MIT-WPU
T.Y. B.Tech

System Software and Compilers

Course Objective & Course Outcomes

- **Course Objectives:**

1. To learn and understand different component of system software and fundamentals language processing activity.
2. To understand the process of converting assembly language program to machine language
3. To understand linking and loading concepts
4. Understand the basic concept of compiler design, and its different phases and tools.

- **Course Outcomes:**

1. Obtain knowledge in different component of systems software and fundamentals of language processing activity.
2. Design two pass assembler and Direct Linking Loaders.
3. Acquire knowledge in different phases and passes of Compiler.
4. Design different types of compiler tools to meet the requirements of the realistic constraints of compilers using LEX and YACC tools.

Text Books & Reference Books

Text Books:

1. Dhamdhere D., "Systems Programming and Operating Systems", McGraw Hill, ISBN 0 - 07 -463579 – 4.
2. A V Aho, R Sethi, J D Ullman, \Compilers: Principles, Techniques, and Tools", Pearson Edition, ISBN 81-7758-590-8.
3. John Donovan, “System Programming”, McGraw Hill, ISBN 978-0--07-460482-3.

Reference Books:

1. John. R. Levine, Tony Mason and Doug Brown, “Lex and Yacc”, O'Reilly, 1998, ISBN: 1- 56592-000-7.
2. Leland L. Beck, “System Software An Introduction to Systems Programming” 3rd Edition, Person Education, ISBN 81-7808-036-2.
3. Adam Hoover, “System Programming with C and Unix”, Pearson,2010

Module II

- Macro processor: Macro Definition, Macro expansion and nested macros
- Loaders: Loader schemes: Types of loaders, direct linking loaders.
- Linkers: Relocation and linking concepts, self-relocating programs, Static and dynamic link libraries.

Introduction to Macro

- In the mid-1950s, when assembly language programming was commonly used to write programs for digital computers, the use of **macro instructions** was initiated for two main purposes:
  - to reduce the amount of program coding that had to be written by generating several assembly language statements from one macro instruction and
  - to enforce program writing standards, e.g. specifying input/output commands in standard ways.
  - Macro instructions were effectively a middle step between assembly language programming and the high-level programming languages that followed, such as FORTRAN and COBOL
  - Eg.
  - **#define** PI 3.14159 //"PI" to be replaced with "3.14159" wherever it occurs
  - **#define** pred(x) ((x)-1) // What this macro expands to depends on what argument *x* is passed to it. Here are some possible expansions:
    - pred(2) → ((2) -1)
    - pred(y+2) → ((y+2) -1)
    - pred(f(5)) → ((f(5))-1)

Introduction

- Macro instructions (macros) are single -line abbreviations for group of instructions.
- Macros are used to provide a program generation facility through **macro expansion**.
- Many languages provide built in facilities for writing macros.

  e.g. PL/I, C, Ada and C++, Assembly languages.
- Generating preprocessors or software tools like Awk of Unix has an equivalent effect.
- **A macro is a unit of specification for program generation through expansion.**

Introduction (contd…)

Macros are defined at the start of the program.

A Macro Definition consists of

- MACRO Pseudo opcode
- Name of macro
- List of formal parameters
- Body of macro(instruction) or Sequence to be abbreviated
- MEND Pseudo opcode

Macro Definition Syntax :-

1) Macro header :- It contains keyword ‘MACRO’.
2) Macro prototype statement syntax :-

   < Macro Name > [& < Formal Parameters >]
3) Model Statements :- It contains 1 or more simple assembly statements, which will replace MACRO CALL while macro expansion.
4) MACRO END MARKER :- It contains keyword ‘MEND’.

Introduction

- Macro name with a set of actual parameters, is replaced by some code, generated from macro body. **This is called macro expansion.**
- **Two types of expansions:**
- **Lexical expansion:** It implies replacement of a character string by another character string during program generation.
- **Semantic expansion:** Generation of type specific instructions for manipulation of byte and word operands.

Example 4.1 (Benefits of semantic expansion) The following sequence of statements is used to increment the value stored in a memory word by a given constant:

1. Move the value from the memory word into a CPU register.
2. Increment the value in the CPU register.
3. Move the new value into the memory word.

