Chapter 3:
Macro and Macro Processors

Mrs. Sunita M Dol (Aher),
Assistant Professor,
Computer Science and Engineering Department,
Walchand Institute of Technology, Solapur, Maharashtra

Chapter 3.

Macro and Macro Processors

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

Introduction

- Macro: is a unit of specification for program generation through expansion.

- Many languages provide built-in facilities for writing macros. E.g. Ada, C and C++

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

Macro Definition

- Macro definition is enclosed between a macro 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.

Macro Definition

- The macro prototype statement has syntax:

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

- Formal parameter specification has form

  &<*parameter name*>[<*parameter kind*>]

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*>[,..]]

Macro Definition and Call

- 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

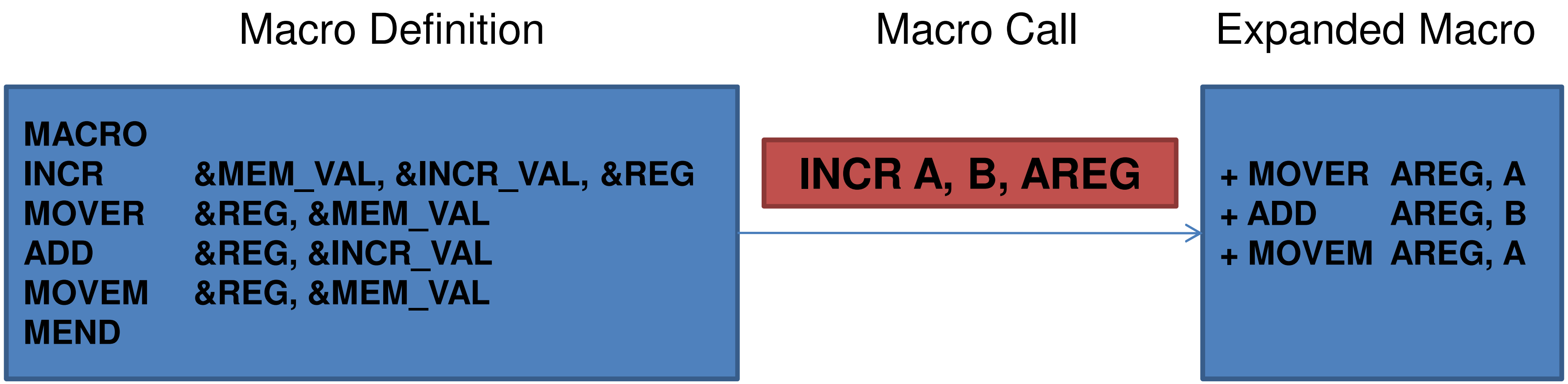
```
INCR      A, B, AREG
```

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

Macro Expansion

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



Macro Definition

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

Macro Call

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

Expanded Macro

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

Macro Expansion

- 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.

