System Programming-Macro

Macro and Macro Processors

- Macro is a unit of specification for program for generation through expansion.
- Macro is a single line abbreviation for groups of instruction

  - Macro Definition
  - Macro call

Macro and Macro Processors

- Macro definition is enclosed between a macro header and macro end statement.
- A macro instruction (Macro) is a notational convenience for the programmer
  - Allows the programmer to write short hand programs (modular programming).
- The macro processor replaces each macro instruction with its equivalent block of instructions.

BASIC MACRO PROCESSOR FUNCTIONS

- Directives used during usage of Macro:
  - Macro: Indicates begin of Macro
  - MEND: indicates end of Macro
- Prototype for Macro:
  - Each argument starts with Name and macro
  - Parameter list
    - .
    - .
    - MEND

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
```

Macro Expansion

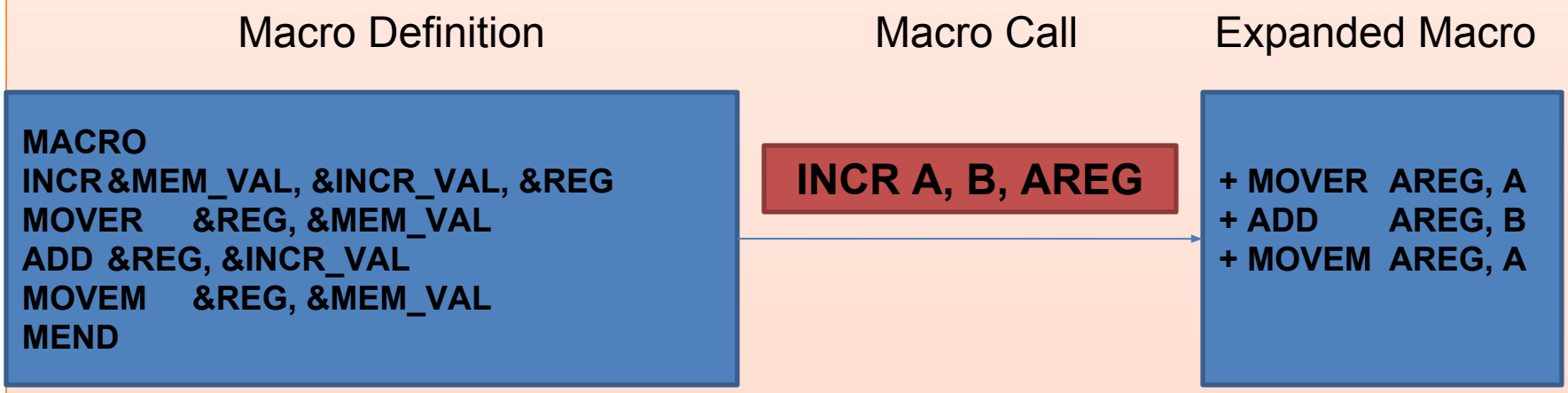
```
macro read
mov ah,1
int 21h
endm

