UNIVERSITY INSTITUTE of COMPUTING

Asia's Fastest Growing University

System Software
CAT-204

Design By:

Prof. Pawandeep Sharma

A.P

Chandigarh University-Gharuan

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-I

Introduction to System Software: Machine Structure, evolution of operating system, machine language.

Assembler: Elements of Assembly Language Programming, General design procedure, design of a Two Pass Assemblers, A Single Pass Assemblers Design.

Table Processing: Searching & Sorting.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-II

Macro and Macro Processors: Macro instructions, Features of a macro Facility: macro Instruction arguments, Conditional macro expansion, Macro calls within macros, Macro instruction defining macros, Advanced Macro Facilities, Implementation of simple macro processor, Two-pass algorithm, Implementation of macro calls within macros, Implementation within an assembler.

Linkers – Translated linked and load time addresses, relocation and linking concepts, Design of a linker, self relocating programs.

Syllabus

UNIT-III

Loaders: Loader scheme, absolute loaders, Subroutine linkages, Relocating loaders, Direct linking loaders, binders, linking loaders, overlays, Dynamic Binders, Design of an Absolute Loader, Design of a Direct-Linking Loader. Compilers: Phases of Compiler Construction, Symbol Table, Top-down and bottom-up Parsing, Operator-Precedence Parsing, LR Parsers, Code Generation and Code Optimization, Memory management, Design & other issues.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro & Macro Processor

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Introduction to Macro

•Sometimes need to repeat some blocks of code several times in our program/task. In such cases to avoid this repetition the system software will be provided one special component i.e., MACRO

•Macro instructions is a notational convenience for the programmer, it allows the programmer to write a short hand version of a program

•Definition: Macro instructions are the single line of abbreviation for group of instructions

Advantages of Macro

The Macros provide several advantages when writing assembly programs:

•The frequent use of macros can reduce programmer-induced errors. A macro allows you to define instruction sequences that are used repetitively throughout your program. Subsequent use of the macro faithfully provide the same results each time.

•The scope of symbols used in a macro is limited to that macro. You need not be concerned about using a previously used symbol name.

Macro vs. Subroutine

1.Macro doesn't have any return statement...but a subroutine can have ..

2.Execution time needed for a macro is much lesser than subroutine.

3.Memory requirement for a macro is generally higher than subroutine.

4.Macro is always local to the program that defines it..subroutine may or may not be local.

5.A macro call results in macro expansion, whereas subroutine call results in execution.

6.Macro expansion increases the size of the program but subroutine execution doesn’t affect the size of the program.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro Processor

- A macro processor enables you to define and to use macros in your assembly programs.
- When you define a macro, you provide text (usually assembly code) that you want to associate with a macro name.
- Then, when you want to include the macro text in your assembly program, you provide the name of the macro. The assembler replaces the macro name with the text specified in the macro definition.
- The design of macro processor is generally machine independent.
- Every programming language must be uses the macro processor
- Ex: C-programming uses a macro processor to support for defining symbolic constant #define, #include

Basic Macro-Processor Functions

There are 3 main functions of macro processor. which are as follows:

1. Macro definition
2. Macro calls or invocation
3. Macro expansion

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro Definition

• Macro definition attaches a name to a sequence of instruction

•Structure of macro-definition is as follows :

MACRO	**Starting of macro definition**
[ ]	**Give a macro name**
…….	
…….	**Sequence of instruction**
…….	
MEND	**Ending of macro definition**

•The macro definition starts with MACRO pseudo op code. It indicates beginning of the macro definition.

•The Macro-definition terminated with the MEND pseudo op code.

Macro Definition Example 1

Ex: Sequence of instruction will be

```
A 1, data
A 2, data
A 3, data
  -----
  -----
A 1, data
A 2, data
A 3, data
  -----
  -----
A 1, data
A 2, data
A 3, data
  -----
  -----
Data DC f '5'
```

In the above example the sequence of A 1, data
A 2, data
A 3, data

It repeats 3 times MACRO permits as to attach a name to this sequence and to use the name in its place then the macro definition will be

Macro Definition Example 1

```
MACRO
Add
A 1, data
A 2, data
A 3, data
MEND
```

Where

MACRO=> is a pseudo op indicates beginning of definition

Add => is the name of the macro

MEND => is a pseudo op indicate end of the macro definition

Between the name of the Macro Add and MEND we have the sequence of instruction.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

MACRO Calls or Invocation

•Once the macro has been defined, the macro name can be used at the point of call in the place of sequence code this is referred as macro call or invocation.

