System Programming and Compiler Construction

CSC 602

Subject Incharge

Varsha Shrivastava

Assistant Professor

email: varshashrivastava@sfit.ac.in

Room No: 407

CSC 602 System Programming and Compiler Construction

Module 3

Macros and Macro Processor

Contents as per syllabus

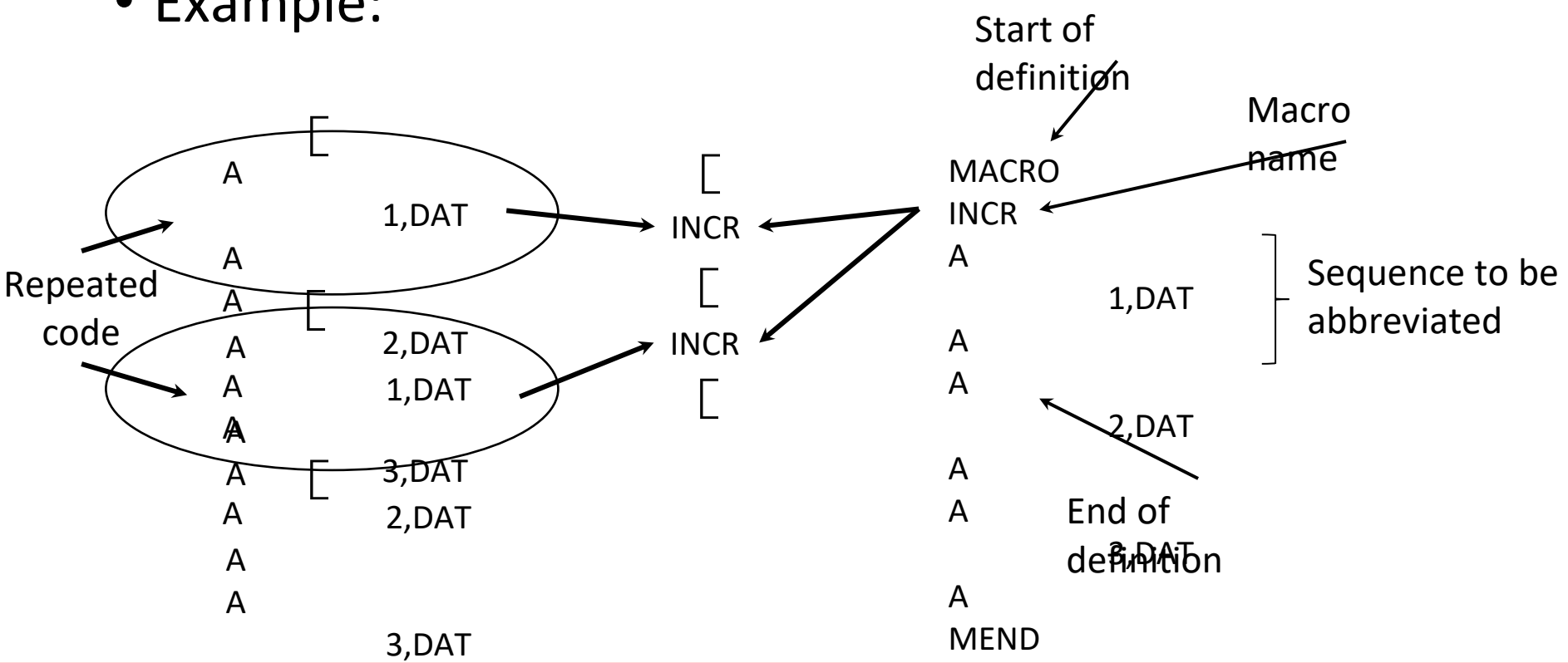
- Introduction
- Macro definition and call
- Features of Macro facility: Simple, parameterized, conditional and nested.
- Design of Single pass Macro Processor, data structures used.