macro print m,m1
mov ah,9
lea dx, m
int 21h
lea dx, m1
```

```
.code
read
print  msg, msg1
add ah, al
end
```

Macro Expansion

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

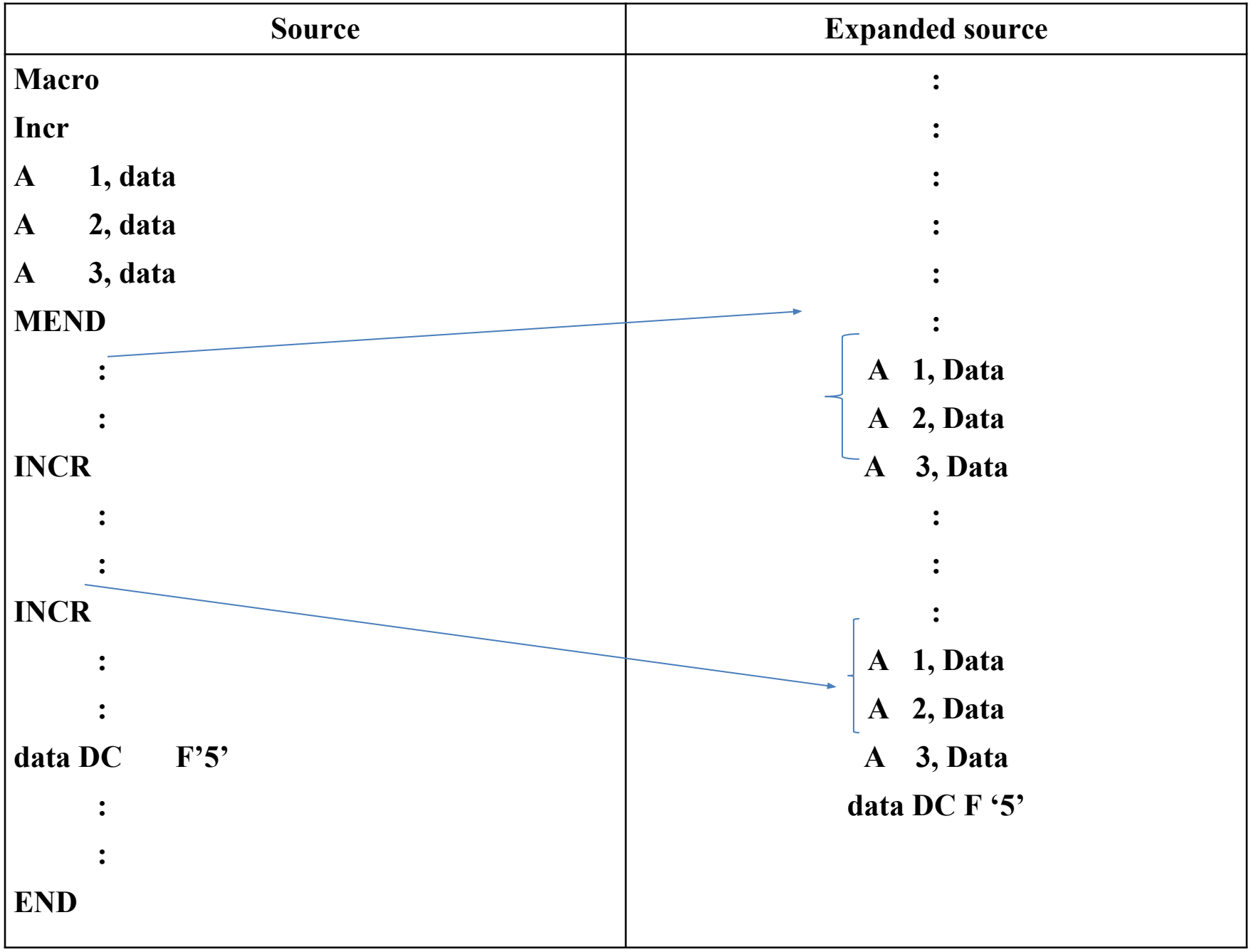


Macro Expansion

- Two key notions concerning macro expansion are:
  - Expansion Time Control Flow(Semantic Expansion) determines the order in which model statements are visited during macro expansion.
    - Conditional expansion
    - Expansion time loop
  - 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


Source
Expanded source
Macro
Incr
A 1, data
A 2, data
A 3, data
MEND
:
:
INCR
:
:
INCR
:
:
data DC F'5'
:
:
END
:
:
:
:
:
:
A 1, Data
A 2, Data
A 3, Data
:
:
:
A 1, Data
A 2, Data
A 3, Data
data DC F '5'

Macro Expansion