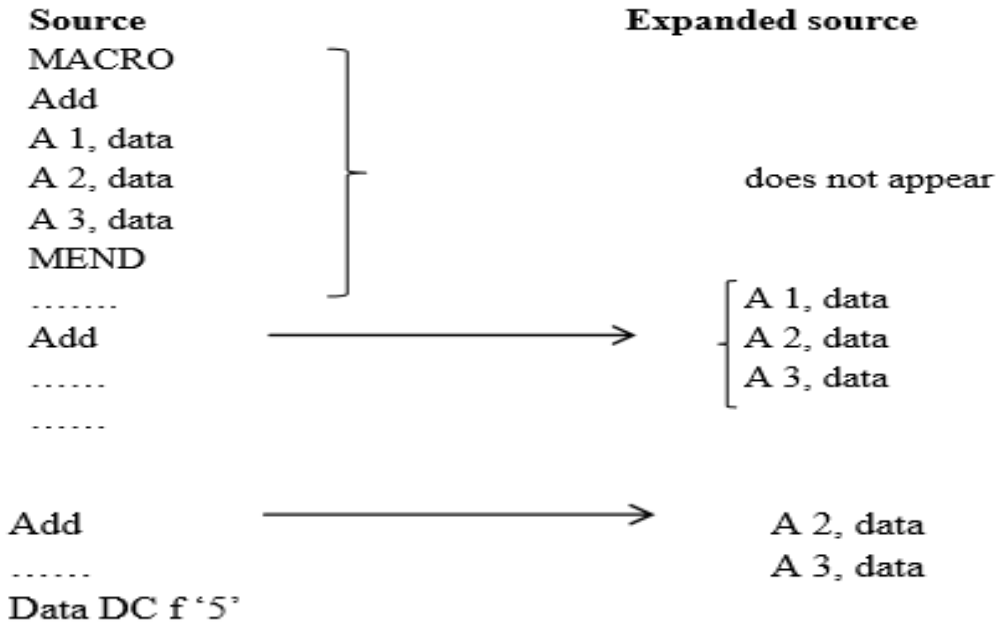
•Ex: In the above mentioned example 1 sequence will be repeated thrice, then we need to replace sequence by macro name . Here is the re-written form of example 1:

```
MACRO
Add
A 1, data
A 2, data
A 3, data
MEND
.....
Add
.....
Add
.....
Data DC f '5'
```

MACRO Expansion

•Whenever the program needs the instruction in the place of macro name then we need macro expansion.

•The macro-processor replace each macro calls with the defined set of instructions. This process of replacement is called macro expansion.



UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Features of macro facility

There are mainly 4 important feature of macro. Listed as follows:

1. Macro instruction argument
2. Conditional macro expansion
3. Macro calls within macros
4. Macro instruction defining macros

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro instruction argument

•As we have seen before, macro will be replaced by identical blocks. In such case no way for a specific macro call to modifying, coding etc. operations

•To overcome this problem we use macro instructions arguments where these arguments are appears in macro call. The corresponding macro dummy arguments are appears in macro definition.

• Ex:

```
A 1, data1
A 2, data2
……..
A 1, data1
A 2, data2
……..
```

•In the above example operations are the same with different parameter value.

• The first sequence performs an operation using data1 as operand.

•The second sequence performs an operation using data2 as operand.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Keyword argument

•The arguments which are presented in the macro definition known as Keyword argument or dummy argument.

• These arguments must be preceded by & symbol .

```
MACRO
Add &arg1, &arg2, &arg3
A 1, &arg1
A 2, &arg2
A 3, &arg3
MEND
```

Positional argument

- The arguments which are presented along with the macro call outside the definition referred as positional argument or actual argument.
- Positional and keyword argument must be match according to the number.

