, MNT is used for recognizing the macro calls.

FORMAT OF DATABASES

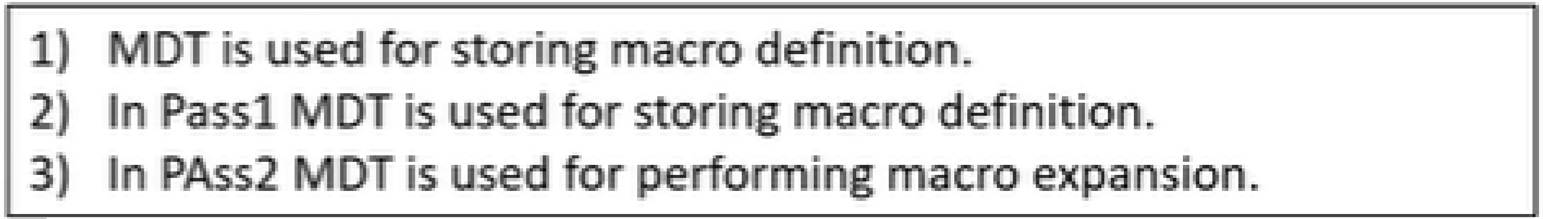
MDTC

MACRO DEFINTION TABLE (MDT)

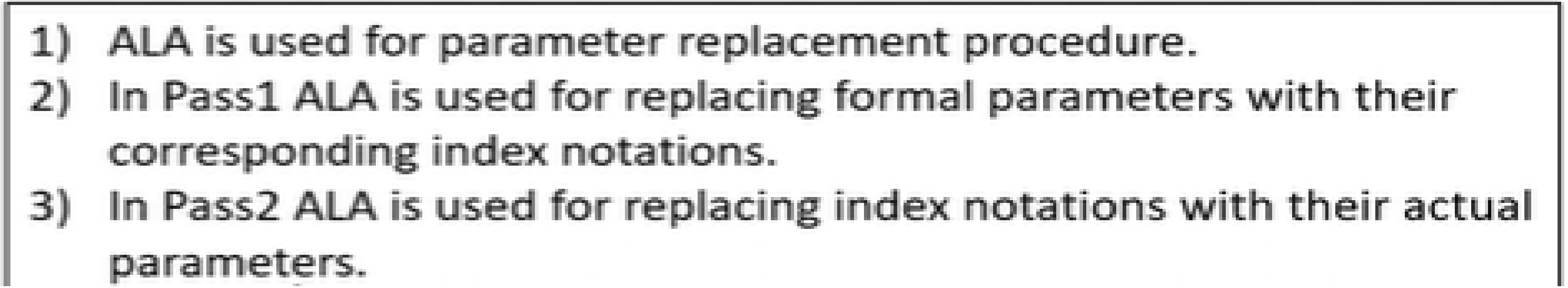
INDEX	MACRO DEFINTION

ARGUMENT LIST ARRAY (ALA)