Macro Expansion

- 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

Macro Expansion

- 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
```

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.

Macro Expansion

- The rule for determining the value of formal parameter depend on kind of parameter.

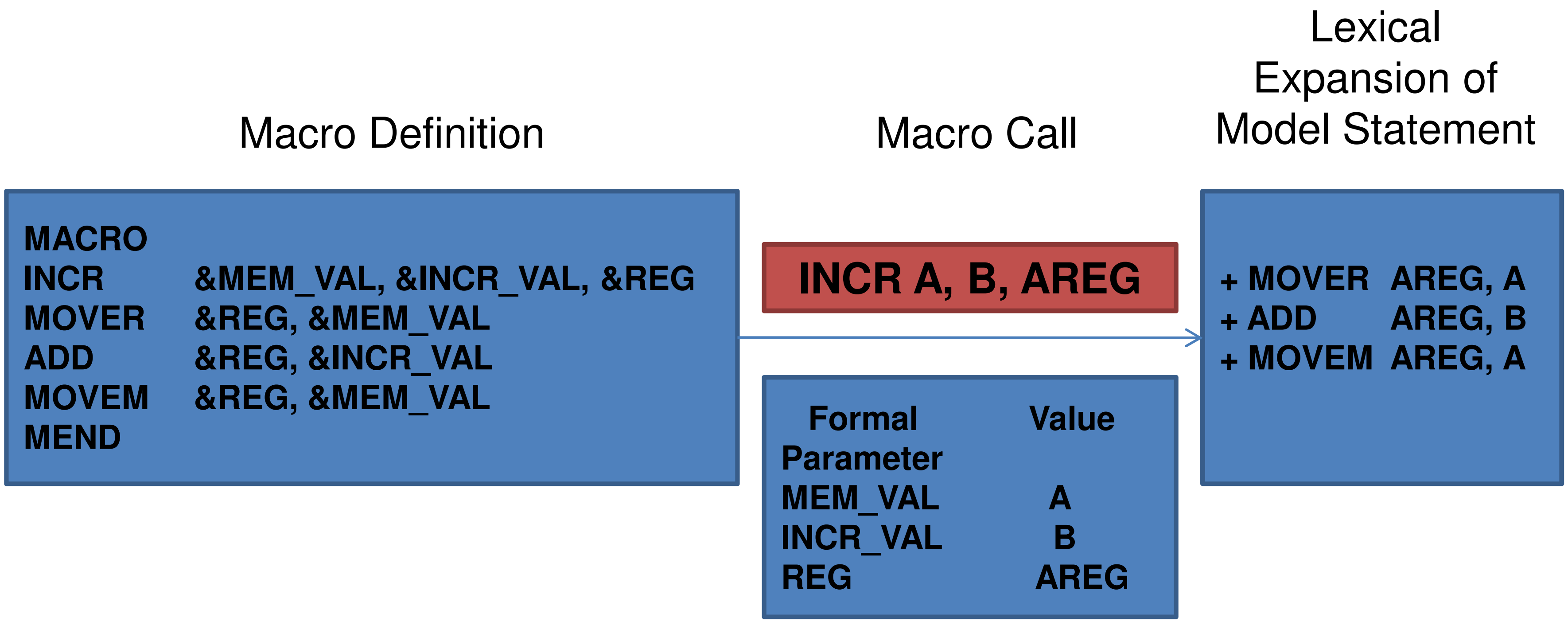
1. Positional Parameter
2. Keyword Parameter
3. Default Specification of Parameter
4. Macros with Mixed Parameter
5. Other uses of Parameter

Macro Expansion

- 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.

Macro Expansion

- Positional Parameter Example



Macro Definition

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

Macro Call

INCR A, B, AREG

Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	AREG

Lexical Expansion of Model Statement

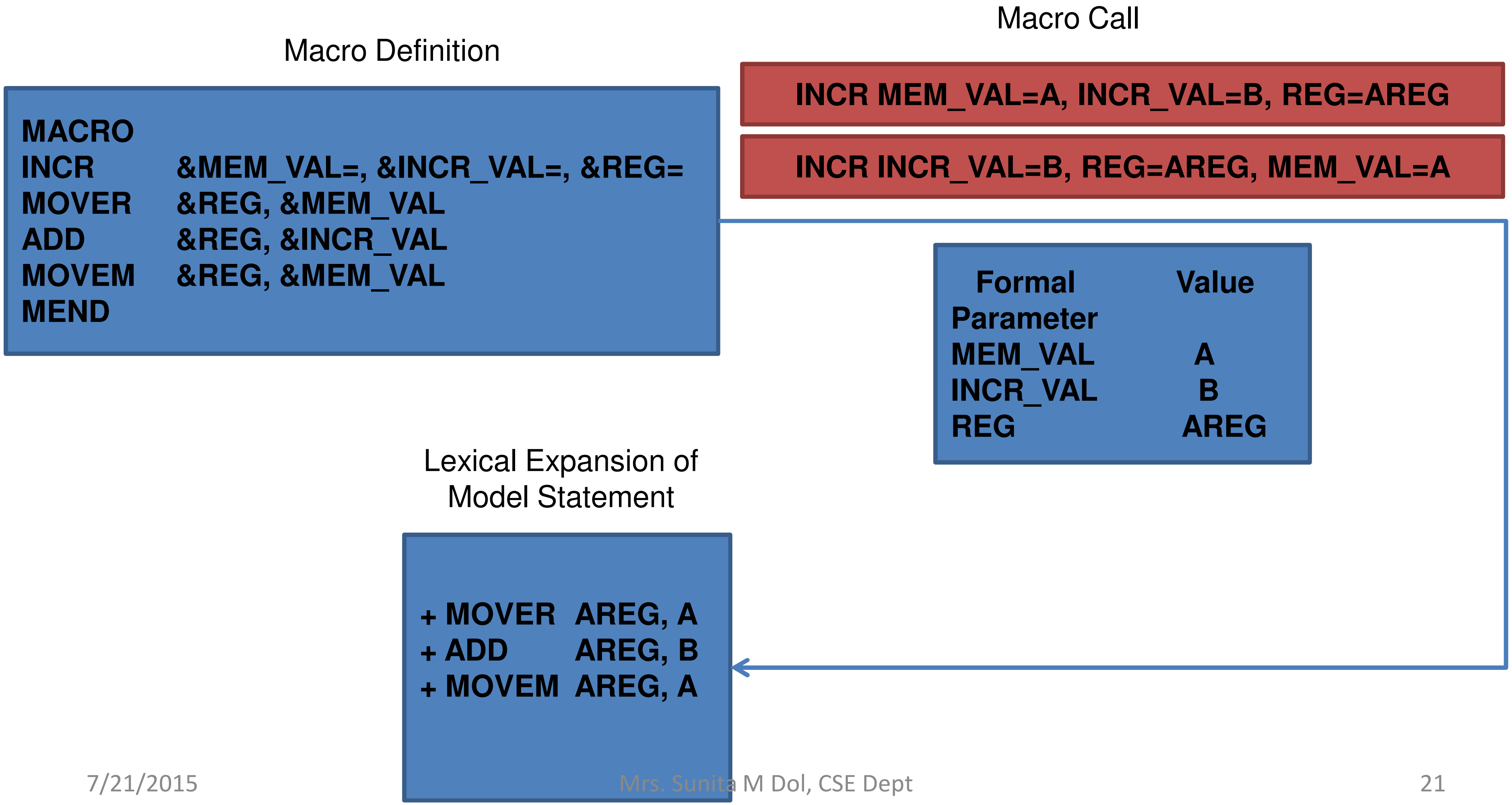
```
+ MOVER  AREG, A
+ ADD    AREG, B
+ MOVEM  AREG, A
```

Macro Expansion

- 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.

Macro Expansion

- Keyword Parameter Example



Macro Call

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

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

Macro Definition

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

Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	AREG

Lexical Expansion of Model Statement

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

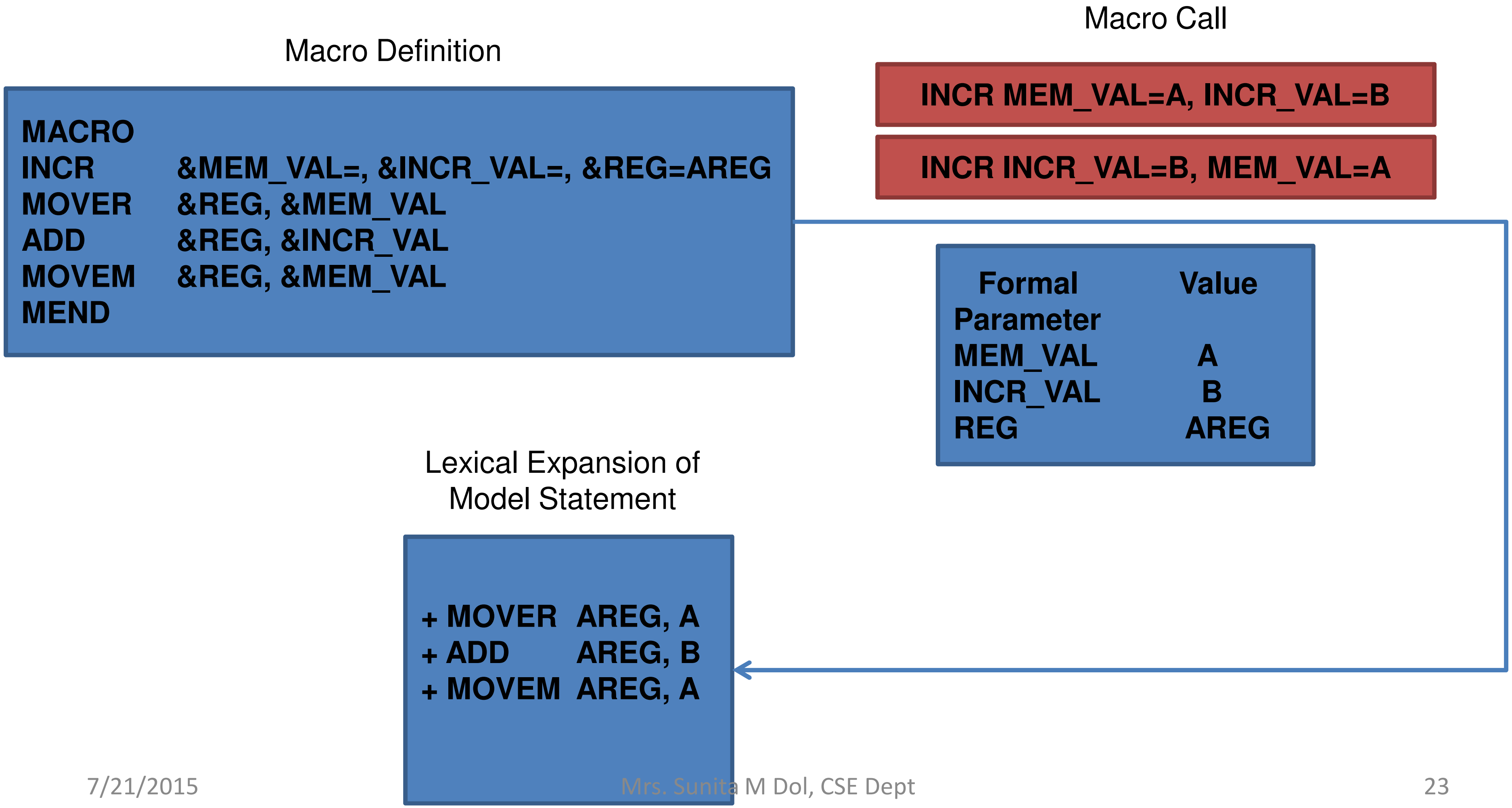
Macro Expansion

- 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*>]]

Macro Expansion