Ex: Add a, b, c

Where

a replaces the first keyword argument

b replaces the second keyword argument

c replaces the third keyword argument

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Conditional Macro Expansion

- The sequence of macro expansion can be reordered or change based on some conditions. There are 2 Important macro processor pseudo op. they are: i. AIF and ii. AGO
- AIF: It is a conditional branching pseudo op.
- The format of AIF is AIF<expression>.<sequencing label>
- Where expression is a relational expression, it involves strings, numbers etc.
- If the expression evaluates to true them the control transferred to the statements containing the sequencing label Otherwise, the control transferred to the next statement followed by AIF .
- AIF statement does not appears in the expanded source code.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Conditional Macro Expansion

- AGO: It is an unconditional branching pseudo op.
- The format of AGO is AGO.<Sequencing label>
- It is conditional transfer control to the statement containing sequencing label Each and every label must be starting with a .(dot) operator.
- AGO statement does not appear in the expanded source code.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Example of Conditional Macro Expansion

Ex:

```
Loop1    A 1,data1
         A 2, data2
         A 3, data3

Loop2    A 1,data3
         A 2, data2

Loop3    A 1, data1
```

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Example of Conditional Macro Expansion

In this example, the operands, labels and number of instructions generated are different in each sequence. Rewriting the set of instructions in a program might look like:

```
                :
                :
                MACRO
&PAR0   VARY            &COUNT,&PAR1,&PAR2,&PAR3
                A               1,&PAR1
                AIF             (&COUNT EQ 1).FINI          Test if &COUNT = 1
                A               2,&PAR2
                AIF             (&COUNT EQ 2).FINI          Test if &COUNT = 2
                ADD             3,&PAR3
.FINI           MEND
                :
                :
LOOP1   VARY            3,DATA1,DATA2,DATA3
                :
                :
LOOP2   VARY            2,DATA3,DATA2
                :
                :
DATA1   DC              F'5'
DATA2   DC              F'10'
DATA3   DC              F'15'
```

Expanded source

```
LOOP1   A       1,DATA1
        A       2,DATA2
        A       3,DATA3
LOOP2   A       1,DATA3
        A       2,DATA2
```

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Example of Conditional Macro Expansion

- The labels starting with a period (.) such as .FINI are macro labels.
- These macro labels do not appear in the output of the macro processor.
- The statement AIF (& COUNT EQ 1).FINI directs the macro processor to skip to the statement labelled .FINI, if the parameter corresponding to &COUNT is one.
- Otherwise, the macro processor continues with the statement that follows the AIF pseudo-op.

Example of Conditional Macro Expansion

- AIF pseudo-op performs an arithmetic test and since it is a conditional branch pseudo-op, it branches only if the tested condition is true.

- Another pseudo-op used in this program is AGO, which is an unconditional branch pseudo-op and works as a GOTO statement.

- This is the label in the macro instruction definition that specifies the sequential processing of instructions from the location where it appears in the instruction.

- These statements are indications or directives to the macro processor that do not appear in the macro expansions.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro calls within the macros (Nested Macros)

- Nested macro calls refer to the macro calls within the macros.
- A macro is available within other macro definitions also.
- In the scenario where a macro call occurs, which contains another macro call, the macro processor generates the nested macro definition as text and places it on the input stack.
- The definition of the macro is then scanned and the macro processor compiles it.
- This is important to note that the macro call is nested and not the macro definition.
- If you nest the macro definition, the macro processor compiles the same macro repeatedly, whenever the section of the outer macro is executed.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Example of Macro calls within the macros (Nested Macros)

The following example can make you understand the nested macro calls:

```
MACRO
SUB 1  &PAR
L      1, & PAR
A      1, = F'2'
ST     1, &PAR
MEND
MACRO
SUBST   &PAR1, &PAR2, &PAR3
SUB1   &PAR1
SUB1    &PAR2
SUB1    &PAR3
2MEND
```

You can easily notice from this example that the definition of the macro 'SUBST' contains three separate calls to a previously defined macro 'SUB1'. The definition of the macro SUB1 has shortened the length of the definition of the macro 'SUBST'. Although this technique makes the program easier to understand, at the same time, it is considered as an inefficient technique. This technique uses several macros that result in macro expansions on multiple levels. The following code describes how to implement a nested macro call:

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro calls within the macros (Nested Macros)

- Macro calls within macros can involve several levels.