Macro-processor

- Many times some blocks of code is repeated in the course of a program
- They may consists of code

-- to save or exchange set of registers

-- to set up linkages

-- to perform a series of arithmetic operations

- In this situation macro instruction facility is useful.
- Programmer defines a single instruction to represent a block of code.
- For every occurrence of this one line instruction the assembler will substitute the entire block.

Macros and Functions

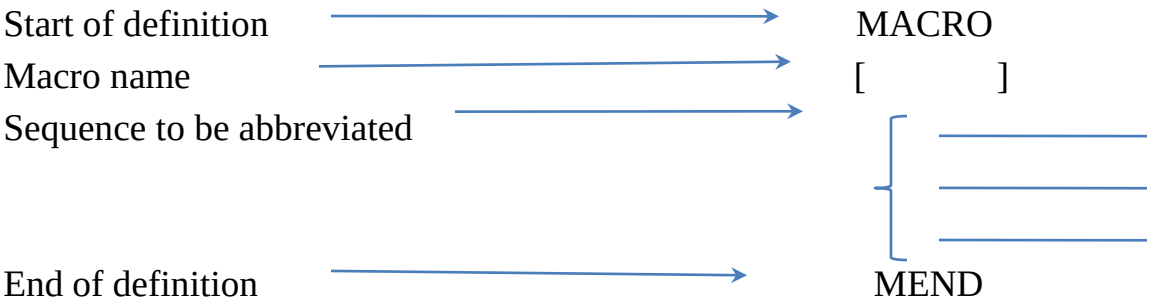
Macros

- Macro is Preprocessed
- No Type Checking is done in Macro
- Using Macro increases the code length
- Speed of Execution using Macro is Faster
- Before Compilation, macro name is replaced by macro value
- Macros are useful when small code is repeated many times
- Macro does not check any Compile-Time Errors
- Does not require CALL and RETURN

Functions

- Function is Compiled
- Type Checking is done in functions
- Using Function keeps the code length unaffected
- Speed of Execution using Function is Slower
- During function call, transfer of control takes place
- Functions are useful when large code is to be written
- Function checks Compile-Time Errors
- Requires CALL and RETURN

Macro Instruction



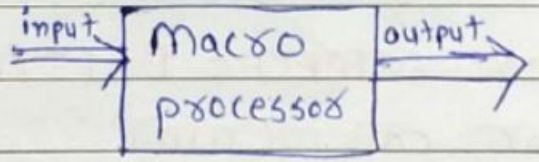
Macro Example

```
MACRO
INCR
        ADD     1,DATA
        ADD     2,DATA
        ADD     3,DATA
MEND
:
INCR
:
INCR
```

Example of macro

* Source Program ⇒

```
MACRO
My MACRO
    ADD AREG,X
    ADD BREG,X
MEND
 START 100
 My MACRO
 MyMACRO
 MyMACRO
  STOP
```



Target Program with Macro expansion ⇒

```
START 100
ADD AREG,X
ADD BREG,X
ADD AREG,X
ADD BREG,X
ADD AREG,X
ADD BREG,X
STOP
```

Example-Macro

- Macro instructions are **single line abbreviations for group of instructions**

e.g.

```
A     1,DATA
A     2,DATA
A     3,DATA
……..
A     1,DATA
A     2,DATA
A     3,DATA

DATA   DC  F’5’
```

	EXPANDED SOURCE
MACRO	
INCR	
A 1,DATA	
A 2,DATA	
A 3,DATA	
MEND	
……	A 1,DATA
INCR	A 2,DATA
………..	A 3,DATA
	…….
……	A 1,DATA
INCR	A 2,DATA
……	A 3,DATA
DATA DC F’5’	
……	

Contd…

Source program with Macro definitions & macro calls

```
MACRO
    INCR  &ARG
    ADD AREG ,&ARG
    ADD BREG,&ARG
MEND
MOVER CREG,LABEL1
INCR DATA1
SUB CREG,LABEL1
---
---
---
```

expanded source without definition & call

```
 MOVER CREG,LABEL1
ADD AREG,DATA1
ADD BREG,DATA1
SUB CREG ,LABEL1
```

Macro features

1.Macro instruction arguments

2.Conditional macro expansion

3. Macro calls within macros

4.Macro instructions defining macros

Macro Instruction Arguments

1.Macro instruction arguments

- Macro facility lacks in flexibility: there is no way for a specific macro call to modify the coding that replaces it.
- So extension of this facility consists of providing for arguments or parameters in macro call
  - dummy arguments
  - positional arguments
  - keyword arguments
  - label arguments

these arguments are preceded by &.