- Default Specification of Parameter Example



Macro Definition

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

Macro Call

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

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

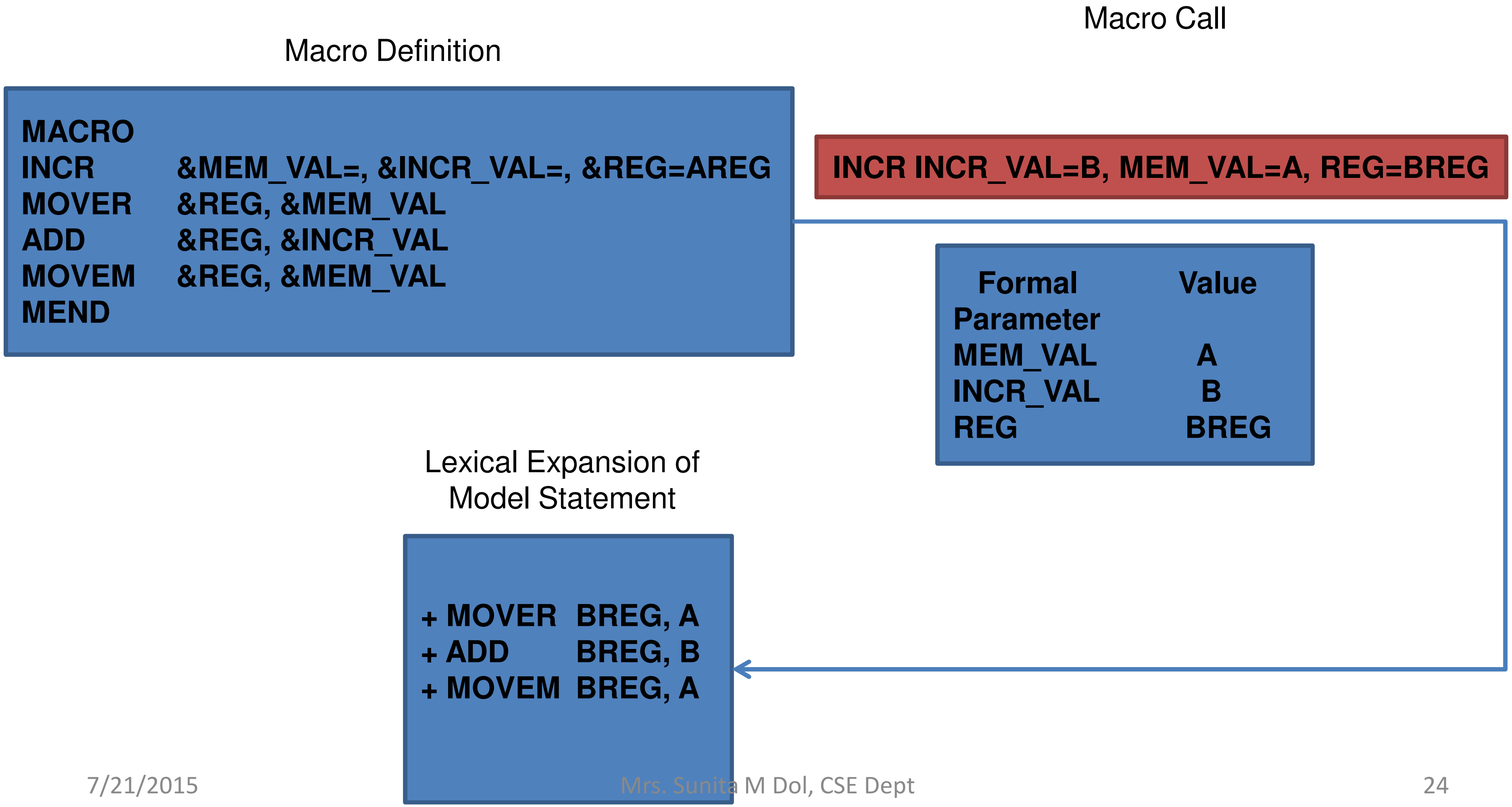
Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	AREG

Lexical Expansion of Model Statement

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

Macro Expansion

- Default Specification of Parameter Example



Macro Definition

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

Macro Call

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

Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	BREG

Lexical Expansion of Model Statement

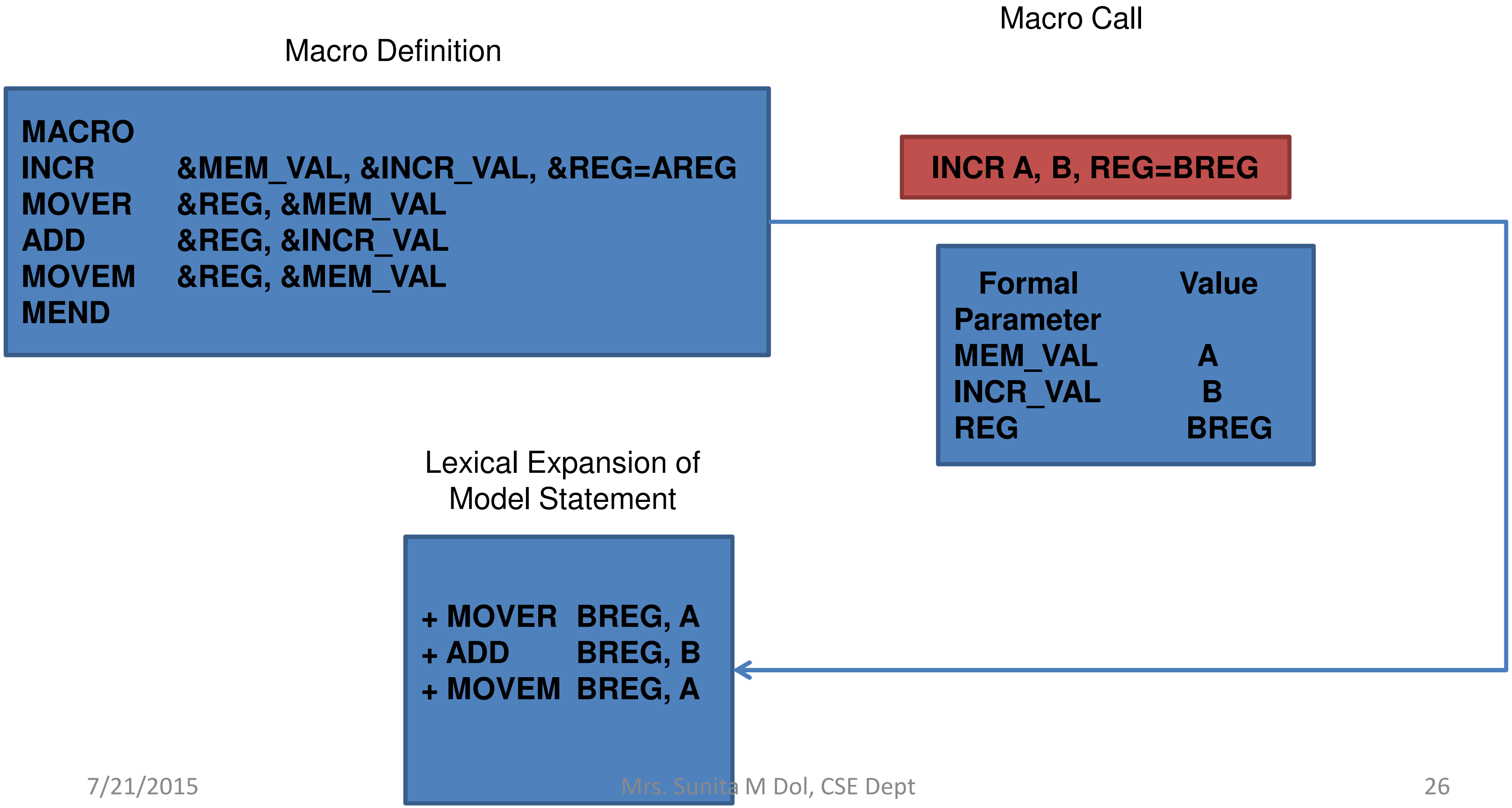
```
+ MOVER  BREG, A
+ ADD    BREG, B
+ MOVEM  BREG, A
```

Macro Expansion

- 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.

Macro Expansion

- Macros with mixed parameter list Example



Macro Call

Macro Definition

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

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

Formal Parameter	Value
MEM_VAL	A
INCR_VAL	B
REG	BREG

Lexical Expansion of Model Statement

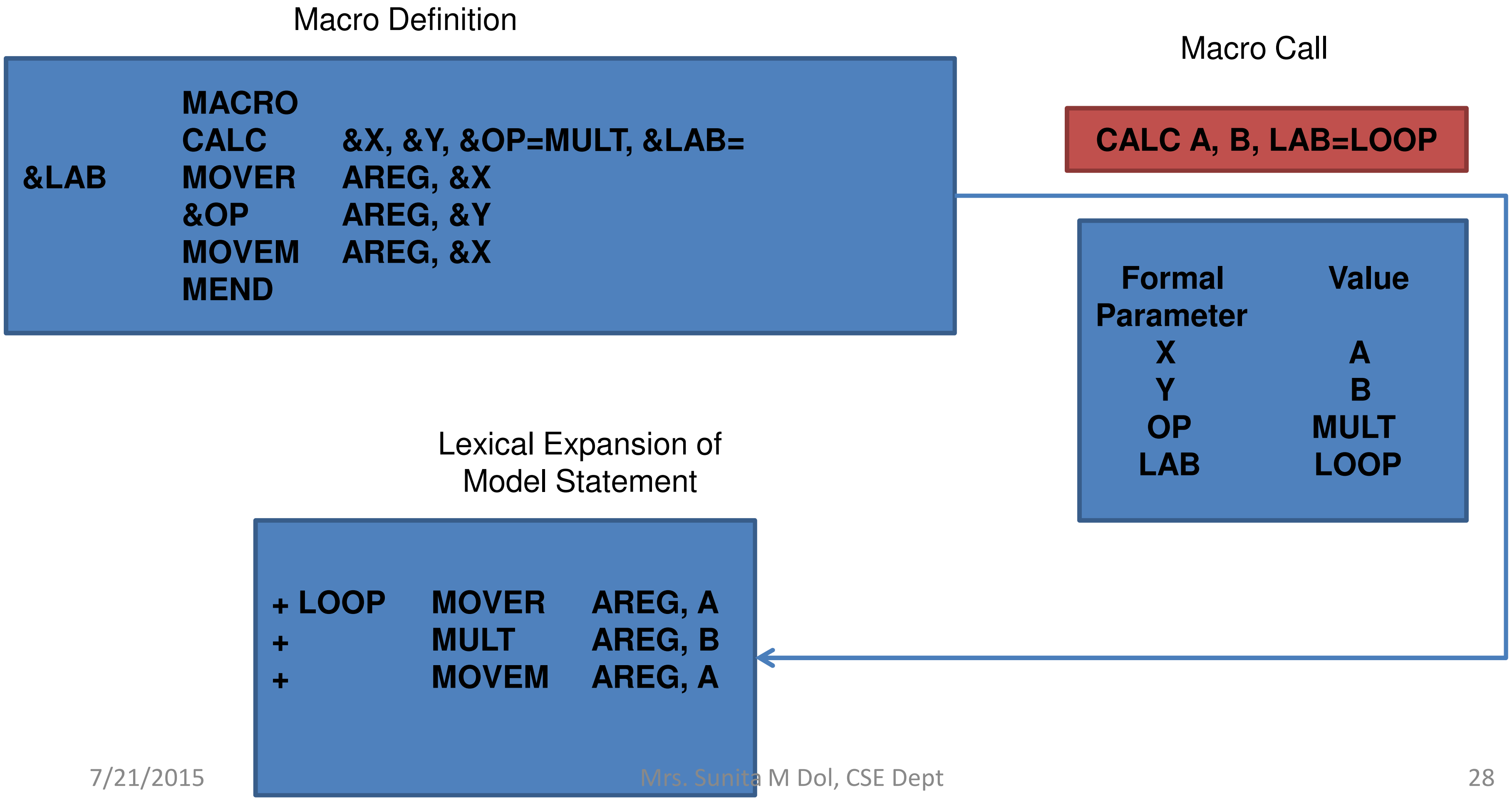
```
+ MOVER  BREG, A
+ ADD    BREG, B
+ MOVEM  BREG, A
```

Macro Expansion

- Other uses of parameters
  - Formal parameter can also appear in the label and opcode fields of model statement.

Macro Expansion

- Other uses of Parameter Example



Macro Definition