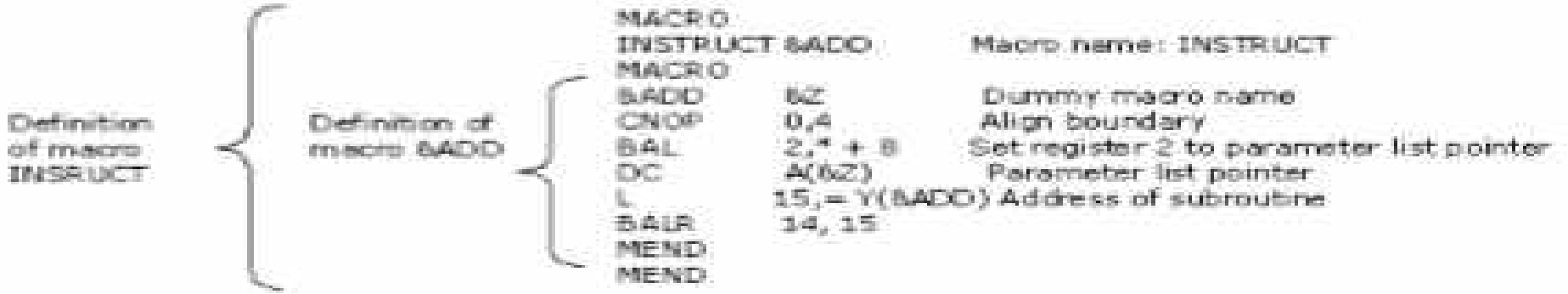
- This means a macro can include within itself any number of macros, which can further include macros. There is no limit while using macros in a program.

- The use of nested macro calls is beneficial until it causes an infinite loop.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Macro Instructions Defining Macros

The macro instructions can define macros. A single macro instruction can also simplify the process of defining a group of similar macros. The considerable idea while using macro instructions defining macros is that the inner macro definition should not be defined until the outer macro has been called once. This is explained in the following macro instruction.



In this code, first the macro INSTRUCT has been defined and then within INSTRUCT, a new macro &ADD is being defined. Macro definitions within macros are also known as "macro definitions within macro definitions".

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Bibliography

- https://www.youtube.com/watch?v=VG9VopzV_T0
- http://whatis.techtarget.com/definition/system-software
- http://searchdatacenter.techtarget.com/definition/assembler
- http://www.icse.s5.com/notes/m2.html

UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University

Thank You

UNIVERSITY INSTITUTE of COMPUTING

Asia's Fastest Growing University

System Software
CAT-204

Design By:

Prof. Pawandeep Sharma

A.P

Chandigarh University-Gharuan

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-I

Introduction to System Software: Machine Structure, evolution of operating system, machine language.

Assembler: Elements of Assembly Language Programming, General design procedure, design of a Two Pass Assemblers, A Single Pass Assemblers Design.

Table Processing: Searching & Sorting.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

UNIT-II

Macro and Macro Processors: Macro instructions, Features of a macro Facility: macro Instruction arguments, Conditional macro expansion, Macro calls within macros, Macro instruction defining macros, Advanced Macro Facilities, Implementation of simple macro processor, Two-pass algorithm, Implementation of macro calls within macros, Implementation within an assembler.