Macro Instruction Arguments Contd…

e.g.

```
…….
……
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’
```

Source

```
        MACRO
        INCR  &ARG
        A     1,&ARG
        A     2,&ARG
        A     3,&ARG
        MEND
        ……
        ……
        INCR  DATA1
        …….
        …….
       INCR  DATA2
        ……
        ……
DATA1  DC  F’5’
DATA2   DC F’10’
……
……
```

Expanded source

```
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’
```

Macro Instruction Arguments (Label Arguments)

```
        MACRO
&LAB  INCR   &ARG1,&ARG2,&ARG3
&LAB  A       1,  &ARG1
      A       2, &ARG2
      A       3,  &ARG3
      MEND
  …….
LOOP1  INCR   DATA1,DATA2,DATA3
  …….
LOOP2  INCR   DATA3,DATA2,DATA1
  ……..
DATA1  DC F’5’
DATA2   DC F’10’
DATA3   DC F’15’
…….
```

```
Loop1  A     1,DATA1
       A     2,DATA2
       A     3,DATA3
 …….
Loop2  A     1,DATA3
       A     2,DATA2
       A     3,DATA1
  ……
DATA1   DC F’5’
DATA2   DC F’10’
DATA3   DC F ’15’
```

Macro Instruction Arguments

Positional arguments

arguments are matched with dummy arguments according to the order in which they appear

e.g. INCR A B C

Keyword arguments:

it allows reference to dummy arguments by name as well as by position

e.g. INCR &ARG1=A,&ARG3=C,&ARG2=B

Macro Instruction Arguments (Positional Arguments)

```
        MACRO
        INCR   &ARG1,&ARG2,&ARG3,&LAB
&LAB    A       1,  &ARG1
        A       2, &ARG2
        A       3,  &ARG3
        MEND
  …….
        INCR   DATA1,DATA2,DATA3,LOOP1
  …….
        INCR   DATA3,DATA2,DATA1,LOOP2
  ……..
DATA1   DC F’5’
DATA2   DC F’10’
DATA3   DC F’15’
…….
```

2 Pass Macroprocessor

- Recognize macro definitions
- Save the definitions
- Recognize calls
- Expand calls and substitute arguments

2 Pass Macroprocessor structure

- **Pass I**

1. **Input** macro source deck
2. **Output** macro source deck copy for use by pass 2
3. Macro definition table (**MDT**) ,used to store body of macro def.
4. Macro name table (**MNT**) ,used to store names of defined macros
5. **MDTC**-MDT counter ,used to indicate next available entry in MDT
6. **MNTC**-MNT counter, used to indicate next available entry in MNT
7. Argument List Array(**ALA**) used to substitute index markers for dummy arguments before storing macro definition

Example

MACRO
M1 &ARG1,&ARG2
ADD AREG &ARG1
ADD BREG &ARG2
MEND
MACRO
M2 &ARG3,&ARG4
SUB AREG &ARG3
SUB BREG &ARG4
MEND
START 300
MOVER AREG S1
MOVEM BREG S2
M1 D1 D2
MOVER AREG S1

M2 D3 D4
PRINT S1
PRINT S2
S1 DC 5
S2 DC 6
END

Contd…

- **Pass 2**

1. Copy of **input** macro source deck
2. **Output** expanded source deck to be used as input to assembler
3. **MDT**, created by Pass 1
4. **MNT**, created by pass 1
5. **MDTP-** MDT pointer used to indicate next line of text to be used during macro expansion
6. **ALA** ,used to substitute macro call arguments for the index markers in the stored macro definition.