```
        MACRO
        CALC      &X, &Y, &OP=MULT, &LAB=
&LAB    MOVER     AREG, &X
        &OP       AREG, &Y
        MOVEM     AREG, &X
        MEND
```

Macro Call

```
CALC A, B, LAB=LOOP
```

Formal Parameter	Value
X	A
Y	B
OP	MULT
LAB	LOOP

Lexical Expansion of Model Statement

```
+ LOOP    MOVER     AREG, A
+         MULT      AREG, B
+         MOVEM     AREG, A
```

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

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.

Nested Macro Calls

- 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

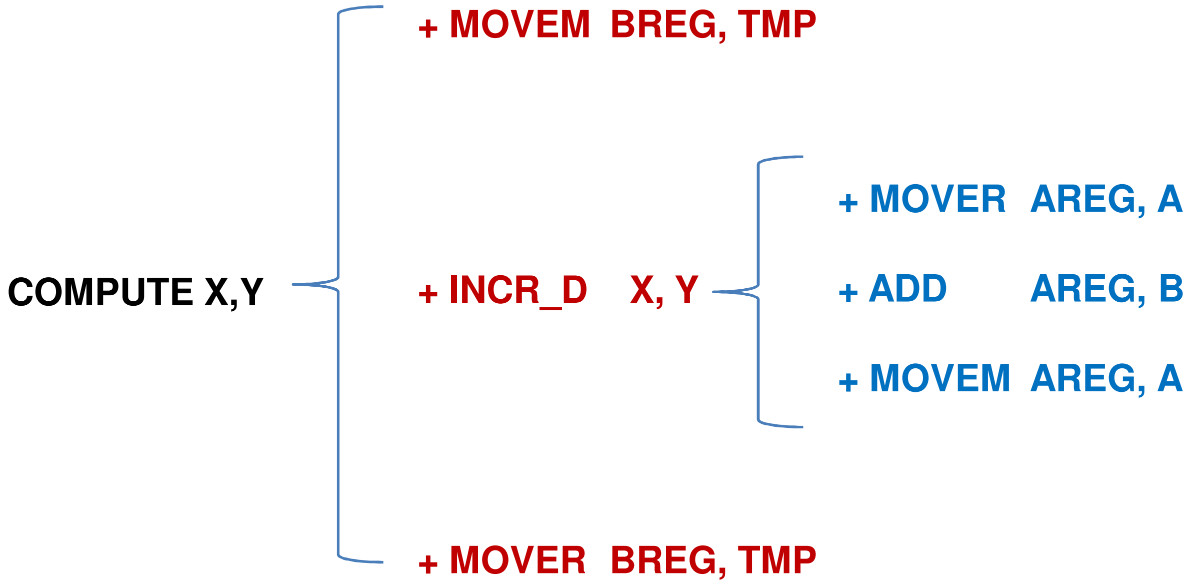
Nested Macro Calls

```
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



COMPUTE X,Y

+ MOVEM BREG, TMP

+ INCR_D   X, Y

+ MOVER AREG, A

+ ADD   AREG, B

+ MOVEM AREG, A

+ MOVER BREG, TMP

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

Advanced Macro Facilities

- These facilities support semantic expansion.
- These facilities can be grouped into
  a) Facilities for alteration of flow of control during expansion
  b) Expansion time variables
  c) Attributes of parameters

Advanced Macro Facilities

- a) Facilities for alteration of flow of control during expansion
  - Two features are provided to facilitate alteration of flow of control during expansion
    - a.1 Expansion time sequencing symbol
    - a.2 Expansion time statements AIF, AGO and ANOP

Advanced Macro Facilities

a.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.

Advanced Macro Facilities

a.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

Advanced Macro Facilities

- 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
```

Advanced Macro Facilities

- 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.

Advanced Macro Facilities

- b) Expansion time variables (EV's)
  - Value of EV's can be manipulated through the preprocessor statement SET which is written as

    <EV specification> SET <SET-expression>

Advanced Macro Facilities

- b) Expansion time variables (EV's)-Example

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

Advanced Macro Facilities

- 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

Advanced Macro Facilities

- c) Attributes of parameters Example

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

Advanced Macro Facilities

- 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.

Advanced Macro Facilities

- Conditional Expansion Example

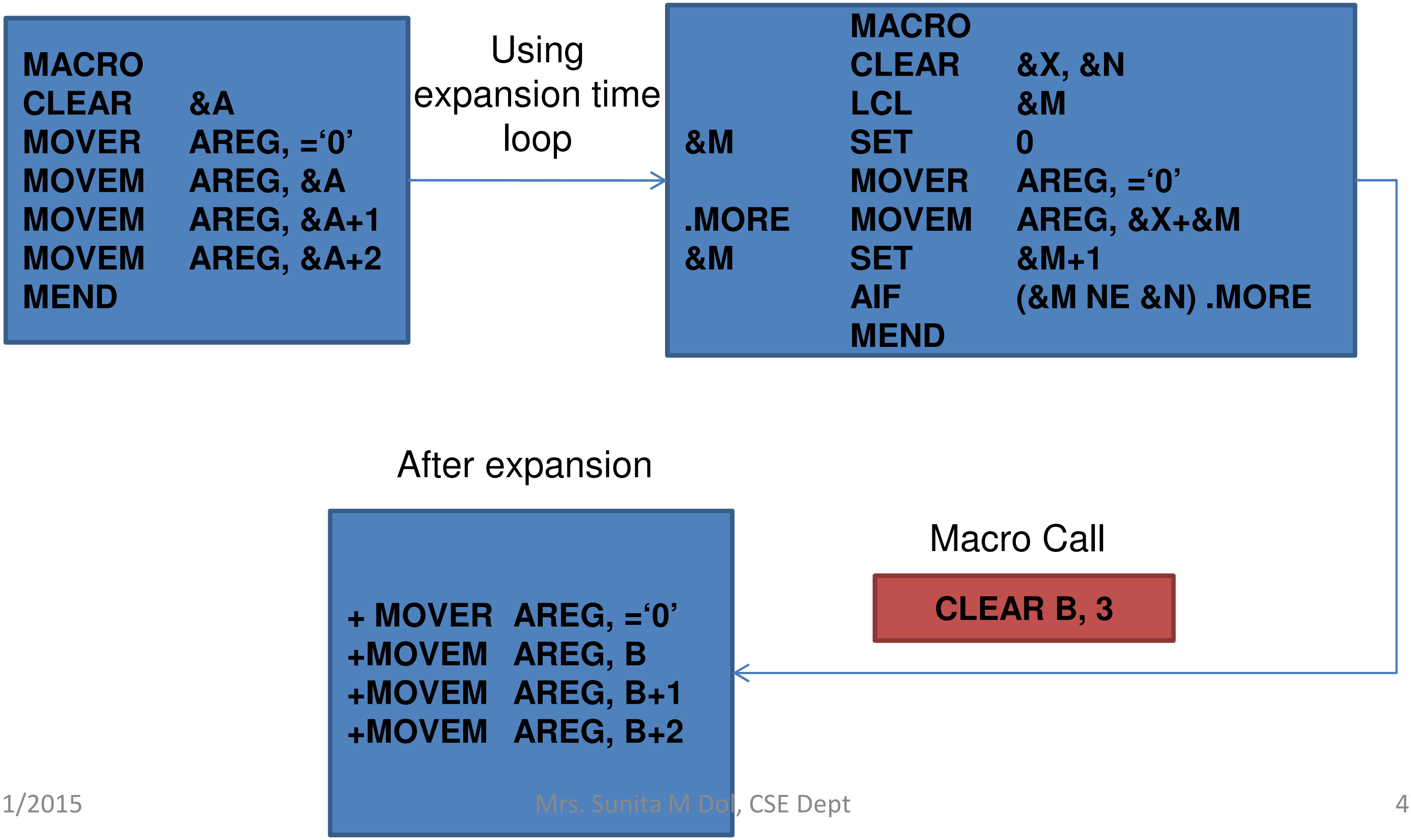
```
          MACRO
          EVAL   &X, &Y, &Z
          AIF         (&Y EQ &X)  .ONLY
          MOVER       AREG, &X
          SUB         AREG, &Y
          ADD         AREG, &Z
          AGO         .OVER
.ONLY     MOVER       AREG, &Z
.OVER     MEND
```

Advanced Macro Facilities

- 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.

Advanced Macro Facilities

- Expansion time loops Example