Linkers – Translated linked and load time addresses, relocation and linking concepts, Design of a linker, self relocating programs.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Syllabus

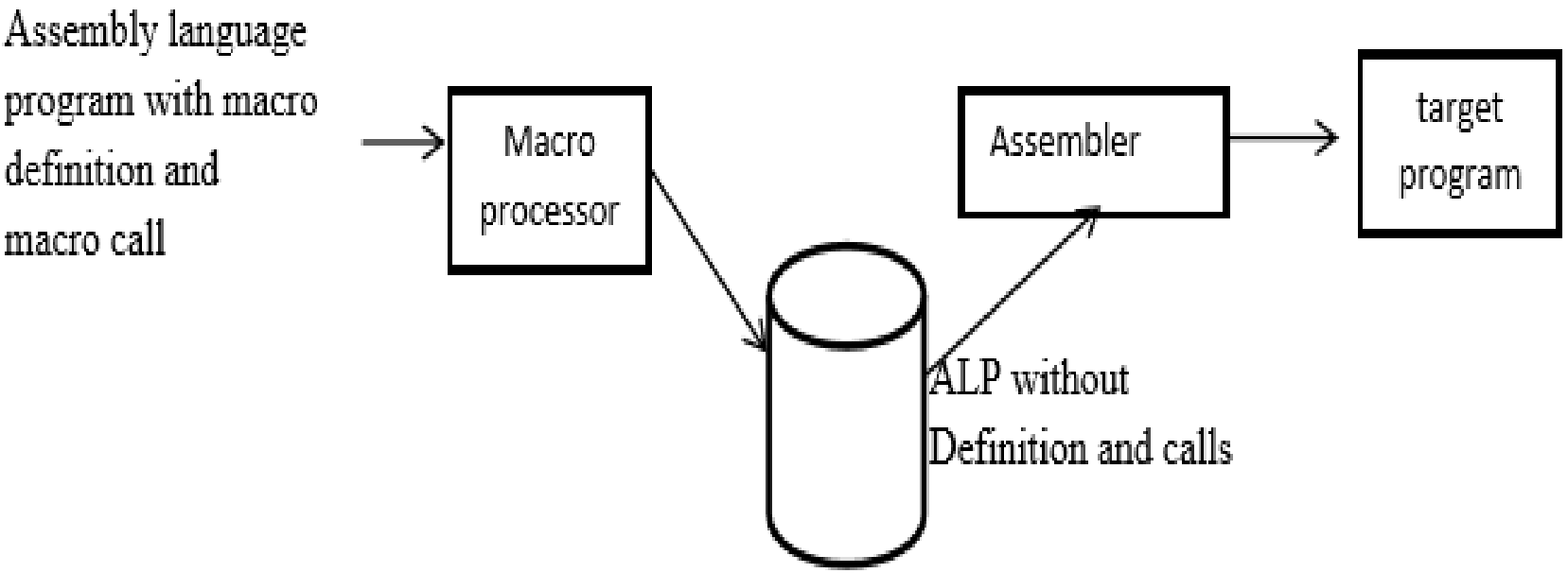
UNIT-III

Loaders: Loader scheme, absolute loaders, Subroutine linkages, Relocating loaders, Direct linking loaders, binders, linking loaders, overlays, Dynamic Binders, Design of an Absolute Loader, Design of a Direct-Linking Loader.
Compilers: Phases of Compiler Construction, Symbol Table, Top-down and bottom-up Parsing, Operator-Precedence Parsing, LR Parsers, Code Generation and Code Optimization, Memory management, Design & other issues.

Implementation of a macro instruction processor

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Implementation of a macro instruction processor



The macro process taken as input an ALP which contains macro definition and macro calls. Then it transforms to expanded source without consisting macro definition and macro calls is through the translator it will be convert as an object code.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Basic functions/task of a macro instruction processor

There are 4 basic functions performed by macro processor. They are:

1.Recognize macro definition: The macro processor must recognize macro definition by the MACRO and MEND pseudo op.

2.Save the definition: The macro processor must store the definition in memory which is required for expanding macro call.

3.Recognize macro call: The macro processor must organize macro names appears as operations mnemonics.

4.Expanded calls and substitute arguments: The macro substitute dummy/ macro definition arguments with the corresponding positional arguments in a macro call.