MDT

- **Macro definition table(MDT)**
- Used to store macro definition
- Created by pass-1
- Pass 1 identifies and stores all macro definitions in MDT
- Pass 2 can identify macro calls and expand these calls by using their macro definitions stored in MDTs
- Every line of macro definition except MACRO is stored in MDT as MACRO is not used to expand macro
- MEND indicates end of macro definition so it is stored in MDT
- MDT has 80 bytes per entry

Index	instruction

MNT

- **Macro name table(MNT)**
- Used to store name of macros and corresponding MDT index
- Created by Pass 1 and used by Pass 2
- So that pass 2 can decide whether opcode of this source instruction is macro call or not by searching through name field of MNT.

MNT index	Macro Name	MDT index

- **MDTC(Macro def table counter)**

  this variable stores the last count from the MDT table
- **MNTC(Macro name table counter)**

  this variable stores the no. of macros defined in the program

ALA

- **Argument List Array(ALA)**
- Used for association of dummy arguments and actual arguments
- Partially table is created by pass 1 where pass 1 associates an unique integer number with each dummy arguments in the order in which they appear
- Macros are stored in MDT by using these numbers
- It is partially constructed and used by pass 2
- It keeps track of dummy parameters when the macro is being defined and maintain the actual arguments when expanding the call

Integer index	dummy argument	Actual argument

Example

```
MACRO
   INCR &ARG1
   MOVER AREG,& ARG1
   ADD AREG,& ARG1
   MOVEM AREG,& ARG1
MEND

START
MOVEM BREG, A
ADD CREG,=‘1’
SUB CREG,A
INCR DATA1
MUL CREG,=‘1’
END
```

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,=‘1’
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	

MNT

MNT Index	Macro Name	MDT index
1		

ALA

ALA Index	Formal Argument
#1	

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A
3	ADD AREG,B

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A
3	ADD AREG,B
4	MOVEM AREG,A

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Macro processor example

START 200
MACRO
INCR &ARG1,&ARG2
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
MEND
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A
3	ADD AREG,B
4	MOVEM AREG,A
5	MEND

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Output of Pass 1

Output file

START 200
MOVER AREG,='1'
MOVEM BREG,M
INCR DATA1,DATA2
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A
3	ADD AREG,B
4	MOVEM AREG,A
5	MEND

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Output of Pass 2

Final Expansion

START 200
MOVER AREG,='1'
MOVEM BREG,M
MOVER AREG, A
ADD AREG,B
MOVEM AREG,A
DATA1 DC 5
DATA2 DC 10
END

MDT

MDT index	Instruction
1	INCR &ARG1,&ARG2
2	MOVER AREG, A
3	ADD AREG,B
4	MOVEM AREG,A
5	MEND

MNT

MNT Index	Macro Name	MDT index
1	INCR	1

ALA

ALA Index	Formal Argument
#1	&ARG1
#2	&ARG2

Contd…

```
        MACRO
&LAB  INCR   &ARG1,&ARG2,&ARG3
&LAB   MOVER   AREG,  &ARG1
         A DD        AREG, &ARG2
         MOVEM    BREG,  &ARG3
        MEND

  …….
LOOP1  INCR   DATA1,DATA2,DATA3
  …….
LOOP2  INCR   DATA3,DATA2,DATA1
  ……..
DATA1  DC F’5’
DATA2   DC F’10’
DATA3   DC F’15’
…….
```

Example Contd…

- MDT

Index	instruction
1	&LAB INCR &ARG1,&ARG2,&ARG3
2	#0 MOVER AREG,#1
3	ADD AREG,#2
4	MOVEM BREG ,#3
5	MEND

ALA

Index	arguments
#0	“LOOP1BBB”
#1	“DATA1BBB”
#2	“DATA2BBB”
#3	“DATA3BBB”