```
MACRO
CLEAR      &A
MOVER      AREG, ='0'
MOVEM      AREG, &A
MOVEM      AREG, &A+1
MOVEM      AREG, &A+2
MEND
```

Using expansion time loop

```
            MACRO
            CLEAR      &X, &N
            LCL        &M
&M          SET        0
            MOVER      AREG, ='0'
.MORE       MOVEM      AREG, &X+&M
&M          SET        &M+1
            AIF        (&M NE &N) .MORE
            MEND
```

Macro Call

```
CLEAR B, 3
```

After expansion

```
+ MOVER  AREG, ='0'
+MOVEM   AREG, B
+MOVEM   AREG, B+1
+MOVEM   AREG, B+2
```

Advanced Macro Facilities

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

Advanced Macro Facilities

- 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.

Advanced Macro Facilities

- The RPET statement Example

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

Advanced Macro Facilities

- The IRP statement
  - The IRP statement has the syntax

    IRP <*formal parameter*>, <*argument-list*>
  - The formal parameter mentioned in the statement takes successive values from the argument list.
  - For each value, the statement between the IRP and ENDM statements are expanded once.

Advanced Macro Facilities

- The IRP statement Example

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

Advanced Macro Facilities

- 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.

Advanced Macro Facilities

- Semantic Expansion Example

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

3. Macro and Macro Processors

- Introduction
- Macro Definition and Call
- Macro Expansion
- Nested Macro Calls
- Advanced Macro Facilities
- Design of Macro preprocessor

Design of a Macro Preprocessor

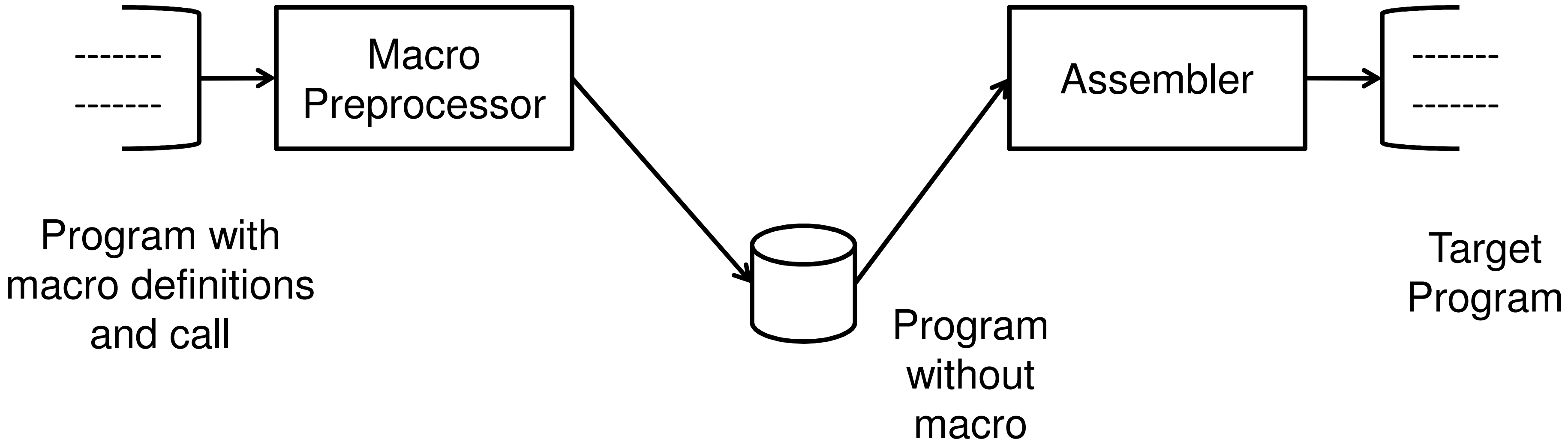


Figure: A Schematic of a macro preprocessor

Design of a Macro Preprocessor

- Design Overview

1. Identify macro calls in the program
2. Determine the values of formal parameter.
3. Maintain the values of expansion time variables declared in a macro.
4. Organize expansion time control flow.
5. Determine the values of sequencing symbol.
6. Perform expansion of a model statement.

Design of a Macro Preprocessor

1. Identify macro calls in the program
   - Macro Name Table (MNT) is designed to hold the names of all macro definition in a program.
   - A macro name is entered in this table when a macro definition is processed.
   - While processing a statement in the source program, the preprocessor compares the string found in its mnemonic field with the macro names in MNT.

Design of a Macro Preprocessor

2. Determine the values of formal parameter.

- Actual Parameter Table (APT) is designed to hold the values of formal parameters during expansion of macro call.
- Each entry in the table is a pair

  (<formal parameter name>, <value>)
- Parameter Default Table (PDT) is designed to hold the default values of keyword parameters during expansion of macro call.
- Each entry in the PDT is a pair

  (<formal parameter name>, <default value>)
- If a macro call statement does not specify a value for some parameter then its default value would be copied from PDT to APT.

Design of a Macro Preprocessor

3. Maintain the values of expansion time variables declared in a macro.
    - Expansion Time Variable’s Table(EVT)
    - Each entry in the table is a pair

      (<EV name>, <value>)
    - The value field of a pair is accessed when a preprocessor statement or model statement under expansion refers to an EV

Design of a Macro Preprocessor

4. Organize expansion time control flow.
   - Macro definition Table(MDT) is used to store the body of macro.
   - The flow of control during macro expansion determines when a model statement is to be visited for expansion.

Design of a Macro Preprocessor

5. Determine the values of sequencing symbol.
   - A sequencing Symbol Table(SST)
   - Each entry in the table is a pair

     (<sequencing symbol name>, <MDT entry #>)

Design of a Macro Preprocessor

6. Perform expansion of a model statement

- This is the trivial task given below:
  - MEC points to the MDT entry containing the model statement.
  - Values of formal parameter and EV's are available in APT and EVT respectively.
  - The model statement defining a sequencing symbol can be identified from SST.

Design of a Macro Preprocessor

Data Structures

– Macro expansion can be made more efficient by storing an intermediate code for a statement rather than it’s source form in MDT table.

e.g. MOVER AREG, ABC

Let the pair (ABC, ALPHA) occupy entry#5 in APT. The search in APT can be avoided if the model statement appears as

MOVER AREG, (P, 5)

in MDT where (P, 5) stands for the words ‘parameter#5’

Design of a Macro Preprocessor

Data Structures

- The information (<*formal parameter name*>,<*value*>) in APT has been split into two tables:
  - PNTAB – formal parameters name
  - APTAB – formal parameter values
- Similarly EVT into ENNTAB and EVTAB
- SST into SSNTAN and SSTAB
- PDT is replaced a Keyword Parameter Default Table (KPDTAB)

Design of a Macro Preprocessor

Data Structures

- Macro Name Table (MNT)
- Parameter Name Table (PNTAB)
- EV Name Table (EVNTAB)
- SS Name Table (SSNTAB)
- Keyword Parameter Default Table (KPDTAB)
- Macro Definition Table (MDT)
- Actual Parameter Table (APTAB)
- EV Table (EVTAB)
- SS Table (SSTAB)

Design of a Macro Preprocessor

Data Structures

- Macro Name Table (MNT) fields
  - 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)

Design of a Macro Preprocessor

Data Structures

- Parameter Name Table (PNTAB) Fields
  - Parameter Name
- EV Name Table (EVNTAB) Fields
  - EV Name
- SS Name Table (SSNTAB)
  - SS name

Design of a Macro Preprocessor

Data Structures

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

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

Design of a Macro Preprocessor

Data Structures

- Actual Parameter Table (APTAB) Fields
  - Value
- EV Table (EVTAB) Fields
  - Value
- SS Table (SSTAB) Fields
  - MDT entry#

Design of a Macro Preprocessor

Example