Database Specification

Pass1 : Processing macro-definition and calls

1. The input is a macro source code
2. The output is a macro code copy to pass2
3. MDT[macro definition table] which is used to store the body of the macro definition
4. MNT[macro name table], which is used to store names of the defined macro
5. MDTC[Macro Definition table Counter] which is used to indicate the next available entry in MDT
6. MNTC[Macro Name Table Counter] which is used to indicate the next available entry in MNT
7. ALA[Argument List array] to substitute index marker for the dummy argument before storing a macro definition

Database Specification

Pass2 : Processing Macro Expansions

1. Input is copy of the output macro source code from pass1
2. Output is expanded source code to be used as input to the assembler
3. MDT[macro definition table] created by pass1
4. MNT[macro name table], created by pass1
5. MDTP[Macro Definition Table Pointer] which is used to indicate the next line of text during expansion
6. ALA[Argument List array] is the reverse function of pass1 to substitute macro call arguments(positional arguments) for the index markers.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Arguments List array (ALA)

- It maintains the details about the parameters.
- It is used in both pass1 and pass2, but the functions are reverse in both the passes.
- ALA in pass1 In this when the macro definition are stored the arguments in the definition are replaced by index markers. # is the index marker, which is preceded by the dummy argument.

```
MACRO
Loop Add &arg1, &arg2, &arg3
A 1, &arg1
A 2, &arg2
A 3, &arg3
MEND
.....
.....
Add data1, data2, data3
```

```
MACRO
#0 Add &arg1, &arg2,&arg3
A 1, #1
A 2, #2
A 3, #3
MEND
```

In the above example loop is a label i.e., replaced by #0
Arg1 replaced by #1 like so on

Arguments List array (ALA)

- ALA in pass2 In this argument in the macro call are substituted for the index marker stored in macro definition In the above example the macro call is

- Loop Add data1, data2, data3

```
MACRO
Loop Add &arg1, &arg2, &arg3
A 1, &arg1
A 2, &arg2
A 3, &arg3
MEND
.....
.....
Add data1, data2, data3
```

```
MACRO
#0 Add &arg1, &arg2,&arg3
A 1, #1
A 2, #2
A 3, #3
MEND
```

In the above example loop is a label i.e., replaced by #0
Arg1 replaced by #1 like so on

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

MDT [Macro Definition table]

- It is used to store the body of macro definition
- The size of MDT is 80-bytes per entry
- It will be read every line in the definition except MACRO

```
MACRO
Loop Add &arg1, &arg2, &arg3
A 1, &arg1
A 2, &arg2
A 3, &arg3
MEND
.....
.....
Add data1, data2, data3
```

The macro definition table will be

Index	80-bytes per entry
10	#0 Add &arg1, &arg2,&arg3
11	A 1, #1
12	A 2, #2
13	A 3, #3
14	MEND
	.....

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

MNT [Macro Name table]

- It is used to store the names of the defined macros
- It has 3 fields, but the 2nd and 3rd field has the 12-bytes capacity
- The 2nd field is stored macro name it has 8 bytes and 3rd field MDT index for the macro name it has 4 bytes
- Total size of MNT is 12-bytes per entry