INDEX	ARGUMENTS

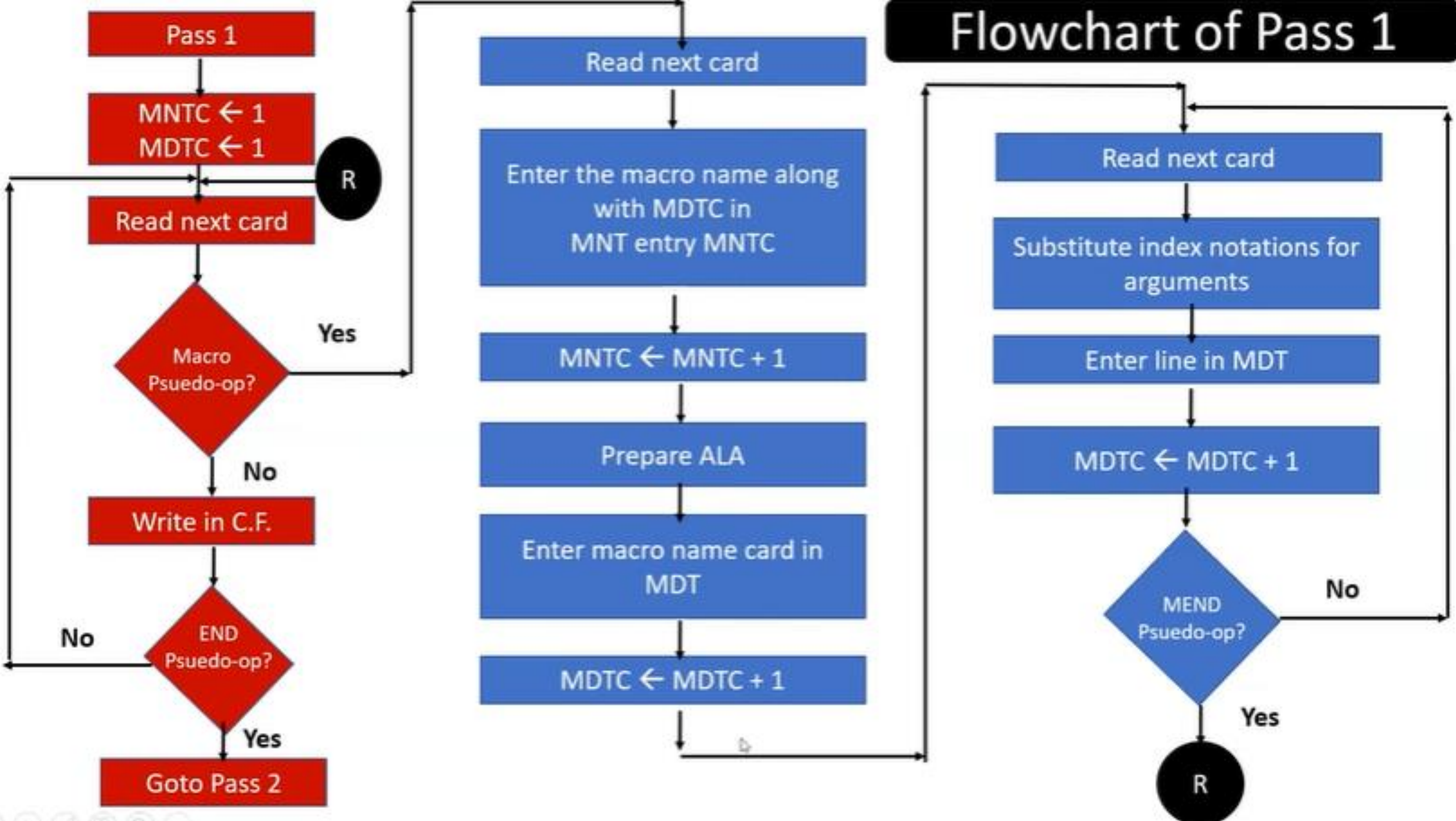


1) MDT is used for storing macro definition.
2) In Pass1 MDT is used for storing macro definition.
3) In PAss2 MDT is used for performing macro expansion.



1) ALA is used for parameter replacement procedure.
2) In Pass1 ALA is used for replacing formal parameters with their corresponding index notations.
3) In Pass2 ALA is used for replacing index notations with their actual parameters.