```
          MACRO
          CLEARMEM    &X, &N, &REG=AREG
          LCL         &M
&M        SET         0
          MOVER       &REG, =‘0’
.MORE     MOVEM       &REG, &X + &M
&M        SET         &M+1
          AIF         (&M NE N) .MORE
          MEND
```

Macro call : CLEARMEM AREA, 10

Design of a Macro Preprocessor

- **PNTAB**

PNTAB_ptr	Parameter Name
1	X
2	N
3	REG

- **EVNTAB**

EV Name
M

- **SSNTAB**

SSNTAB_ptr	SS Name
1	MORE

Design of a Macro Preprocessor

- **MNT**

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

- **KPDTAB**

KPDTAB_ptr	Parameter Name	Default value
.		
10	REG	AREG

- **SSTAB**

SSTAB_ptr	MDT Entry #
.	
5	28

Design of a Macro Preprocessor

- **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	
	.	.	.

Design of a Macro Preprocessor

- **APTAB**

APTAB_ptr	Value
1	AREA
2	10
3	AREG

- **EVTAB**

EVTAB_ptr	Value
1	0

Processing of Macro Definition

Algorithm – Processing of a Macro Definition

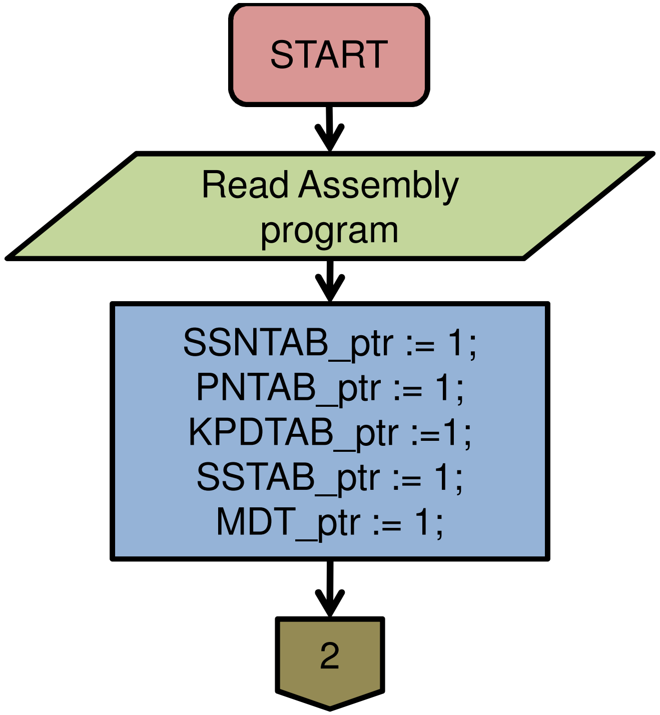
```
1.  SSNTAB_ptr := 1;
    PNTAB_ptr := 1;
    KPDTAB_ptr :=1;
    SSTAB_ptr := 1;
    MDT_ptr := 1;
2.  Process the macro prototype statement and form the MNT
    entry
     (a) name := macro name;
     (b) for each positional parameter
         (i) Enter parameter name in PNTAB[PNTAB_ptr].
         (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
          (ii) For a parameter, generate the specification (P, #n).
          (iii) For an expansion variable, generate the specification (E, #m).
          (iv) Record the IC in MDT[MDT_ptr];
          (v) 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;
```

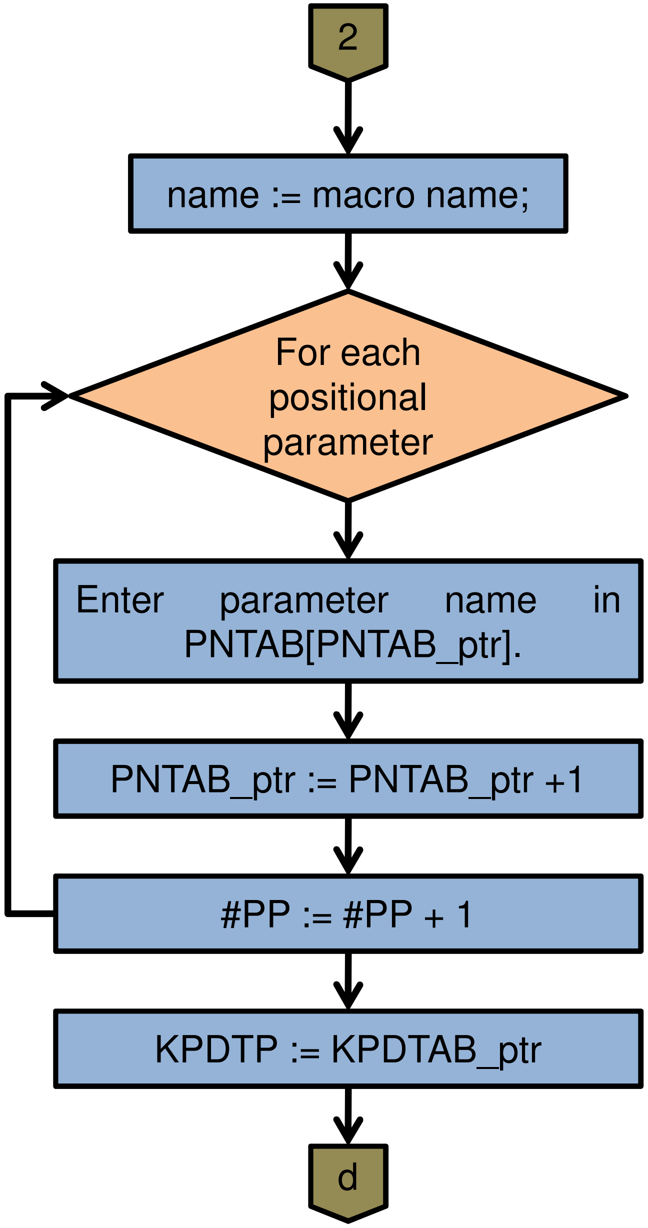


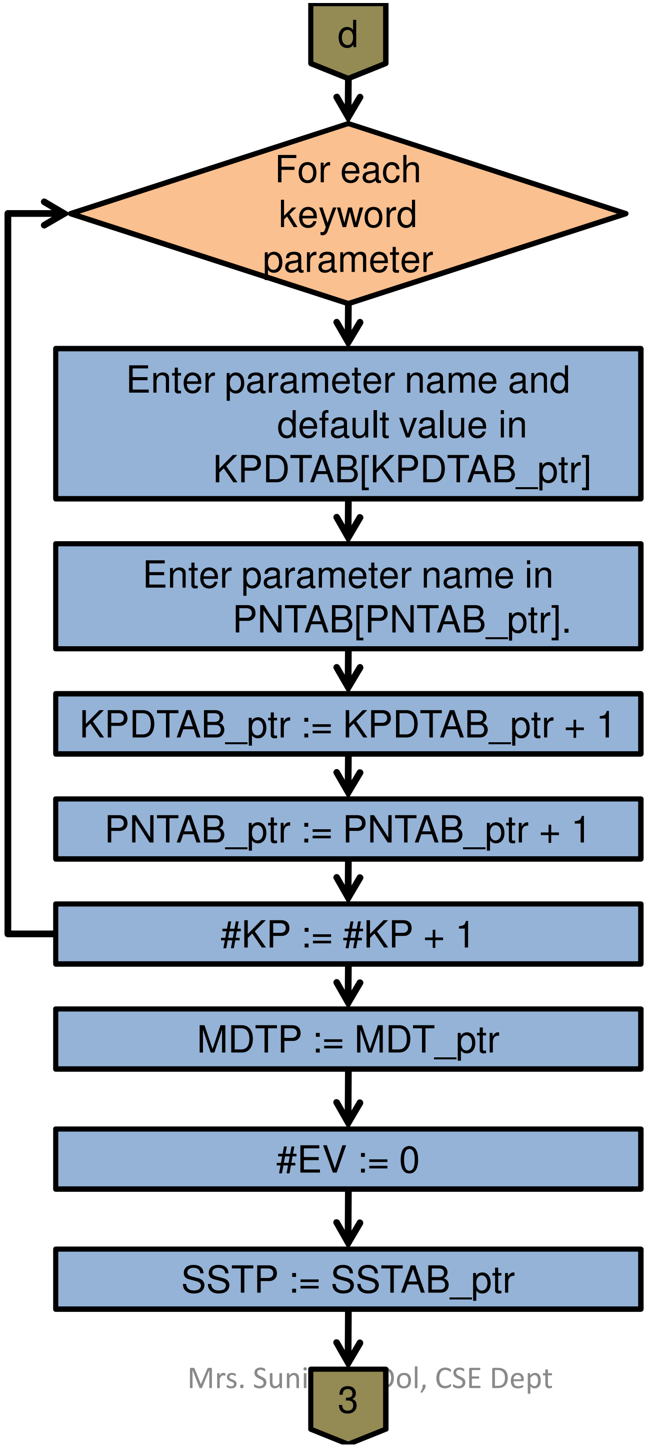
START

Read Assembly program

SSNTAB_ptr := 1;
PNTAB_ptr := 1;
KPDTAB_ptr :=1;
SSTAB_ptr := 1;
MDT_ptr := 1;

2


2
name := macro name;
For each positional parameter
Enter parameter name in PNTAB[PNTAB_ptr].
PNTAB_ptr := PNTAB_ptr +1
#PP := #PP + 1
KPDTP := KPDTAB_ptr
d



d

For each keyword parameter

Enter parameter name and default value in KPDTAB[KPDTAB_ptr]