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

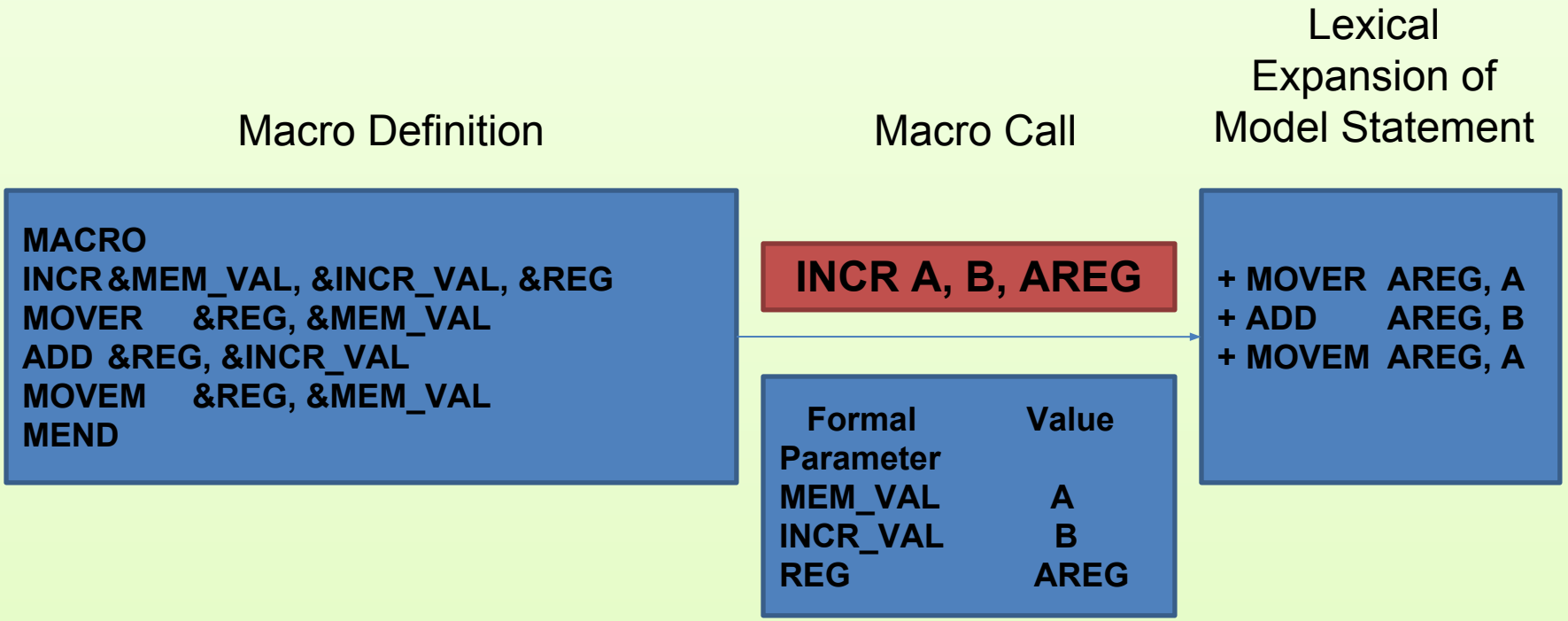
1. Positional Parameter
2. Keyword Parameter
3. Default Specification of Parameter
4. Macros with Mixed Parameter
5. Other uses of Parameter

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 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 Definition

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

Macro Call

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

Macro Classification

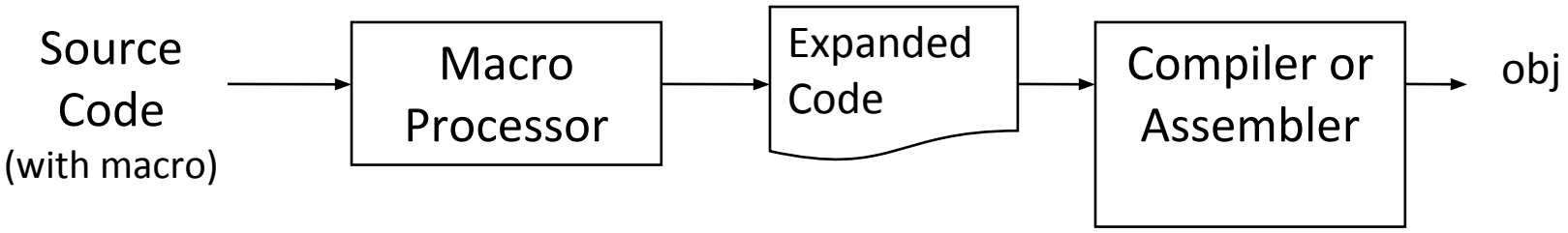
Macro Processors

Program with macro

Macro Preprocessor

Program without macro

Assembler



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      &FIRST, &SECOND, REG=BREG
MOVER     BREG, TMP
MEND
```