Introduction

- Macro instructions or **Macros** are single line abbreviations for groups of instructions.
- Single instruction is used to represent a block of code.
- For every occurrence of this one line macro instruction, the macro processing assembler will substitute the entire block.

Macro Instruction

- Macro is an abbreviation for a series of operations.
- Example:



Macros in C

```
#include <stdio.h>
#define Square(x) ((x)*(x))

int main(void)
{
int a;
printf("enter a no : ");
scanf("%d",&a);
Square(a);
}
```

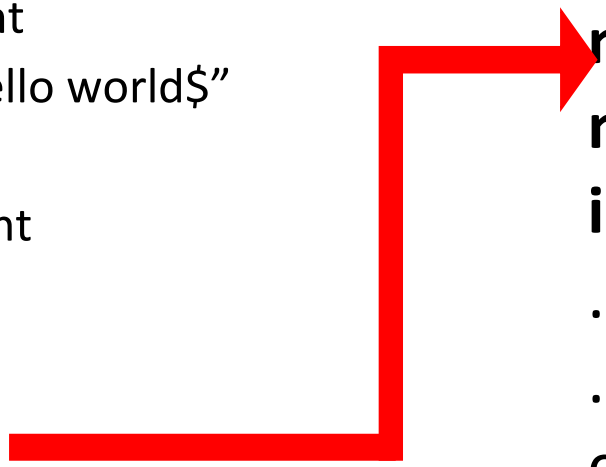
```
#include <stdio.h>
//Macro definition not
included

int main(void)
{
int a;
printf("enter a no : ");
scanf("%d",&a);
((a)*(a));
}
```

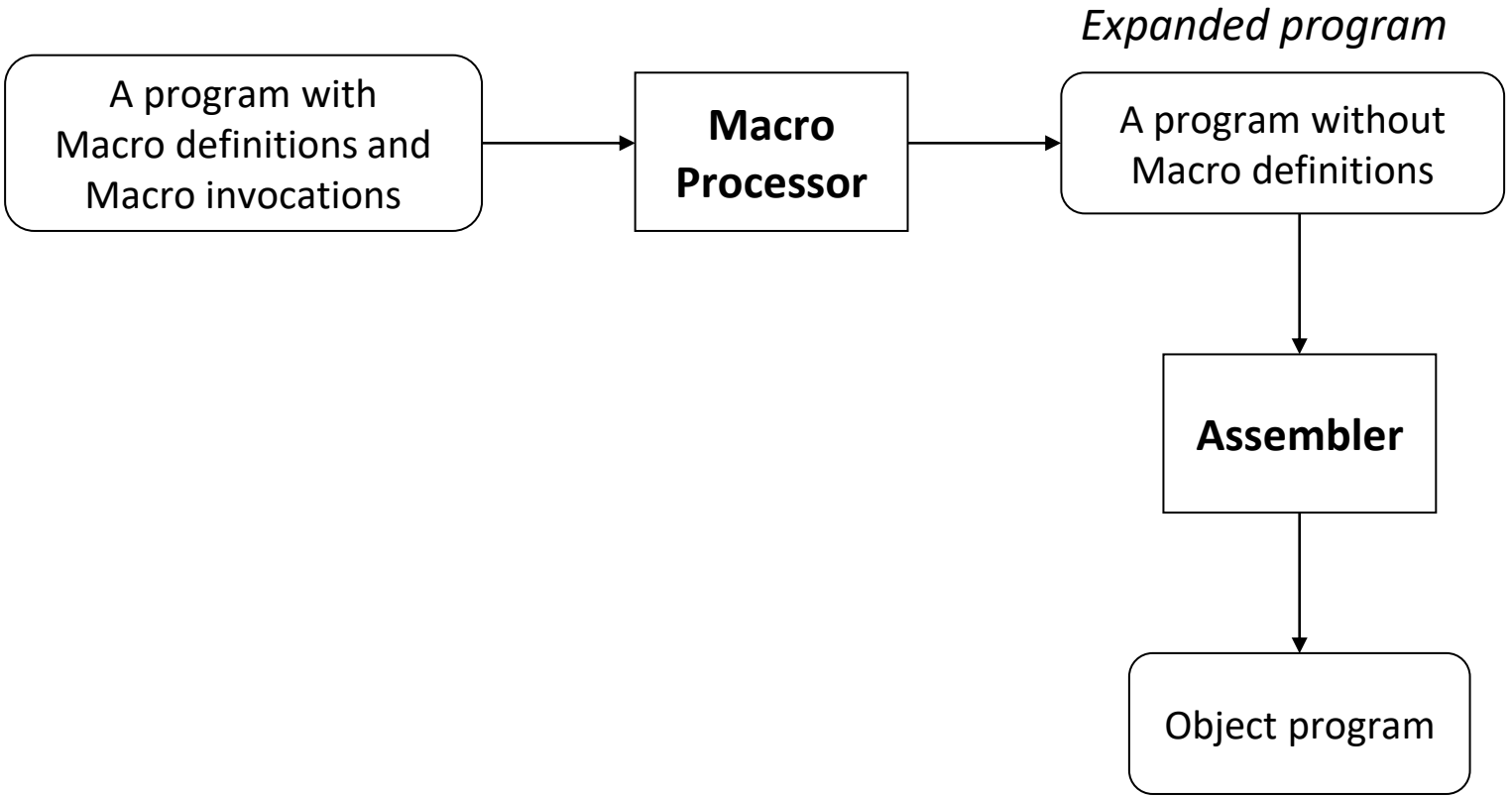
Macros in Assembly code (8086)

```
macro print msg
    mov dx, offset msg
    mov ah, 09h
    int 21h
endm
data segment
Msg1 db “hello world$”
data ends
code segment
.
.
print msg1
.
.
end start
```

```
start:
.
.
mov dx, offset msg
mov ah, 09h
int 21h
.
.
code ends
end start
```