Enter parameter name in PNTAB[PNTAB_ptr].

KPDTAB_ptr := KPDTAB_ptr + 1

PNTAB_ptr := PNTAB_ptr + 1

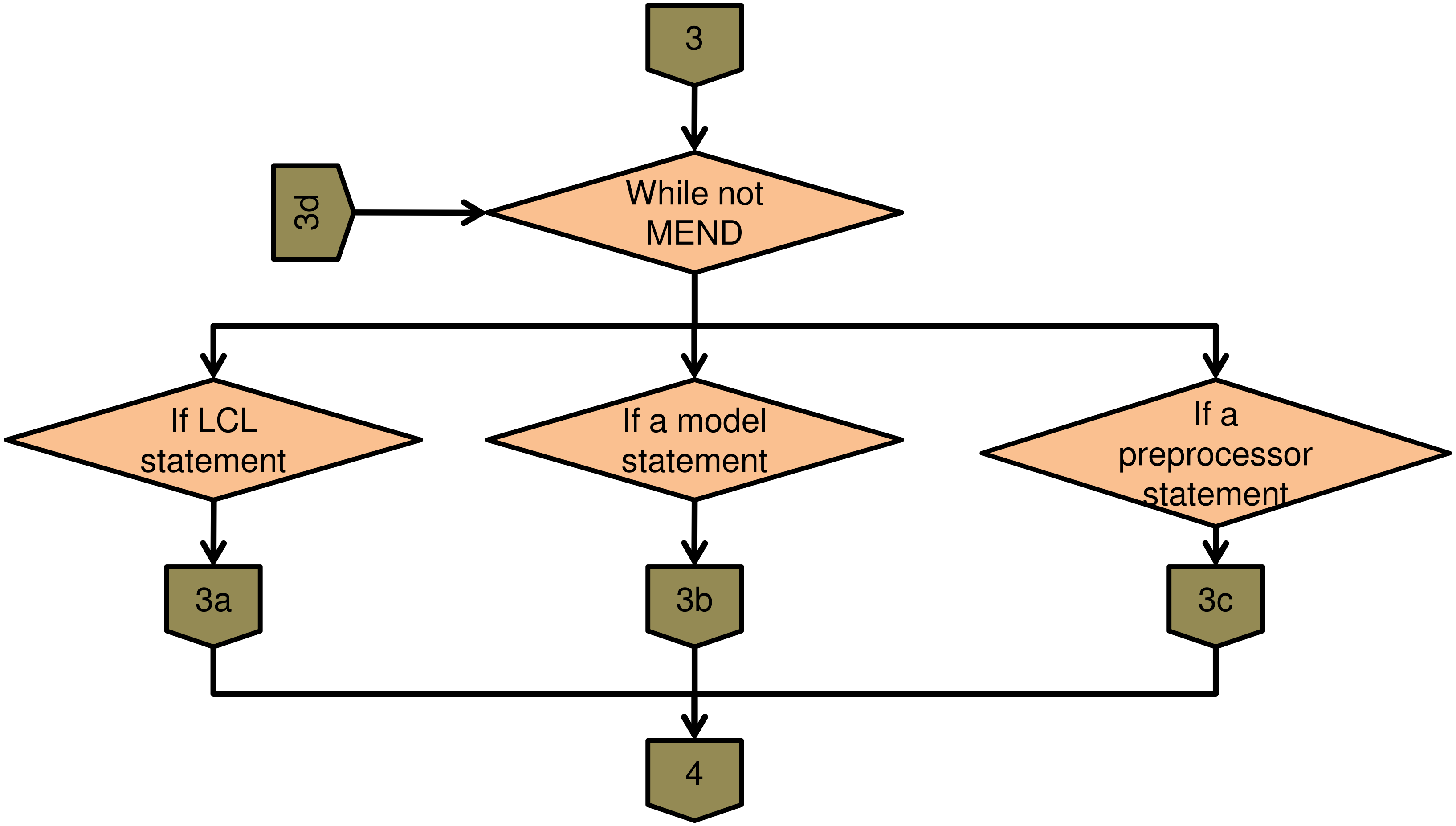
#KP := #KP + 1

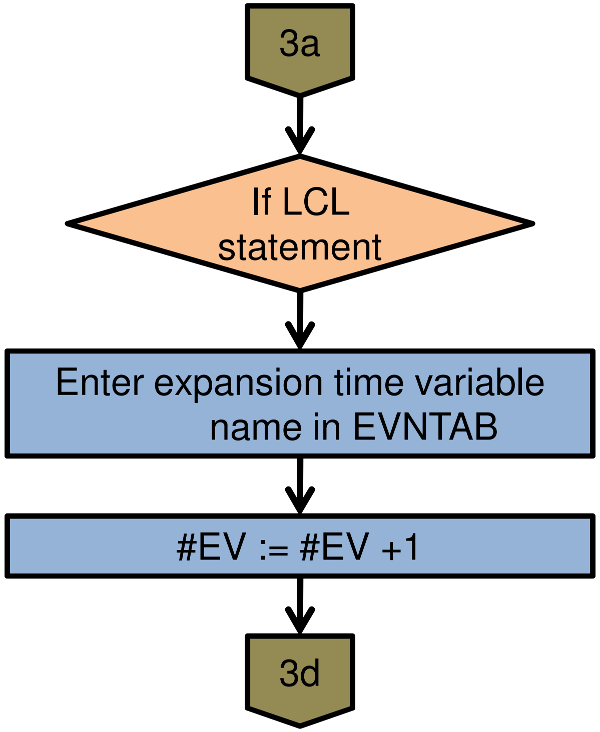
MDTP := MDT_ptr

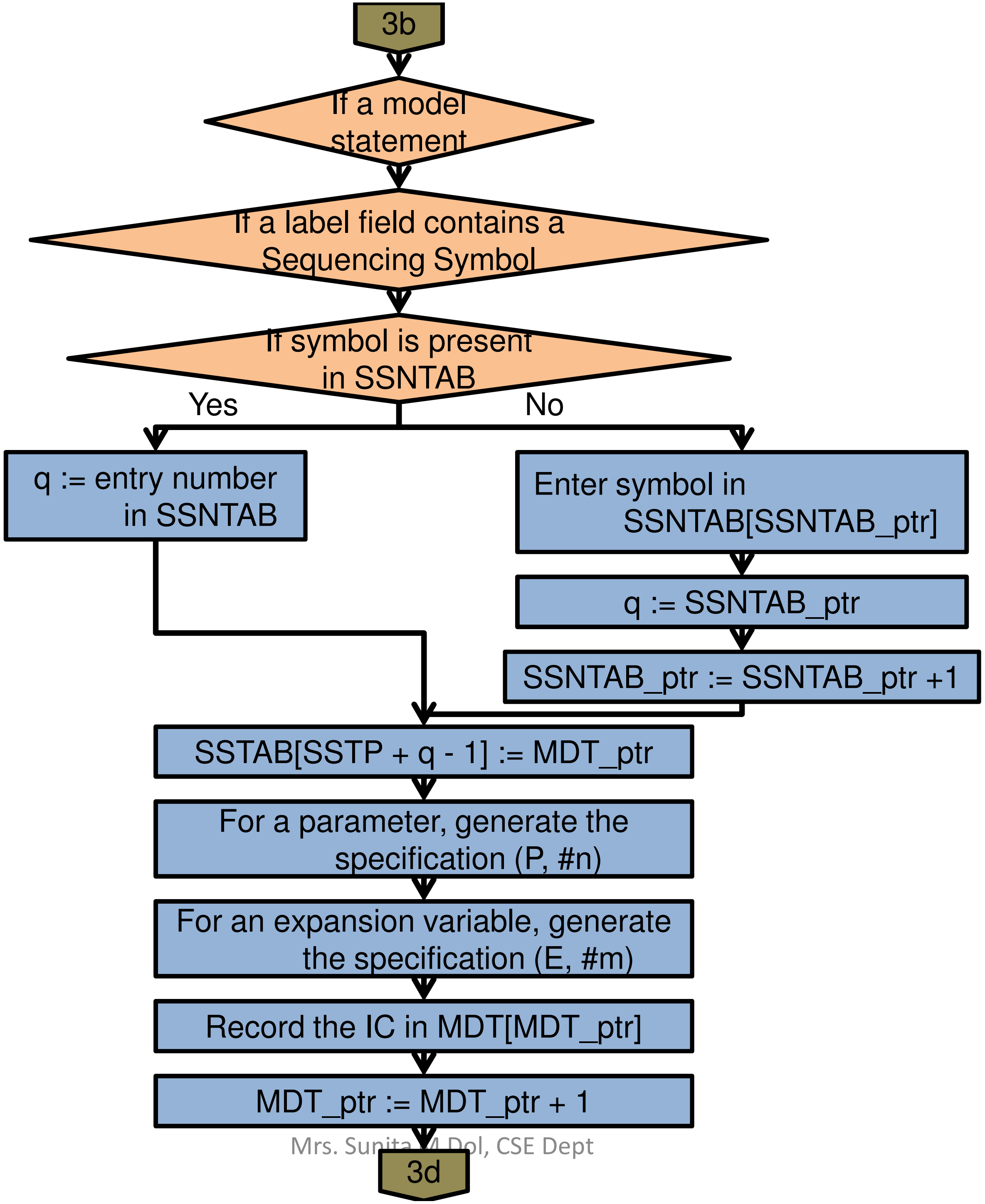
#EV := 0

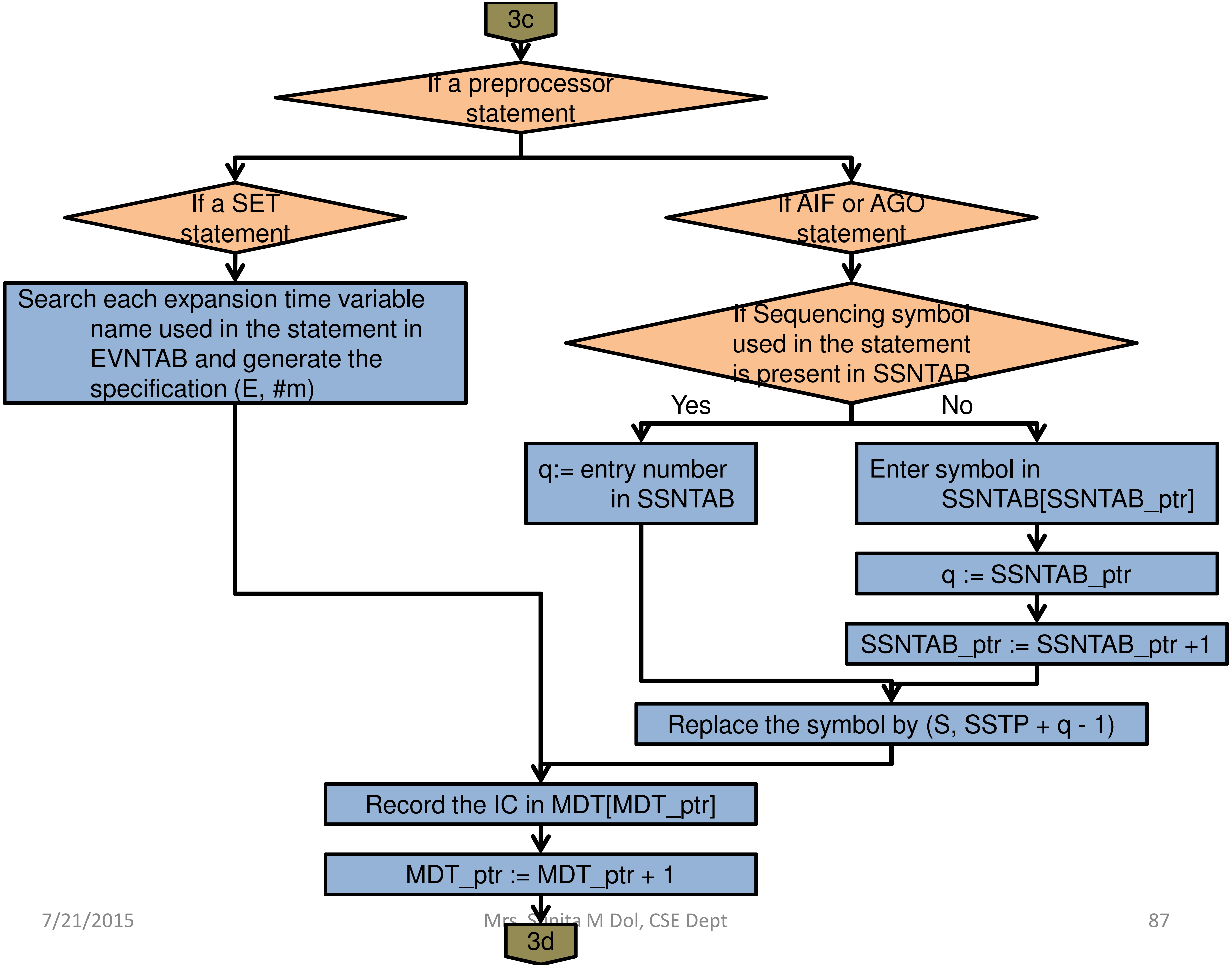
SSTP := SSTAB_ptr

3


3
3d
While not MEND
If LCL statement
If a model statement
If a preprocessor statement
3a
3b
3c
4


3a
If LCL statement
Enter expansion time variable name in EVNTAB
#EV := #EV +1
3d


3b
If a model statement
If a label field contains a Sequencing Symbol
If symbol is present in SSNTAB
Yes
No
q := entry number in SSNTAB
Enter symbol in SSNTAB[SSNTAB_ptr]
q := SSNTAB_ptr
SSNTAB_ptr := SSNTAB_ptr +1
SSTAB[SSTP + q - 1] := MDT_ptr
For a parameter, generate the specification (P, #n)
For an expansion variable, generate the specification (E, #m)
Record the IC in MDT[MDT_ptr]
MDT_ptr := MDT_ptr + 1
3d



3c

If a preprocessor statement

If a SET statement

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

If AIF or AGO statement

If Sequencing symbol used in the statement is present in SSNTAB

Yes

No

q:= entry number in SSNTAB

Enter symbol in SSNTAB[SSNTAB_ptr]

q := SSNTAB_ptr

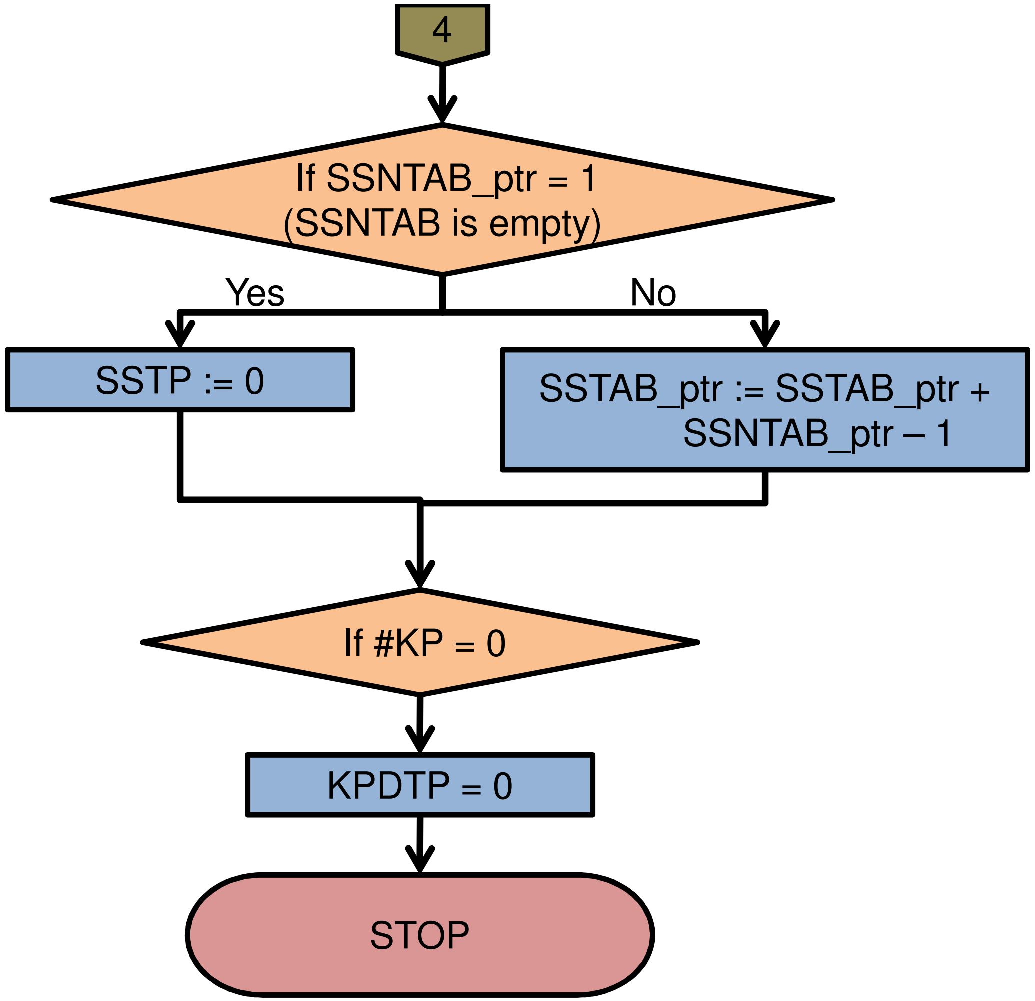
SSNTAB_ptr := SSNTAB_ptr +1

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

Record the IC in MDT[MDT_ptr]

MDT_ptr := MDT_ptr + 1

3d


4
If SSNTAB_ptr = 1
(SSNTAB is empty)
Yes
No
SSTP := 0
SSTAB_ptr := SSTAB_ptr + SSNTAB_ptr – 1
If #KP = 0
KPDTP = 0
STOP

Macro Expansion

- We use the following data structure for macro expansion
  - APTAB Actual Parameter Table

    (The no of entries in APTAB #e_{APTAB} = #PP + #KP)
  - EVTAB EV Table

    (#e_{EVTAB} = the value in #EV field of MNT)
  - MEC Macro Expansion Counter
  - APTAB_ptr APTAB pointer
  - EVTAB_ptr EVTAB pointer

Macro Expansion

Algorithm –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]

Macro Expansion

(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]

Macro Expansion

2. While statement pointed by MEC is not MEND statement