Pass1 Algorithm and Flowchart

Pass I Algorithm

Pass1 is used to store and processing macro definitions

1. Initialize or set MDTC as well as MNTC=1
2. Read the first line from the source code
3. A) If it is a macro pseudo op

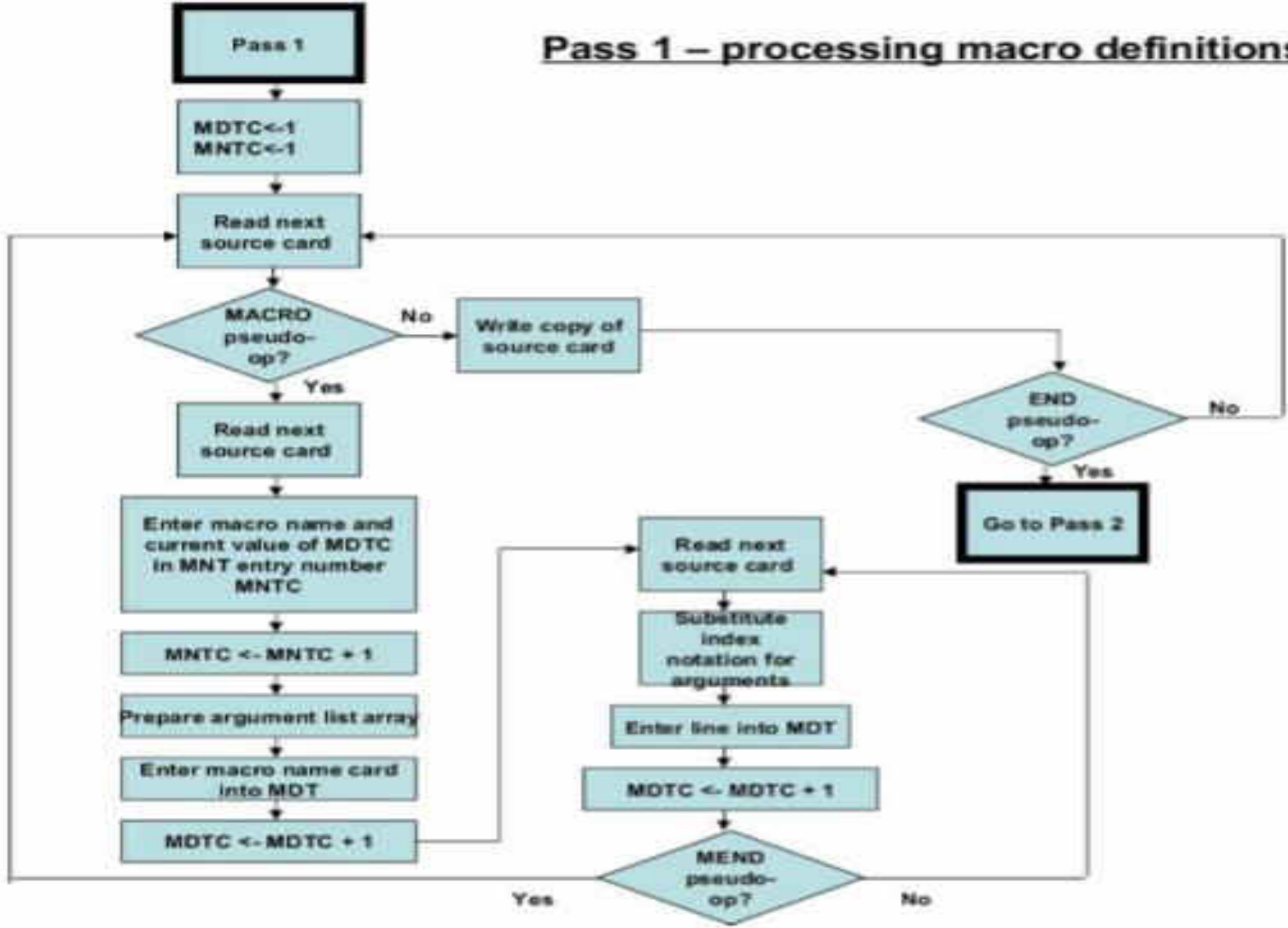
a. The entire macro definition except macro is stored in MDT
b. Read next line from source
c. Enter macro name and MDTC value stored in MNT Then increment MNTC value
d. Prepare ALA
e. Enter the macro name line in MDT then increment MDTC value
f. Read next line from source i.e., sequence then activate ALA in pass1
g. Substitute index marker for the dummy argument
h. Enter these values in MDT then increment MDTC value
i. Repeat these procedure until we get MEND pseudo op code

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Pass I Algorithm

4. If it is a MEND pseudo op code read the next line from source code otherwise the same procedure will be continue 3. B) If it is not a macro pseudo op code then copy the source code into pass2

Pass 1 – processing macro definitions



UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Pass II Algorithm and Flowchart

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Pass II Algorithm

Pass2 is used for specifying macro calls and expansion

1. Read next line from the source code copied by pass1
2. Search MNT for match with that code then

Case : a)

•Check whether you have encountered macro name then the MDT index of that macro name will be entered into MDTP