Basic Macro Processor functions



Macro Processor is an in-built functions of Assembler

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

- Macro calls replaces the call by a block of code.
- No flexibility to modify code that replaces the call.
- Extension for providing arguments or parameters in macro call.
- Macro instruction argument (dummy arguments) are used in definition.
- It is specified in the macro name line and distinguished by '&'
- Arguments that are not specified, are presumed blank by macro processor.

Features of Macro Facility

Macro Instruction Arguments

```
A   1,FIVE
A   2,FIVE
A   3,FIVE
------
------
------


A   1,FOUR
A   2,FOUR
A   3,FOUR

FIVE   DC   F'5'
FOUR   DC  F'4'
```

Features of Macro Facility

Macro Instruction Arguments

```
A   1,FIVE
A   2,FIVE
A   3,FIVE
------
------
------


A   1,FOUR
A   2,FOUR
A   3,FOUR

FIVE   DC   F'5'
FOUR   DC   F'4'
```

```
MACRO
ADDM &ARG
A   1, &ARG
A   2, &ARG
A   3, &ARG
MEND
------
------
------
ADDM FIVE


ADDM FOUR

FIVE   DC   F'5'
FOUR   DC   F'4'
```

Features of Macro Facility

Macro Instruction Arguments

When we pass more than one argument there are 2 ways to specify these arguments:-

1.Positional Arguments
2.Keyword Arguments.