Outer Macro

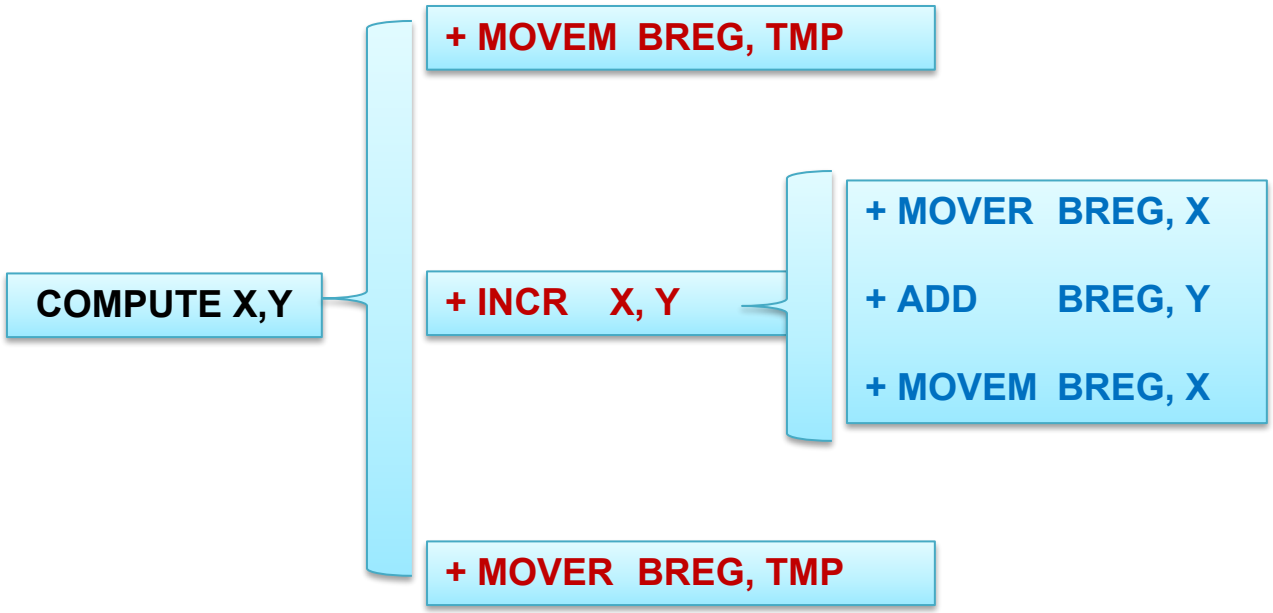
Nested Macro Calls

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

Inner Macro

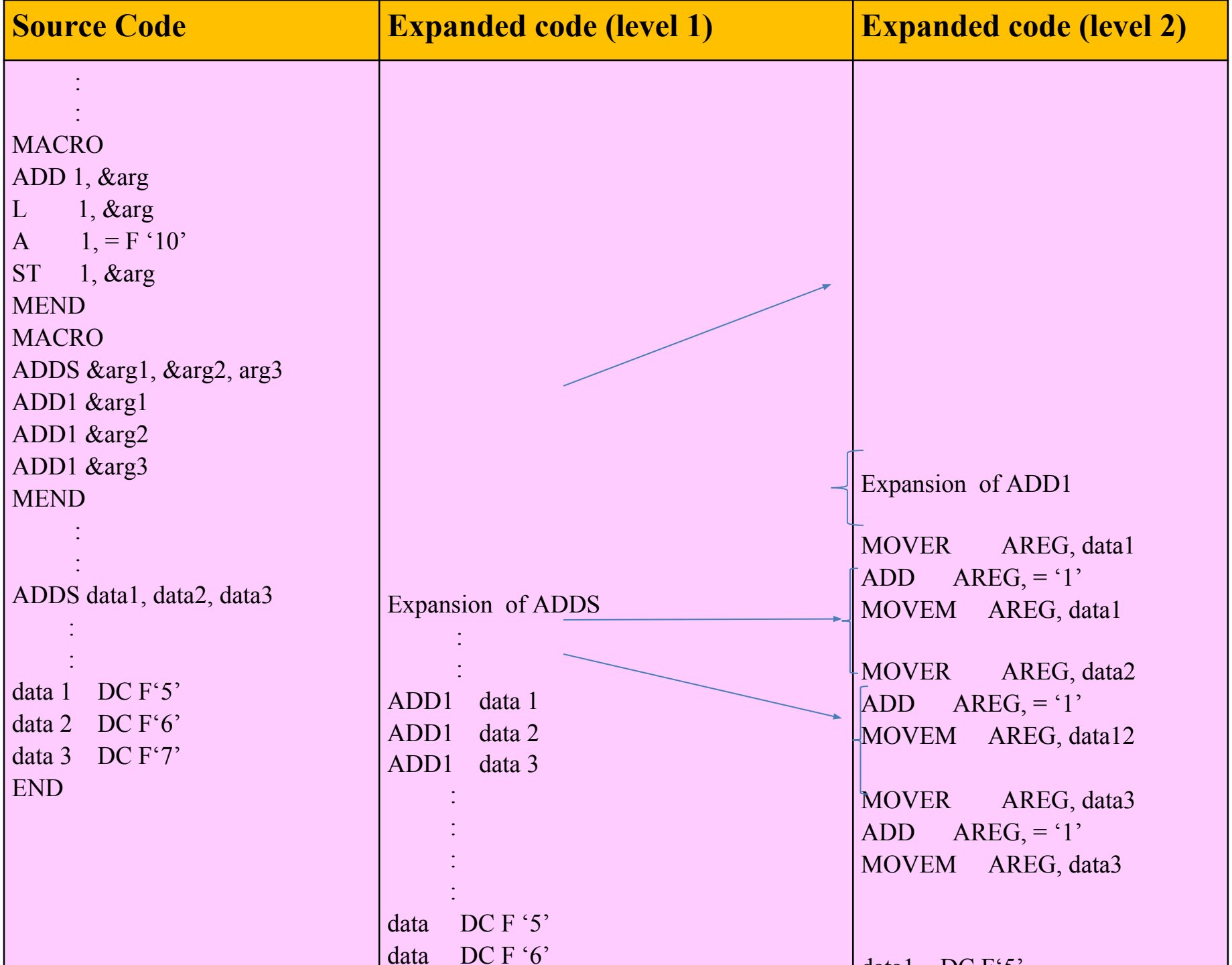
```
MACRO
COMPUTE        &FIRST, &SECOND
MOVEM          BREG, TMP
INCR           &FIRST, &SECOND, REG=BREG
MOVER          BREG, TMP
MEND
```

Outer Macro



Source Code	Expanded code (level 1)	Expanded code (level 2)

:
:
MACRO
ADD1 &arg
MOVER AREG, &arg
ADD AREG, = '1'
MOVEM AREG, &arg
MEND
MACRO
ADDS &arg1, &arg2, arg3
ADD1 &arg1
ADD1 &arg2
ADD1 &arg3
MEND
:
:
ADDS data1, data2, data3
:
:
data 1 DC '5'
data 2 DC '6'
data 3 DC '7'
END
	Expansion of ADDS
	:
	:
	ADD1 data 1
	ADD1 data 2
	ADD 1 data 3
	:
	:
	:
	:
	data DC '5'
	data DC '6'
	data DC '7'
	END	