•Then activate ALA

• Increment MDTP value by one

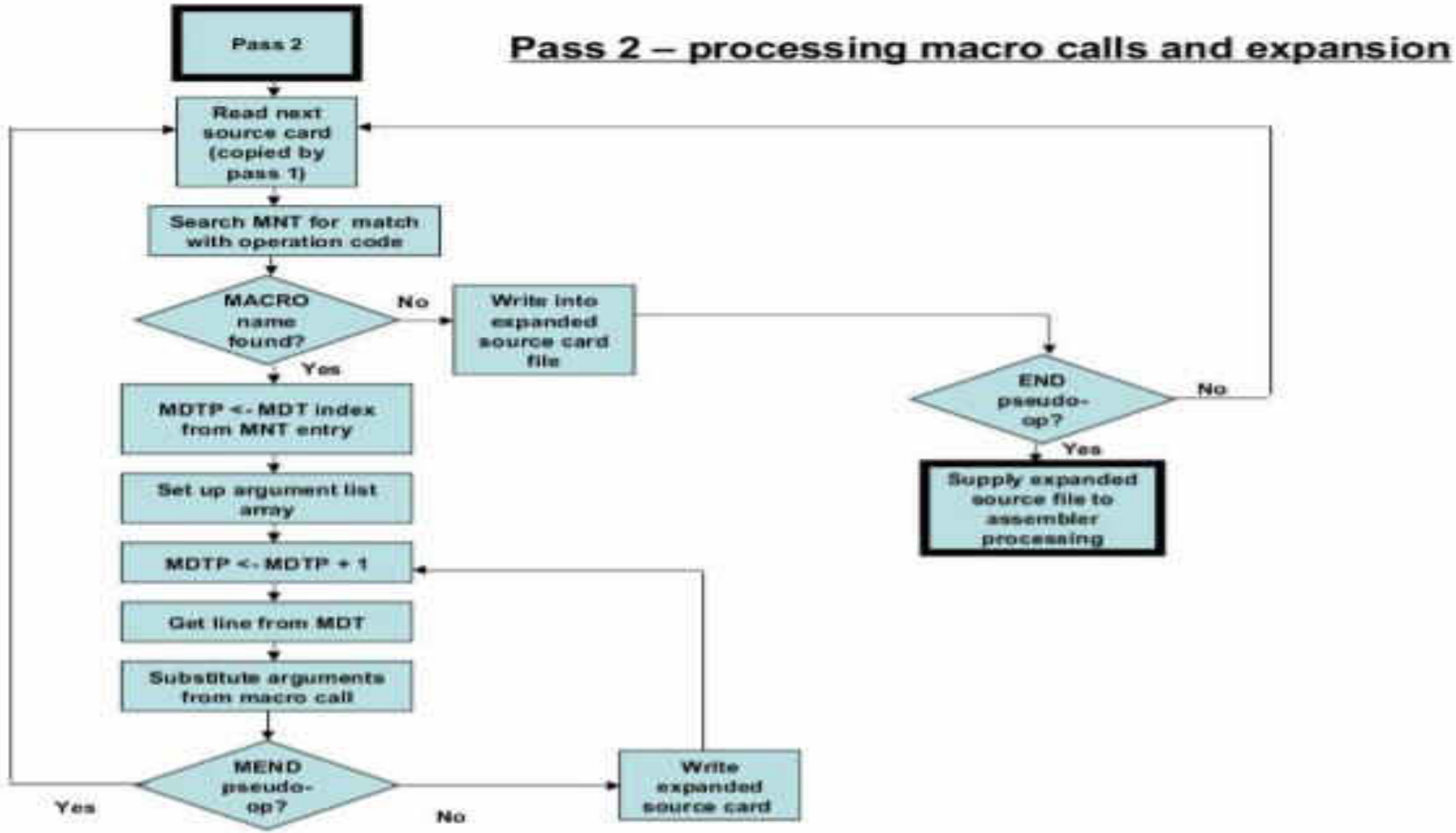
•Read next line from MDT and substitute arguments for macro call

•Repeat this process until we get MEND pseudo op

•If it is a MEND pseudo op then, read the next statement from the source otherwise write the expanded source code

Case : b) If it is not a macro name directly write into expanded source code

Pass2 Flowchart



UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Bibliography

- https://www.youtube.com/watch?v=VG9VopzV_T0
- http://whatis.techtarget.com/definition/system-software
- http://searchdatacenter.techtarget.com/definition/assembler
- http://www.icse.s5.com/notes/m2.html

UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University

Thank You