Flowchart of Pass 1



Specification of databases used in PASS 2

1. Copy of input from PASS 1

2. Output expanded source will serve as an input for assembler

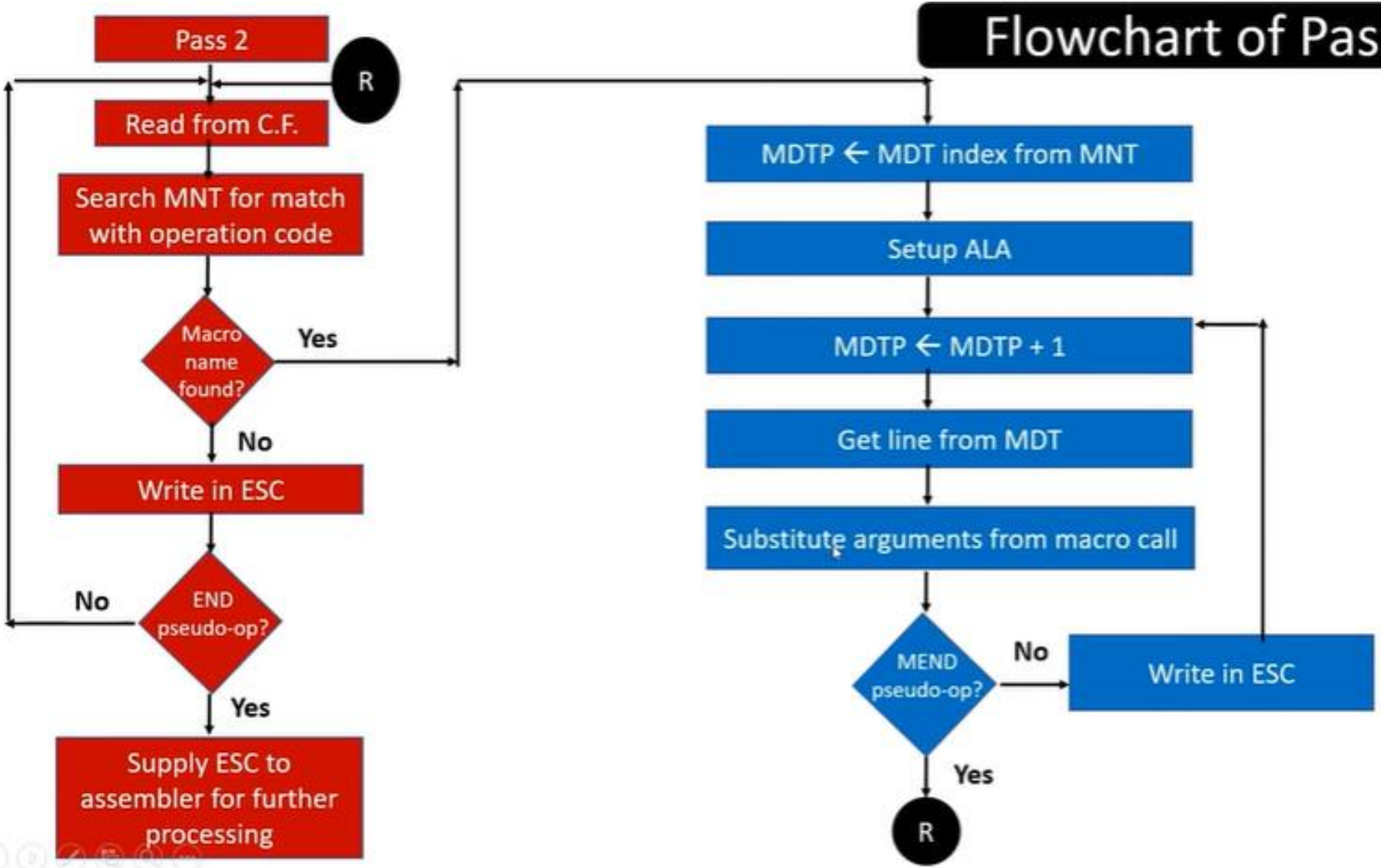
3. MDT – created by PASS 1

4. MNT – created by PASS 1

5. ALA is used to substitute macro call arguments for the index markers in the stored macro definitions.

6. Macro Definition Table Pointer (MDTP) indicates the next line of text to be used during macro expansion

Flowchart of Pass 2



UNIVERSSITY QUESTIONS

- Explain different features of macros. [05] or [10]
- With reference to macroprocessor, explain the following tables with suitable example. (10)
  - (i) MNT
  - (ii) MDT
  - (iii) ALA
- Explain the working of two pass macro processor with neat flowcharts and databases. (Clearly show entries in databases.) [10]
- Parameterized Macros – SN [05]
- Explain with example conditional macro expansion [10]
- Explain with the help of flowchart , the first pass of two pass macro-processor [10]
- SN on Macro Facility [05]
- Explain Macro and Macro Expansion [05]