Source Code
Expanded code (level 1)
Expanded code (level 2)
:
:
MACRO
ADD 1, &arg
L 1, &arg
A 1, = F '10'
ST 1, &arg
MEND
MACRO
ADDS &arg1, &arg2, arg3
ADD1 &arg1
ADD1 &arg2
ADD1 &arg3
MEND
:
:
ADDS data1, data2, data3
:
:
data 1 DC F'5'
data 2 DC F'6'
data 3 DC F'7'
END
Expansion of ADDS
:
:
ADD1 data 1
ADD1 data 2
ADD1 data 3
:
:
:
:
data DC F '5'
data DC F '6'
Expansion of ADD1
MOVER AREG, data1
ADD AREG, = '1'
MOVEM AREG, data1
MOVER AREG, data2
ADD AREG, = '1'
MOVEM AREG, data12
MOVER AREG, data3
ADD AREG, = '1'
MOVEM AREG, data3

Conditional macro expansion

- This allows conditional selection of machine instructions that appear in expansion of macro call.
- The macro processor pseudo op-codes AIF and AGO help to do conditional macro expansion.
- AIF is conditional branch pseudo op, it performs arithmetic test and branch only if condition is true.
- AGO is unconditional pseudo op or it is like go to statement.

Source Program

```
:
MACRO
& arg0  INCR & count, & arg1, & arg 2, & arg3
& arg0  A       AREG, &arg1
          AIF     (& Count EQ.1). FINAL
          A       BREG,& arg2
          AIF     (&count EQ 2). FINAL
          A       CREG, &arg3
.FINAL   MEND
          :
LOOP1 INCR   3, 1, 2, 3
          :
          :
LOOP2  INCR  2, 3, 2,1
          :
LOOP3  INCR  1, 1,2,3
          :
data1      DC  ‘5’
data2      DC  ‘6’
data3      DC  ‘7’
```

Expanded Program

```
LOOP1  A    AREG,1
          A    BREG, 2
          A    CREG, 3
          :
          :
LOOP2  A  AREG, 3
          A  BREG, 2
          :
          :
LOOP3 A   AREG, 1
          :
       data1 DC ‘5’
       data2 DC ‘6’
      data3 DC ‘7’
          :
```