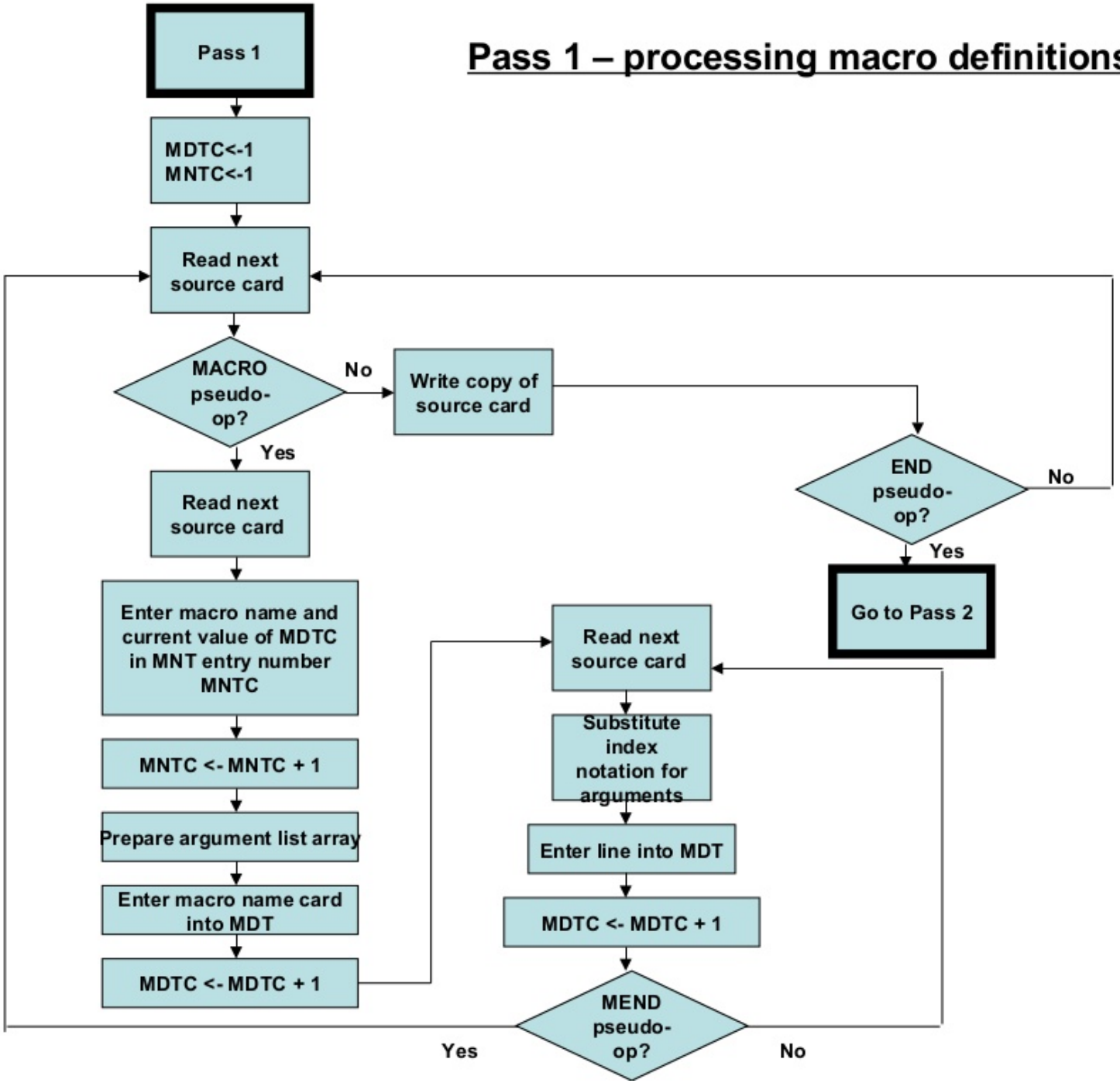
- MNT

Index	Name(8 bytes)	MDT index (4 bytes)
1	“INCRbbbb”	1

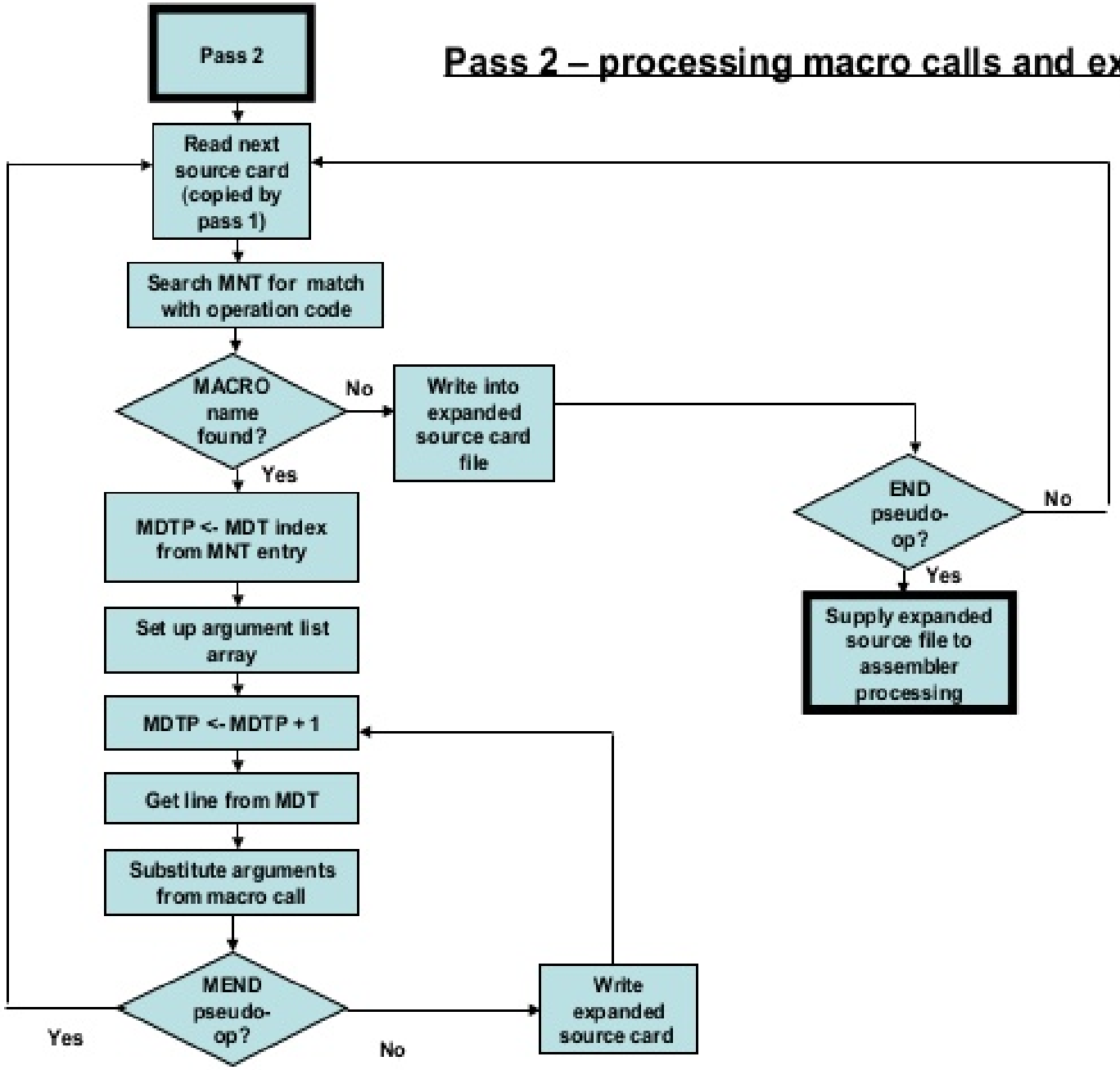
ALA

Index	arguments
#0	“BBBBBBB”
#1	“DATA3BBB”
#2	“DATA2BBB”
#3	“DATA1BBB”

Pass 1 – processing macro definitions



Pass 2 – processing macro calls and expansion



Contd..

```
START
……..
MACRO
INCR    &ARG1,&ARG2,&ARG3
 A          1,  &ARG1
 A          2, &ARG2
 A          3,  &ARG3
MEND
 …….
 INCR   DATA1,DATA2,DATA3
 …….
 INCR   DATA3,DATA2,DATA1
 ……..
 DATA1  DC F’5’
 DATA2   DC F’10’
 DATA3   DC F’15’
 …….
 END
```

Contd…

```
START
  MOVER  AREG,A
  MOVEM AREG,B
  ……
  MACRO
  VARY    &ARG1,&ARG2
  L      1,F'5'
  A      1,&ARG1
  A      1,&ARG2
  MEND
  MACRO
  INCR    &ARG1,&ARG2,&ARG3
  A          1,  &ARG1
  A          2, &ARG2
  A          3,  &ARG3
  MEND
  …….
  INCR   DATA1,DATA2,DATA3
  …….
  VARY    DATA3,DATA2
  ……..
  DATA1  DC F'5'
  DATA2   DC F'10'
  DATA3   DC F'15'
  …….
  END
```