   - (a) If a model statement then
     - (i) Replace the operands of the form (P, #n) and (E, #m) by values in APTAB[n] and EVTAB[m] respectively.
     - (ii) Output the generated statement.
     - (iii) MEC := MEC + 1;
   - (b) If a SET statement with the specification (E, #m) in the label field then
     - (i) Evaluate the expression in the operand field and set an appropriate value in EVTAB[m].
     - (ii) MEC := MEC + 1;

Macro Expansion

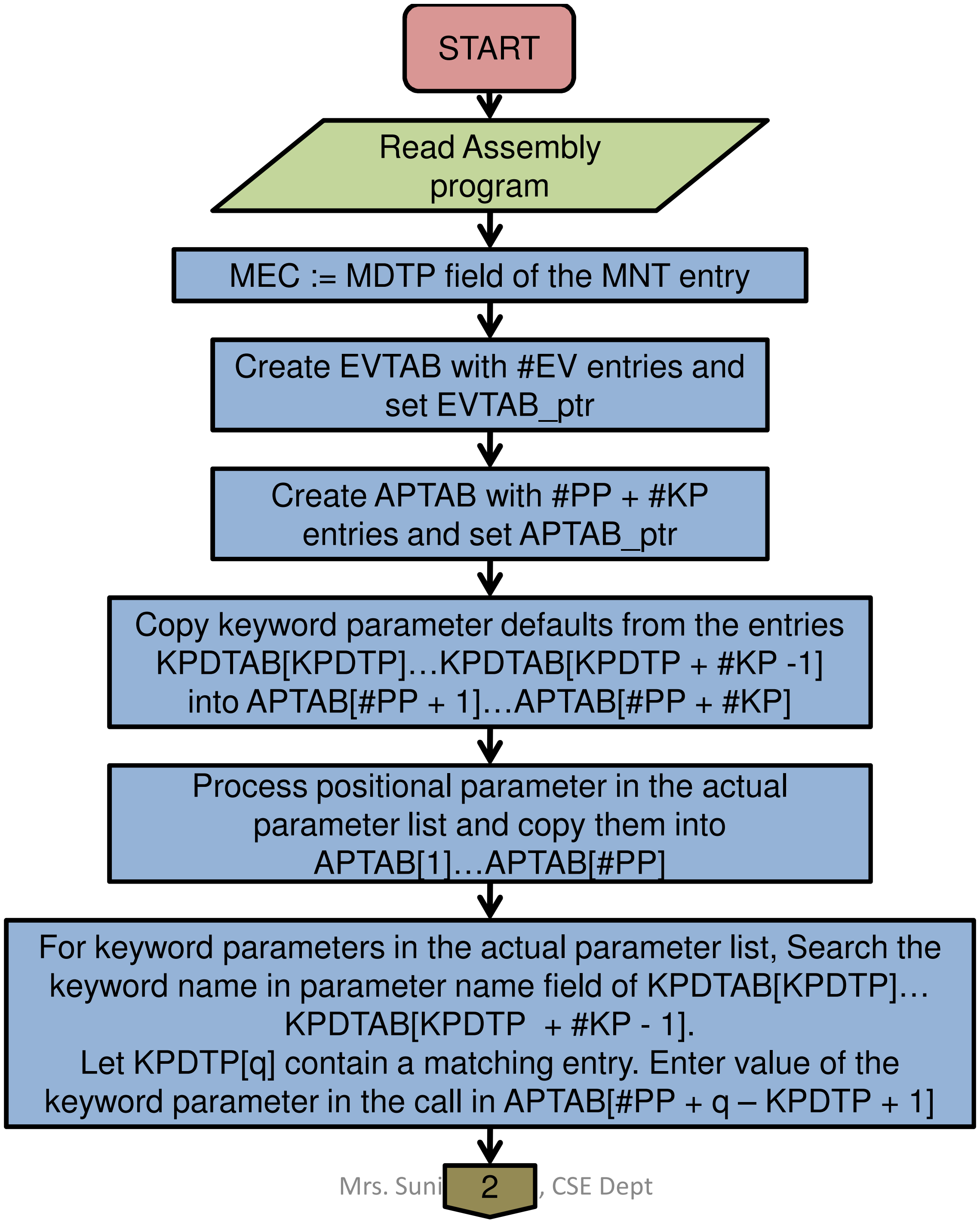
(c)If an AGO statement with (S, #s) in operand field then

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

(d)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];



START

Read Assembly program

MEC := MDTP field of the MNT entry

Create EVTAB with #EV entries and set EVTAB_ptr

Create APTAB with #PP + #KP entries and set APTAB_ptr

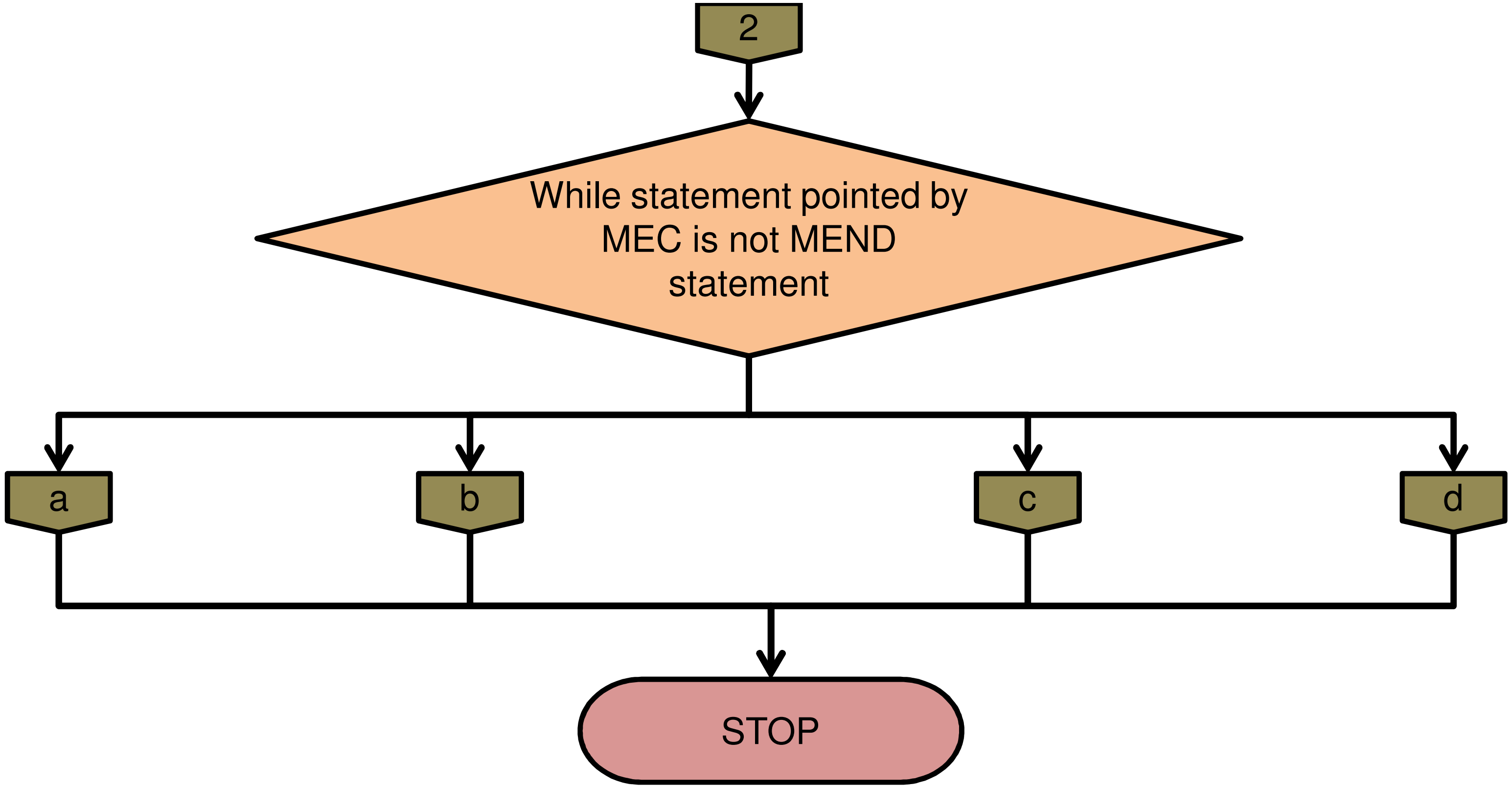
Copy keyword parameter defaults from the entries KPDTAB[KPDTP]…KPDTAB[KPDTP + #KP -1] into APTAB[#PP + 1]…APTAB[#PP + #KP]

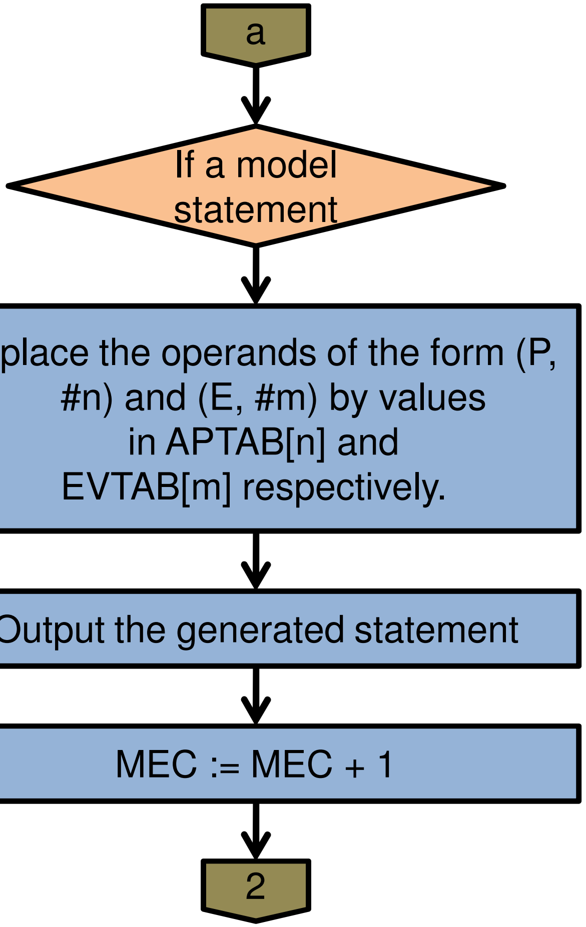
Process positional parameter in the actual parameter list and copy them into APTAB[1]…APTAB[#PP]

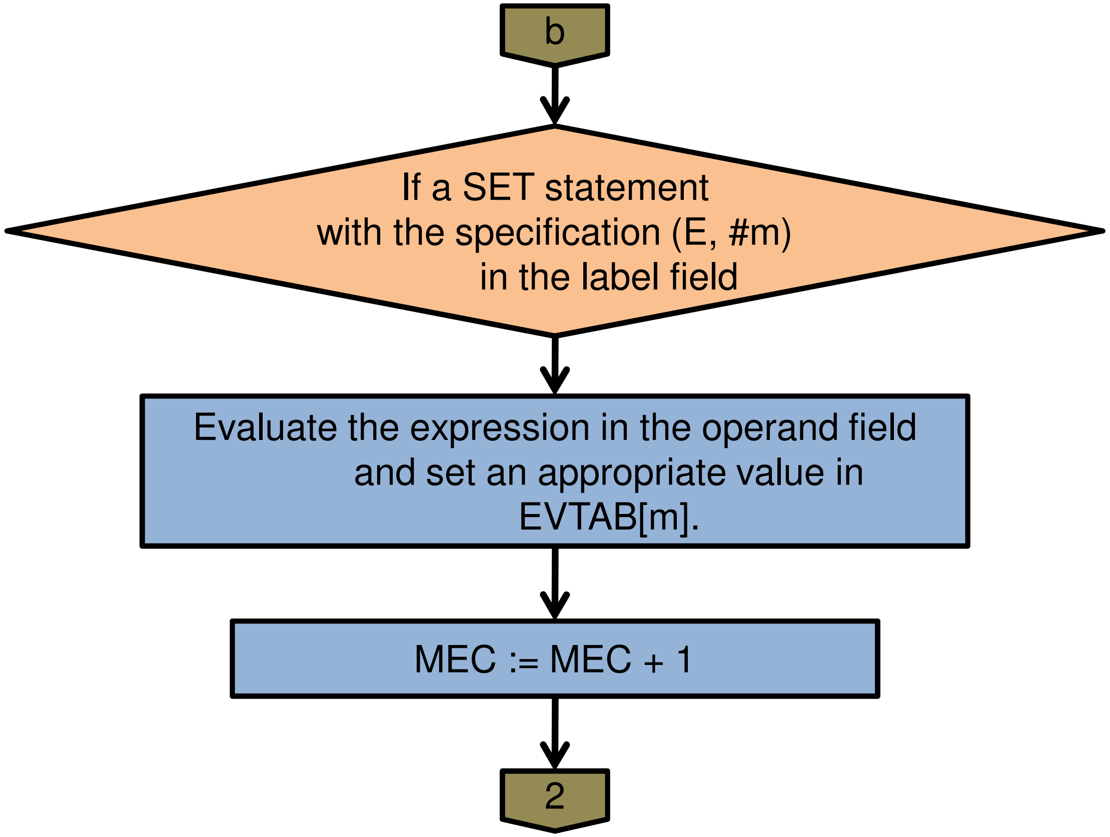
For keyword parameters in the actual parameter list, Search the keyword name in parameter name field of KPDTAB[KPDTP]… KPDTAB[KPDTP + #KP - 1].

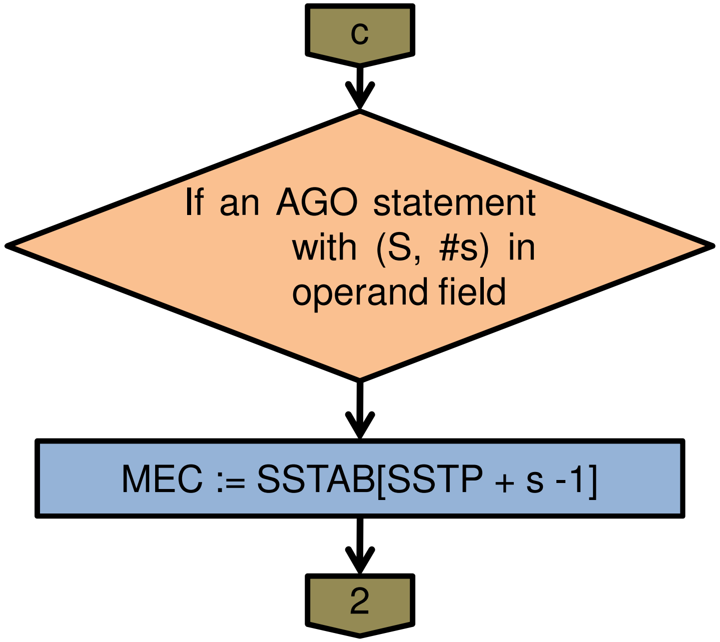
Let KPDTP[q] contain a matching entry. Enter value of the keyword parameter in the call in APTAB[#PP + q – KPDTP + 1]

2


2
While statement pointed by MEC is not MEND statement
a
b
c
d
STOP


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

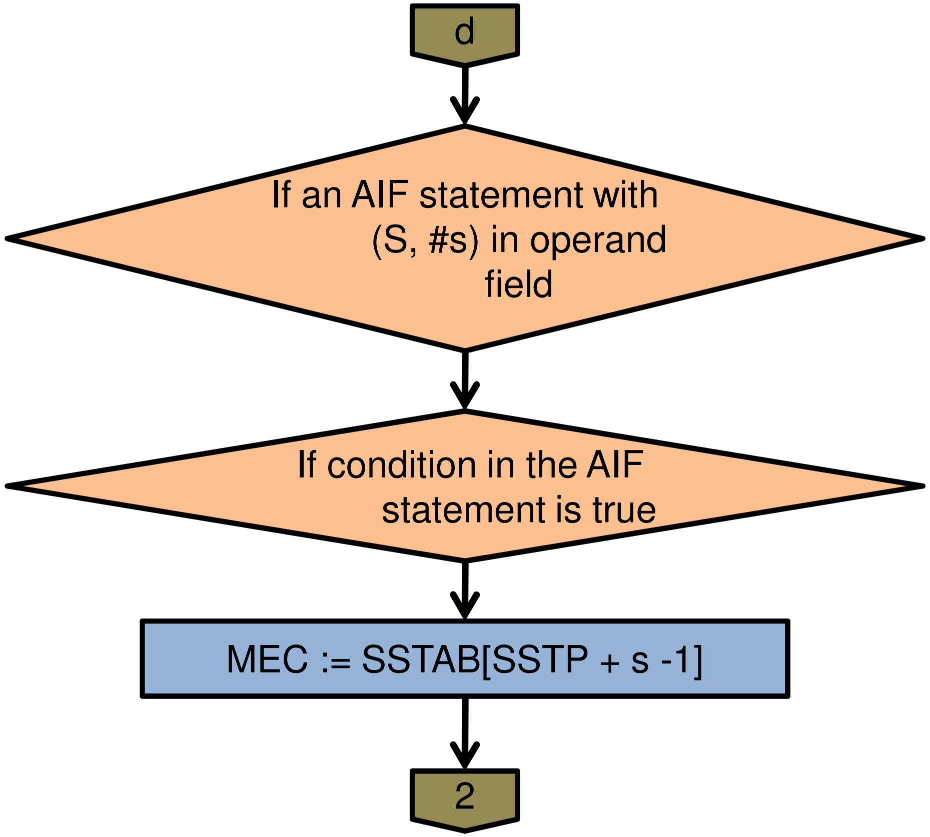

b
If a SET statement with the specification (E, #m) in the label field
Evaluate the expression in the operand field and set an appropriate value in EVTAB[m].
MEC := MEC + 1
2


c

If an AGO statement with (S, #s) in operand field

MEC := SSTAB[SSTP + s -1]

2


d
If an AIF statement with (S, #s) in operand field
If condition in the AIF statement is true
MEC := SSTAB[SSTP + s -1]
2