Features of Macro Facility

Positional Parameter

```
MACRO
M1  &P1,&P2,&P3
----
----
-----
MEND
M1  A, B, C
```

Keyword Parameter

```
MACRO
M1  &P1=,&P2=,&P3=
----
----
-----
MEND
M1  &P1=A, &P2=B, &P3=C
```

Features of Macro Facility

Default Parameter

```
MACRO
M1  &P1=A, &P2=B
----
----
-----
MEND
M1 &P1=, &P2=C
```

Features of Macro Facility

Macro Instruction Arguments

Original Source (single argument)

```
MACRO
INCR    &ARG
A       1,&ARG
A       2,DATA1
INCR    &ARG
A       3,DATA2
MEND
```

Expanded Source

```
A       1,DATA1
A       2,DATA1
A       3,DATA1
A       1,DATA2
A       2,DATA2
A       3,DATA2
```

Original Source (Multiple argument)

```
MACRO
INCR   &ARG1,&ARG2
A      1, &ARG1
A      2, &ARG2
MEND
INCR   DATA1,DATA2
INCR   DATA2,DATA1
```

Expanded Source

```
A      1,DATA1
A      2,DATA2
A      1,DATA2
A      2,DATA1
```

Features of Macro Facility

Conditional Macro Expansion

- **AIF** and **AGO** permit conditional reordering of the sequence of macro expansion.
- AIF
  - Conditional branch
  - Performs arithmetic test and branches if condition is true
- AGO
  - Unconditional branch (similar to 'go to' statement)
- Machine instructions that appear in the expansion of a macro call can be selected based on condition.
- Labels inside a Macro start with a period (.) eg: .FINISH

Features of Macro Facility

Conditional Macro Expansion

Original Source

```
Loop1   A   1,DATA1
        A   2,DATA2
        A   3,DATA3

Loop2   A   1,DATA3
        A   2,DATA2

Loop3   A   1,DATA1
```

Source with Macro

```
        MACRO
&ARG0   VARY    &COUNT, &ARG1, &ARG2,&ARG3
&ARG0   A       1, &ARG1
        AIF     (&COUNT EQ 1).FINISH
        A       2,&ARG2
        AIF     (&COUNT EQ 2).FINISH
        A       3,&ARG3
.FINISH MEND

Loop1   VARY    3,DATA1,DATA2,DATA3

Loop2   VARY    2,DATA3,DATA2

Loop3   VARY    1,DATA1
```

Expanded Source

```
Loop1   A 1,DATA1
        A 2,DATA2
        A 3,DATA3

Loop2   A 1,DATA3
        A 2,DATA2

Loop3   A 1,DATA1
```

Features of Macro Facility

Macro calls within Macros

- Also known as nested macro calls.
- A macro can be called within another macro.
- A macro can call itself (using AIF or AGO) so long as it doesn't go into an infinite loop.
- Macro calls within macros can have several levels

Features of Macro Facility

Macro calls within Macros

Source

```
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,DATA
3
```

Expanded Source (Level 1)

Expansion of ADDS

```
        [
    [   ADD1    DATA1
    |
    |   ADD1    DATA2
    |
    [   ADD1    DATA3
        [
```

Expanded Source (Level 2)

Expansion of ADD1

```
        [
    [   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
        [
```

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).

Features of Macro Facility

Macro Instruction defining Macros

Definition of macro DEFINE

```
MACRO
DEFINE &SUB
```

Macro name: DEFINE

Definition of macro &SUB

```
MACRO
&SUB    &Y
CNOP    0,4
BAL     1,*+8
DC      A(&Y)
L       15,=V(&SUB)
BALR    1          4,15
subroutine
MEND
```

Dummy macro name
Align boundary
Set reg 1 to parameter list pointer
Parameter list pointer
Address of subroutine
Transfer control to

```
MEND
```

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

Address of AR
V denotes Address of external symbol

Implementation of Macros

There are 4 tasks that any macro instruction processor perform:

1. Recognize macro Definition
2. Save the Definition
3. Recognize calls
4. Expand calls and Substitute arguments.

2-Pass Macro Processor

- Assumptions
  - Functionally different from assembler
  - No nested macro calls or macro within macro definitions
- Assembler scans and processes lines of text.
  - A line can refer to another line by its address or name
  - Address or name **must** be available to assembler
- Macro definitions do not refer to anything outside themselves.
  - Macro calls refer only to macro definitions