Conditional macro expansion

```
            MACRO
&ARG0    VARY    &COUNT,&ARG1,&ARG2,&ARG3
&ARG0     A          1,  &ARG1
          A IF       (&COUNT EQ 1).FINI          TEST IF &COUNT=1
          A          2,  &ARG2
          A IF       (&COUNT EQ 2).FINI          TEST IF &COUNT=2
          A          3,  &ARG3
 .FINI          MEND
  …….
LOOP1   VARY   3, DATA1,DATA2,DATA3
  …….
LOOP2   VARY    2,  DATA3,DATA2
  ……..
LOOP3   VARY    1,  DATA1

DATA1   DC F’5’
DATA2   DC F’10’
DATA3   DC F’15’
…….
```

Expanded code

```
  ……
Loop1  A     1,DATA1
       A     2,DATA2
       A     3,DATA3
  …….
Loop2  A     1,DATA3
       A     2,DATA2
       ……
Loop1  A     1,DATA1
  …….
DATA1  DC F’5’
DATA2  DC F’10’
DATA3  DC F ’15’
```

Contd…

- Pseudo opcodes AIF & AGO
- .FINI (labels starting with a period(.) are macro labels)and do not appear in the output of the macro processor
- AIF (conditional branch pseudo opcodes)
- AGO (unconditional branch pseudo opcodes) or goto statement.
- AIF & AGO control the sequence in which the macro processor expands the statements in macro instruction

Macro calls within macros

```
MACRO
ADD1     &ARG
MOVER  AREG, &ARG
A DD AREG, =‘1’
MOVEM   AREG,&ARG
MEND

MACRO
ADDS      &ARG1,&ARG2,&ARG3
ADD1       &ARG1
ADD1       &ARG2
ADD1       &ARG3
MEND
……
ADDS   DATA1,DATA2,DATA3
……
DATA1  DC  F’5’
DATA2  DC  F’10’
DATA3  DC  F’15’
……..
```

Contd…

```
MACRO
ADD1    &ARG
L       1,&ARG
A       1,F'1'
ST      1,&ARG
MEND

MACRO
ADDS    &ARG1,&ARG2,&ARG3
ADD1    &ARG1
ADD1    &ARG2
ADD1    &ARG3
MEND
…….
……

ADDS  DATA1,DATA2,DATA3

……
……
DATA1  DC  F'5'
DATA2  DC  F'10'
DATA3  DC  F'15'
……..
```

expanded source level 1

```
ADD1 DATA1

ADD1 DATA 2

ADD1 DATA3
```

expansion of ADDS

expanded source level 2

```
L    1,DATA1
A    1,F'1'
ST   1,DATA1
L    1,DATA2
A    1,F'1'
ST   1,DATA2
L    1,DATA3
A    1,F'1'
ST   1,DATA3
```

expansion of ADD1

Macro instructions defining macros

Def.of Macro define

def of macro &SUB

```
MACRO
DEFINE &SUB          macro name: DEFINE
MACRO
&SUB &Y              dummy macro name
CNOP 0,4
BAL 1,*+8            set register 1 to parameter list ptr
DC  A(&Y)            parameter list ptr
L   15,=V(&SUB)      address of subroutine
BALR  14,15          transfer control to subroutine
MEND
MEND
…..
…..
DEFINE COS           outer macro
…..
……
COS AR               inner macro
```

A single pass algorithm for macro processor

- **MDI** : macro definition input indicator
  works like switch
  keeps track of macro call

  MDI is ON
  -during expansion of a macro call
  -lines are read from MDT

  MDI is OFF
  all other times

  --the reading of MEND line indicates end of macro and terminates expansion of a call ,MDI is set off next line is obtained from regular i/p stream.