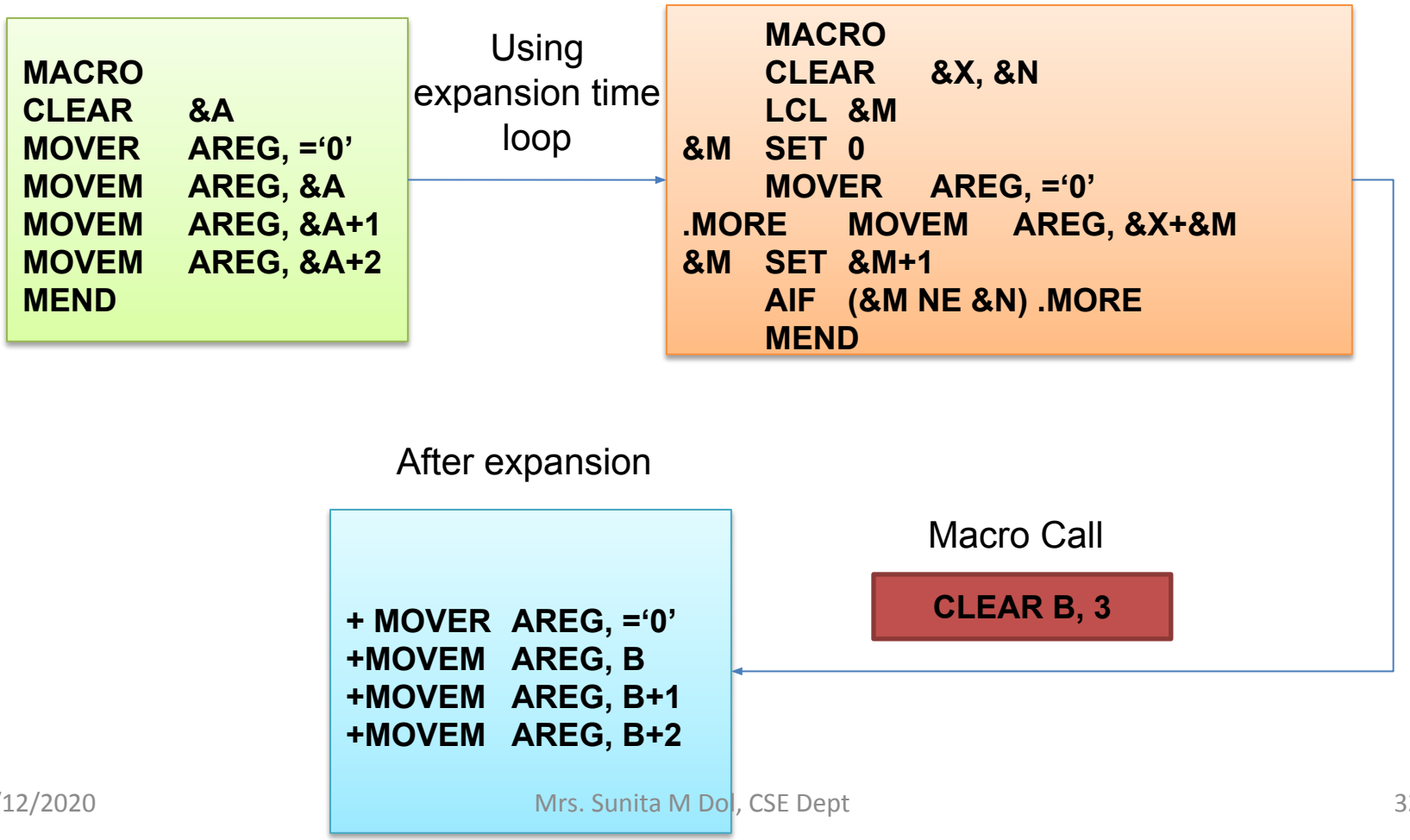
Advanced Macro Facilities

- 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 Example



Data structures

- MNT – Macro Name Table
- MDT – Macro Definition Table
- MNTC – MNT counter
- MDTC – MDT counter
- ALA – Argument List Array

MNT (Macro name table)

Index	Name	MDT index
1	read	1
2	print	4

MDT (Macro Definitions Table)

Index	Card
1	mov ah,1
2	int 21h
3	mend
4	mov ah, 9
5	lea dx, #0
6	int 21h
7	lea dx, #1
8	int 21h
9	mend

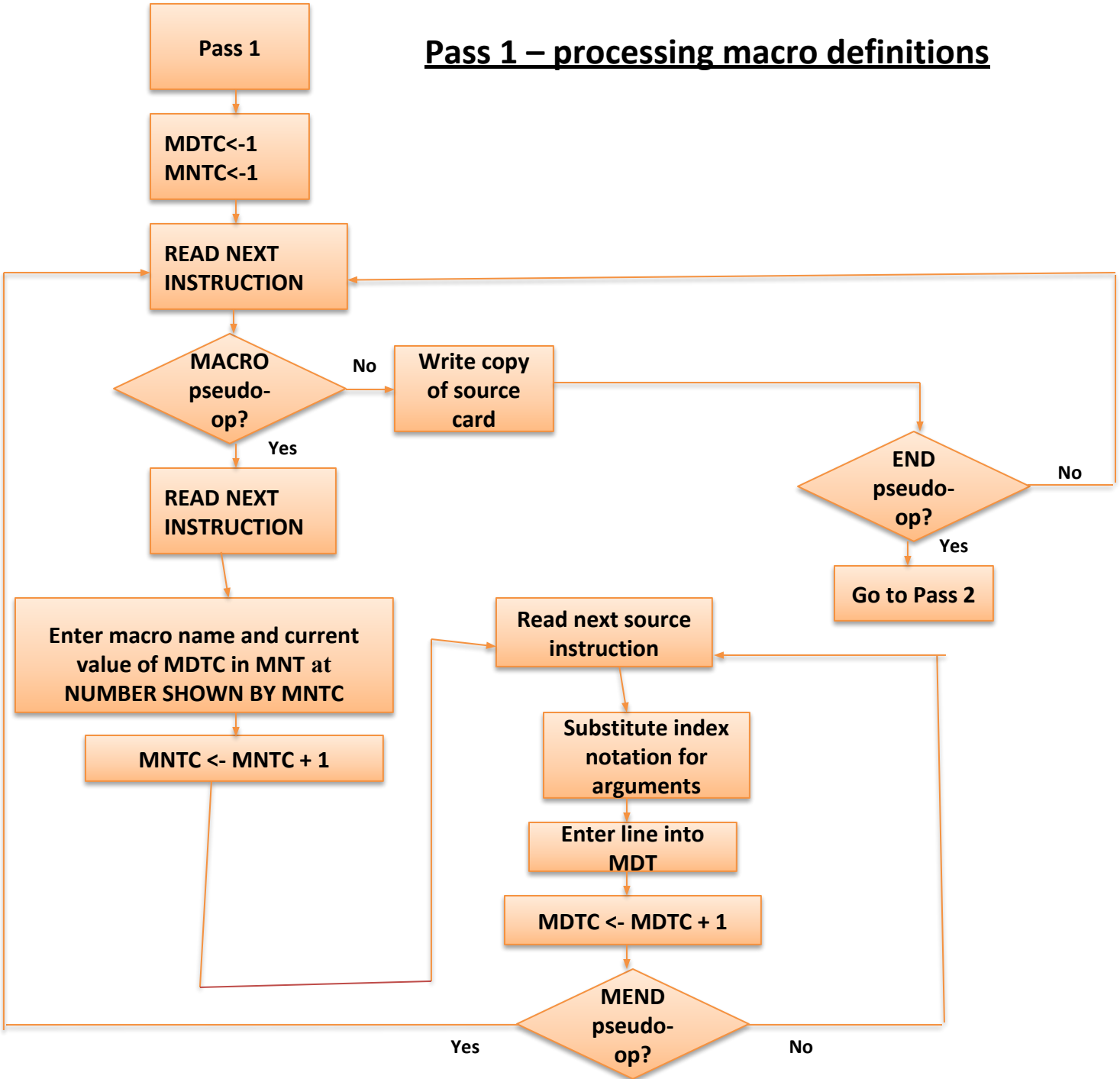
ALA(Argument List Array)