2-Pass Macro Processor

- Pass 1: handles macro definitions (Database)
  - Input macro source deck
  - Output macro source deck copy (for pass 2)
  - Macro Definition Table (MDT) (to store definition body)
  - Macro Name Table (MNT) (to store names of defined macros)
  - Macro Definition Table Counter (MDTC) (next available entry in the MDT)
  - Macro Name Table Counter (MNTC) (next available entry in the MNT)
  - Argument List Array (ALA) (to substitute index markers for dummy arguments before storing macro definition)

2-Pass Macro Processor

- Pass 2: handles macro calls and expansion (Database)
  - Copy of input macro source deck
  - Output expanded source deck (i/p for assembler)
  - MDT
  - MNT
  - ALA

    from Pass 1
  - Macro Definition Table Pointer (MDTP) (indicates the next line of text to be used during macro expansion)

2-Pass Macro Processor

- Macro Definition Table
  - Table of text lines
  - 80 byte string entries
  - Each line of macro definition is stored in MDT
  - MACRO is omitted but MEND is kept to indicate the end.
  - Name line is kept to facilitate keyword replacement

Index	Card		
15	&LAB	INCR	&ARG1, &ARG2, &ARG3
16	#0	A	1,#1
17		A	2,#2
18		A	3,#3
19		MEND	

MDT

2-Pass Macro Processor

- Macro Name Table (MNT)
  - Each entry consists of a character string (macro name) and pointer to entry in MDT
- Argument List Array (ALA)
  - Dummy arguments in definition replaced by index markers (eg. #1) in pass 1
  - Index markers replaced by arguments in macro call

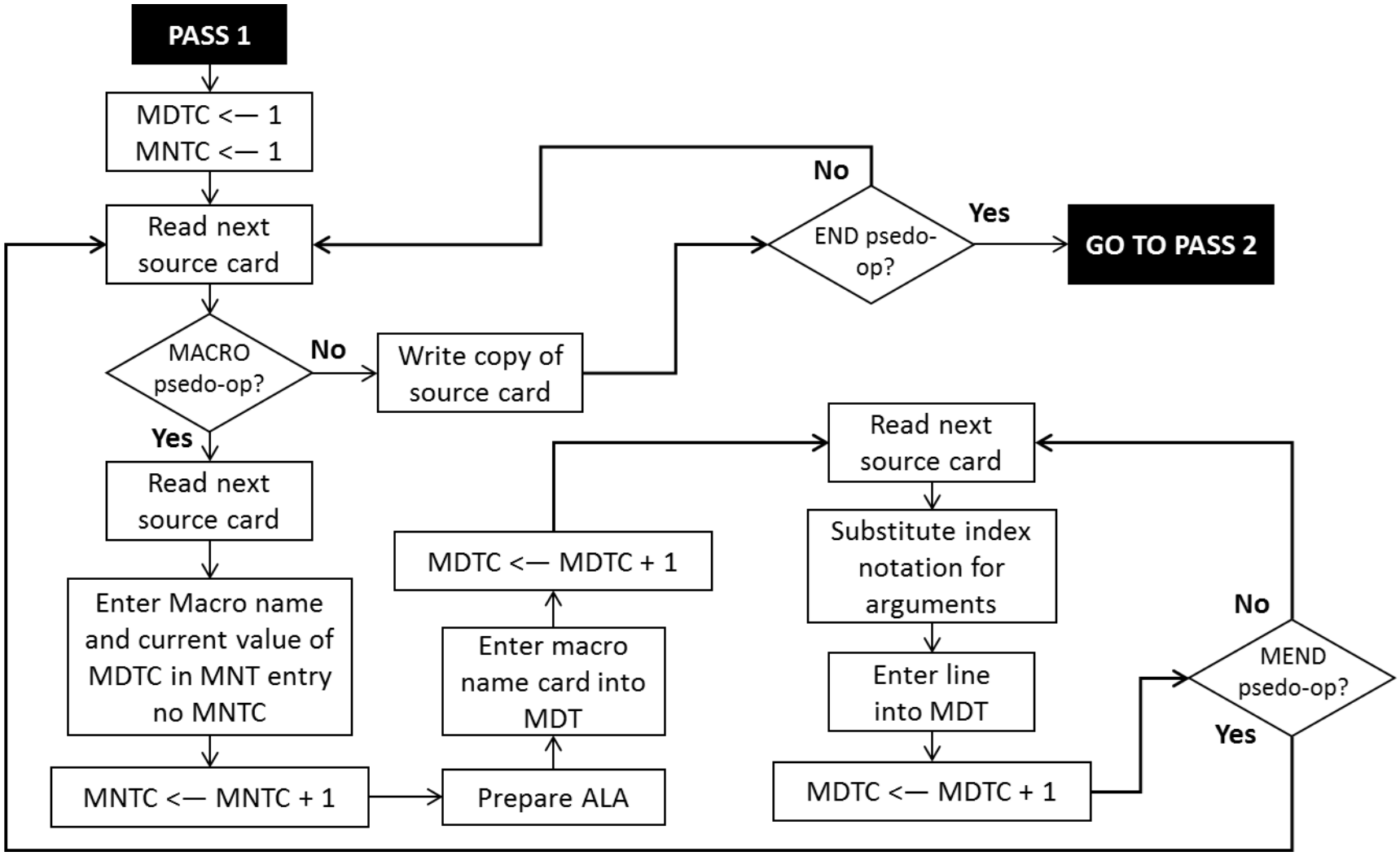
Index	8 bytes Name	4 bytes MDT Index
⋮	⋮	⋮
3	"INCRbbbb"	15
⋮	⋮	⋮