- **MDLC** : macro definition level counter
  -- keeps track of macro definition.
  -- MDLC is incremented by 1 when a MACRO is encountered and decremented by 1 when MEND occurs
  -- MDLC is used to insure that the entire macro def. including MACROs & MENDs get stored in MDT

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions
1	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1		

ALA

ALA Index	Formal Argument	Actual Argument
#1		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions
1	DEFINE &SUB
2	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions
1	DEFINE &SUB
2	MACRO
3	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions
1	DEFINE &SUB
2	MACRO
3	#1 &Y
4	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions
1	DEFINE &SUB
2	MACRO
3	#1 &Y
4	CNOP 0,4
5	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9		
10		
11		
12		
13		
14		
15		
16		
17		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11		
12		
13		
14		
15		
16		
17		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2		

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2		

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14	DC	A(#1)
15		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14	DC	A(#1)
15	L	15,=V(COS)
16		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14	DC	A(#1)
15	L	15,=V(COS)
16	BALR	14,15
17		

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14	DC	A(#1)
15	L	15,=V(COS)
16	BALR	14,15
17	MEND	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	

Expansion of Define Cos call

```
START
   ….
MACRO
   DEFINE &SUB
   MACRO
   &SUB &Y
   CNOP 0,4
   BAL 1,*+8
   DC  A(&Y)
   L   15,=V(&SUB)
   BALR  14,15
   MEND
   MEND
   …..
   …..
   DEFINE COS
   …..
   ……
   COS AR
 …….
 END
```

MDT

Index	instructions	
1	DEFINE	&SUB
2	MACRO	
3	#1	&Y
4	CNOP	0,4
5	BAL	1,*+8
6	DC	A(&Y)
7	L	15,=V(#1)
8	BALR	14,15
9	MEND	
10	MEND	
11	COS	&Y
12	CNOP	0,4
13	BAL	1,*+8
14	DC	A(#1)
15	L	15,=V(COS)
16	BALR	14,15
17	MEND	

MNT

MNT INDEX	MACRO NAME	MDT INDEX
1	DEFINE	1
2	COS	11

ALA

ALA Index	Formal Argument	Actual Argument
#1	&SUB	COS
#2	&Y	AR

Expansion of COS AR call

```
CNOP    0,4
BAL     1,*+8
DC      A(#1)
L       15,=V(COS)
BALR    14,15
```

```
START
…..
…..
MACRO
ADD1     &ARG
L            1,&ARG
A            1,F’1’
ST           1,&ARG
MEND
….
MACRO
ADDS      &ARG1,&ARG2,&ARG3
ADD1       &ARG1
ADD1       &ARG2
ADD1       &ARG3
MEND
……
ADDS   DATA1,DATA2,DATA3
……
DATA1  DC  F’5’
DATA2  DC  F’10’
DATA3  DC  F’15’
……..
END
```

```
START
…..
…..
MACRO
ADD1     &ARG
L          1,&ARG
A          1,F’1’
ST         1,&ARG
MEND
….
MACRO
ADDS     &ARG1,&ARG2,&ARG3
ADD1      &ARG1
ADD1      &ARG2
ADD1      &ARG3
MEND
……
ADDS   DATA1,DATA2,DATA3
……
DATA1  DC  F’5’
DATA2  DC  F’10’
DATA3  DC  F’15’
……..
END
```

Index	instructions
1	ADD1 &ARG
2	L 1,#1
3	A 1,F’1’
4	ST 1 ,#1
5	MEND
6	ADDS &ARG1,&ARG2,&ARG3
7	ADD1 #1
8	ADD1 #2
9	ADD1 #3
10	MEND

MNT INDEX	MACRO NAME	MDT INDEX
1	ADD1	1
2	ADDS	6

Contd…

- MDTP= 6 when ADDS is called
- And MDI is set ON .
- Then READ function increments MDTP , gets line from the MDT(line 7)
- Then MDTP = MDTP+1 =7
- So it is ADD1 DATA1

Then MDTP =1 so here previous value of MDTP i.e. 7 will be lost.

This the problem with macro calls within macros

- So it will work recursively.
- means to process one macro before it is finished with another then to continue with the previous or outer.
- Recursive procedures usually operate by means of stack.
- Each stack frame is associated with each recursive call
- Here status of unfinished computations is preserved.

References

- System Programming & Operating System - TE: Macro Definition and call, Nested Macro Calls, (wikinote.org)