Index	Argument
0	msg
1	msg1

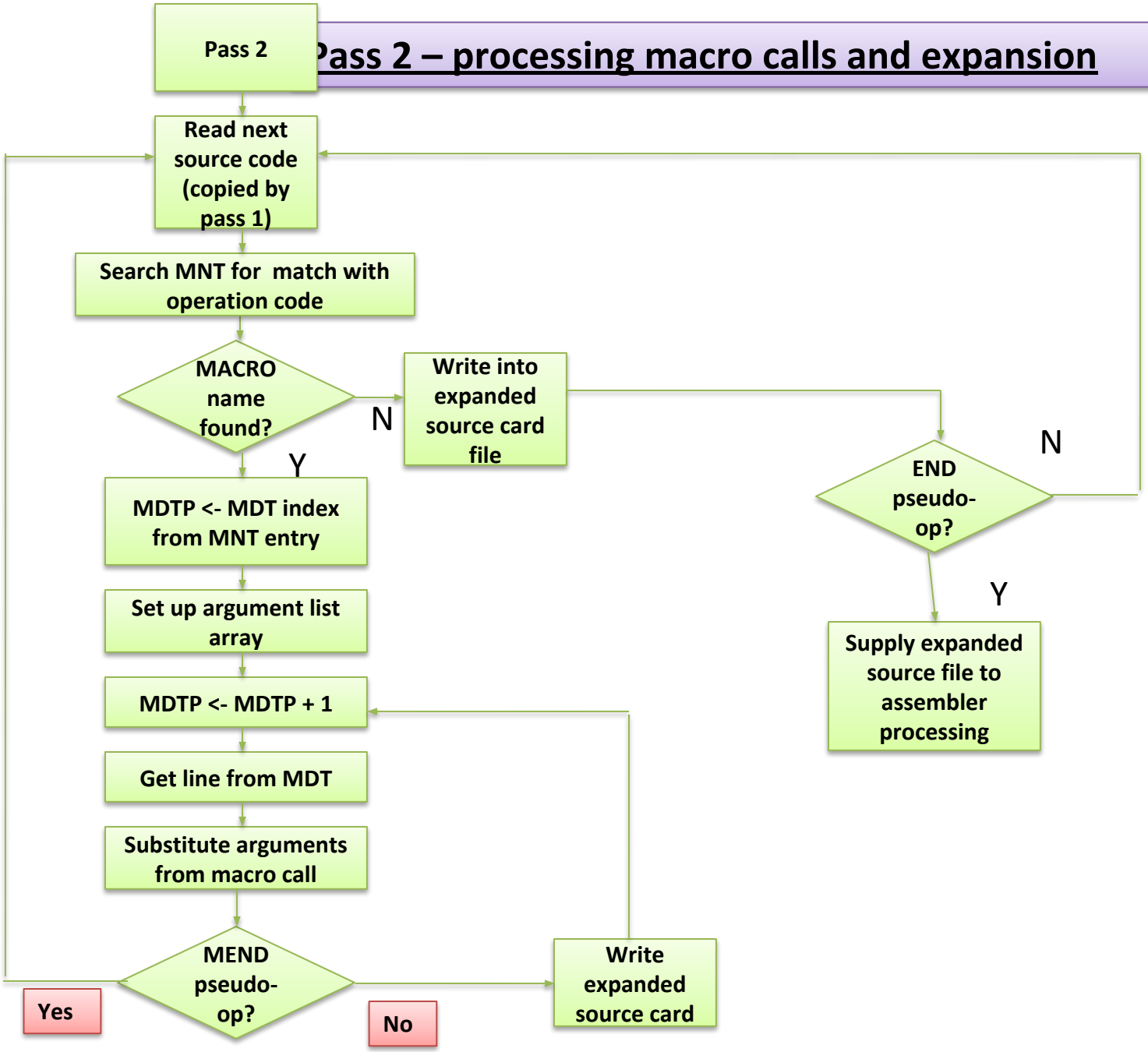
2- pass Microprocessor Design

1. Recognize Macro Definition
2. Save Definitions
3. Recognize Calls
4. Expand Calls and substitute arguments

Pass 1 – processing macro definitions



Pass 2 – processing macro calls and expansion

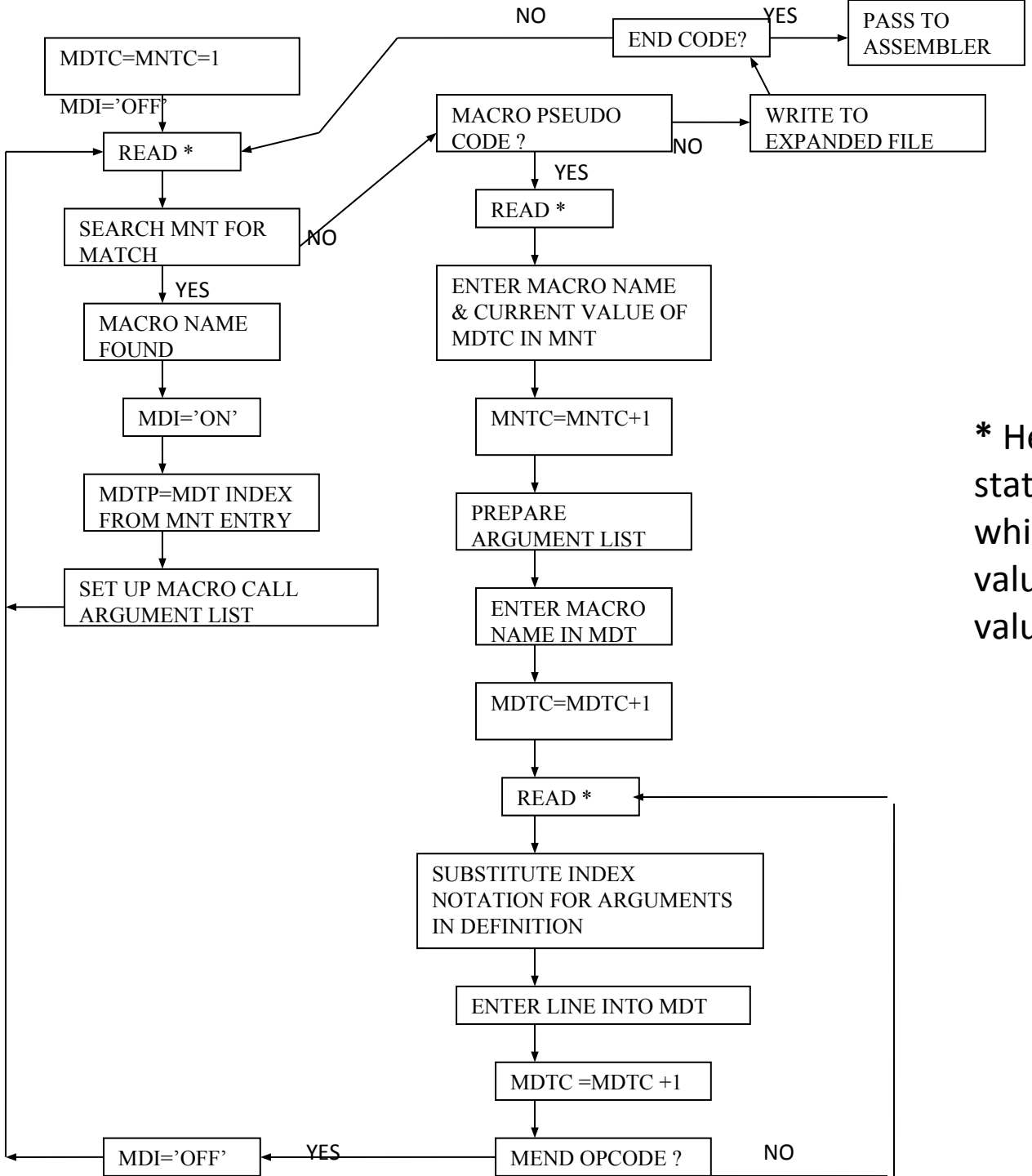


Comparison of Macro Processors Design

- Single pass
  - » every macro must be defined before it is called
  - » one-pass processor can alternate between macro definition and macro expansion
  - » Nested Macro calls are not
- Two pass algorithm
  - » Pass1: Recognize macro definitions
  - » Pass2: Recognize macro calls
  - » nested macro definitions are not allowed

Single Pass Macroprocessor

- (Restriction - all macros must be defined before calling them, we can join 2 passes into a single pass.)
- Here we need another indicator MDI which can have values ‘ON’ or ‘OFF’. MDI ON means lines are read from macro definition table and OFF means lines are read from regular input stream.



* Here read is not just a statement. It is a function which return different values depending on the value of MDI

Assembler &macro processor

- MACROPROCESSOR can be added to an assembler in two ways:

  1. As a preprocessor, which make complete pass over the text before pass 1 of assembler.

  2. With in pass 1 of assembler

(If we create macro processor within assembler some functions can be combined. Database can also be prepared jointly. e.g. MNT can be joined with assembler's MOT or POT. Similarly the read function that expand macro calls and receive the source I/P will be same.)


Thank you!