MNT

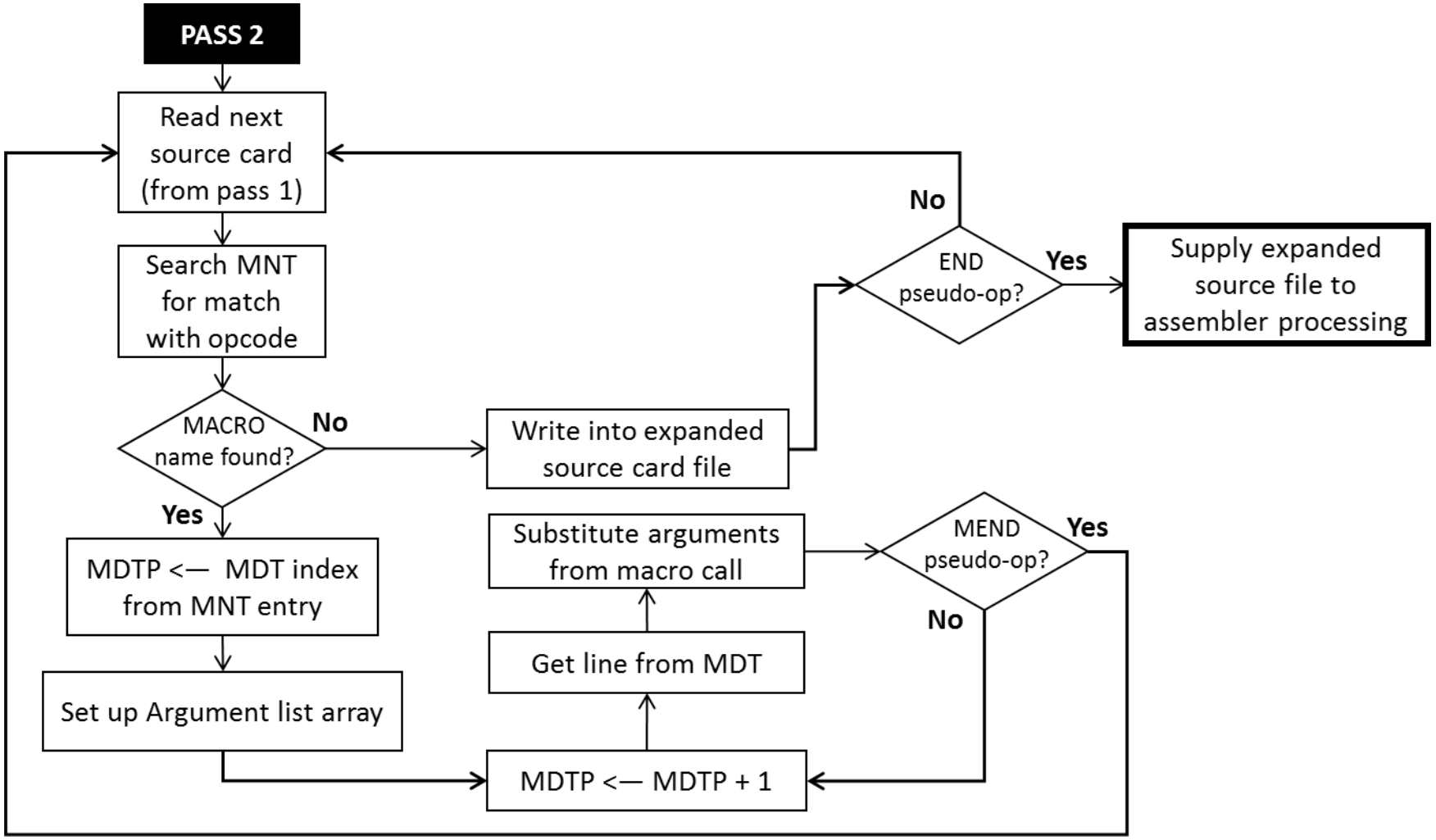
Index	8 bytes Argument
0	"LOOP1bbb"
1	"DATA1bbb"
2	"DATA2bbb"
3	"DATA3bbb"

ALA

Macro Pass 1 Flow Chart



Macro Pass 2 Flow Chart



Example

```
MACRO
    &LAB        INCR    &ARG1, &ARG2, &ARG3
    &LAB        A       1, &ARG1
                A       2, &ARG2
                A       3, &ARG3
MEND
        .
        .
LOOP1   INCR    DATA1,DATA2,DATA3
        .
        .
LOOP2   INCR    DATA1,DATA2,DATA3
        .
        .
```

Example Pass1

MNT

Index	Card	MDT Index
1	INCRbbbb	1

ALA

Index	Argument
0	&LAB
1	&ARG1
2	&ARG2
3	&ARG3

MDT

Index	Card		
1	&LAB	INCR	&ARG1,&ARG2,&ARG3
2	#0	A	1,#1
3		A	2,#2
4		A	3,#3
5		MEND	

MDTC	6
MNTC	2

Example Pass 2

MNT

Inde x	Card	MDT Index
1	INCRbbbb	1

ALA

Index	Argument
0	LOOP1bbb
1	DATA1bbb
2	DATA2bbb
3	DATA3bbb

MD

Index	Card
2	LOOP1 A 1,DATA1
3	A 2,DATA2
4	A 3,DATA3
5	MEND

MDTP	5

University Questions

- Define Macro. Explain macro calls within macro giving example
- What is positional parameter in macro?
- Explain two pass macro processor with flowchart and databases
- Explain the different ways of parameter passing in macros?
- Detail the different features used in macro processing.

Practice Questions

```
ABC     START
        MACRO
        ADD   &ARG1, &ARG2
        L     1, &ARG1
        A     1, &ARG2
        MEND
        MACRO
        SUB   &ARG3, &ARG4
        L     1, &ARG3
        S     1, &ARG4
        MEND
        ADD DATA1, DATA2
        SUB DATA1, DATA2
DATA1   DC    F'9'
DATA2   DC    F'5'
        END
```