HSBPVT's PARIKRAMA GOI COE, KASHTI

SYSTEM PROGRAMMING & OPERATING SYSTEM [310251]

T.E. COMPUTER

(MACRO PROCESSOR)

Prof. SURYAVANSHI ARTI PRASHANT

E-mail:- artips15@gmail.com

Prof. SURYAVANSHI PRASHANT MAHARUDRA

E-mail:- sprashant1234@gmail.com

HSBPVT'S, PARIKRAMA GOI COE KASHTI.

CONTENT

- Introduction
- Basic macro processor functions
- Macro expansion
- Macro invocation
- Pass-1 macro processor Flowchart
- Pass-2 macro processor Flowchart
- Data structures for one-pass macro processor
- Algorithm of Macro Processor
- Comparison of Macro Processors Design

INTRODUCTION

- A macro instruction (macro) is a notational convenience for the programmer
- It allows the programmer to write shorthand version of a program(module programming)
- The macro processor replaces each macro instruction with the corresponding group of source language statements (*expanding*)
- Normally, it performs no analysis of the text it handles.
- It does not concern the meaning of the involved statements during macro expansion.
- The design of a macro processor generally is *machine independent!*

BASIC MACRO PROCESSOR FUNCTIONS

- Two new assembler directives are used in macro definition
- **MACRO:** identify the beginning of a macro definition
- **MEND:** identify the end of a macro definition
- Prototype for the macro Each parameter begins with '&'

```
name MACRO parameters
    :
    body
    :
    MEND
```

Body: the statements that will be generated as the expansion of the macro.

MACRO EXPANSION

- Each macro invocation statement will be expanded into the statements that form the body of the macro.
- Arguments from the macro invocation are substituted for the parameters in the macro prototype (according to their positions).
  - In the definition of macro: Parameter
  - In the macro invocation: Argument
- Comment lines within the macro body will be deleted.
- Macro invocation statement itself has been included as a comment line.
- The label on the macro invocation statement has beenretained as a label on the first statement generated in the macro expansion.

MACRO EXPANSION

Source	*Expanded source*
M1 MACRO &D1, &D2 STA &D1 STB &D2 MEND . M1 DATA1, DATA2 . M1 DATA4, DATA3	. . . STA DATA1 STB DATA2 . STA DATA4 STB DATA3 .

MACRO INVOCATION

- A macro invocation statement (a macro call) gives the name of the macro instruction being invoked and the arguments to be used in expanding the macro.

  macro_name p1, p2, …

- Difference between macro call and procedure call:

Macro call: statements of the macro body are expanded each time the macro is invoked.

Procedure call: statements of the subroutine appear only one, Regardless of how many times the subroutine is called.

EXCHANGE THE VALUES OF TWO VARIABLES

```
void exchange(int a, int b) {
int temp;
temp = a;
a = b;
b = temp;
}
main() {
int i=1, j=3;
printf("BEFORE - %d %d\n", i, j);
exchange(i, j);
printf("AFTER - %d %d\n", i, j);
}
```

PASS BY REFERENCE

```
void exchange(int *p1, int *p2) {
int temp;
temp = *p1;
*p1 = *p2;
*p2 = temp;
}
main() {
int i=1, j=3;
printf("BEFORE - %d %d\n", i, j);
exchange(&i, &j);
printf("AFTER - %d %d\n", i, j);
}
```

ASSEMBLY CODE

```
. Subroutine EXCH
EXCH LDA @P1
STA TEMP
LDA @P2
STA @P1
LDA TEMP
STA @P2
RSUB
P1 RESW 1
P2 RESW 1
TEMP RESW 1
```

```
MAIN LDA #1
STA I
LDA #3
STA J
. Call a subroutine
LDA #I
STA P1
LDA #J
STA P2
JSUB EXCH
I RESW 1
J RESW 1
END MAIN
```

NO LABEL IN THE MACRO BODY

- Problem of the label in the body of macro:
- If the same macro is expanded multiple times at different places in the program…
- There will be *duplicate labels*, which will be treated as errors by the assembler.
- Solutions:

  Do not use labels in the body of macro.

  Explicitly use PC-relative addressing instead.

  Ex, in RDBUFF and WRBUFF macros,

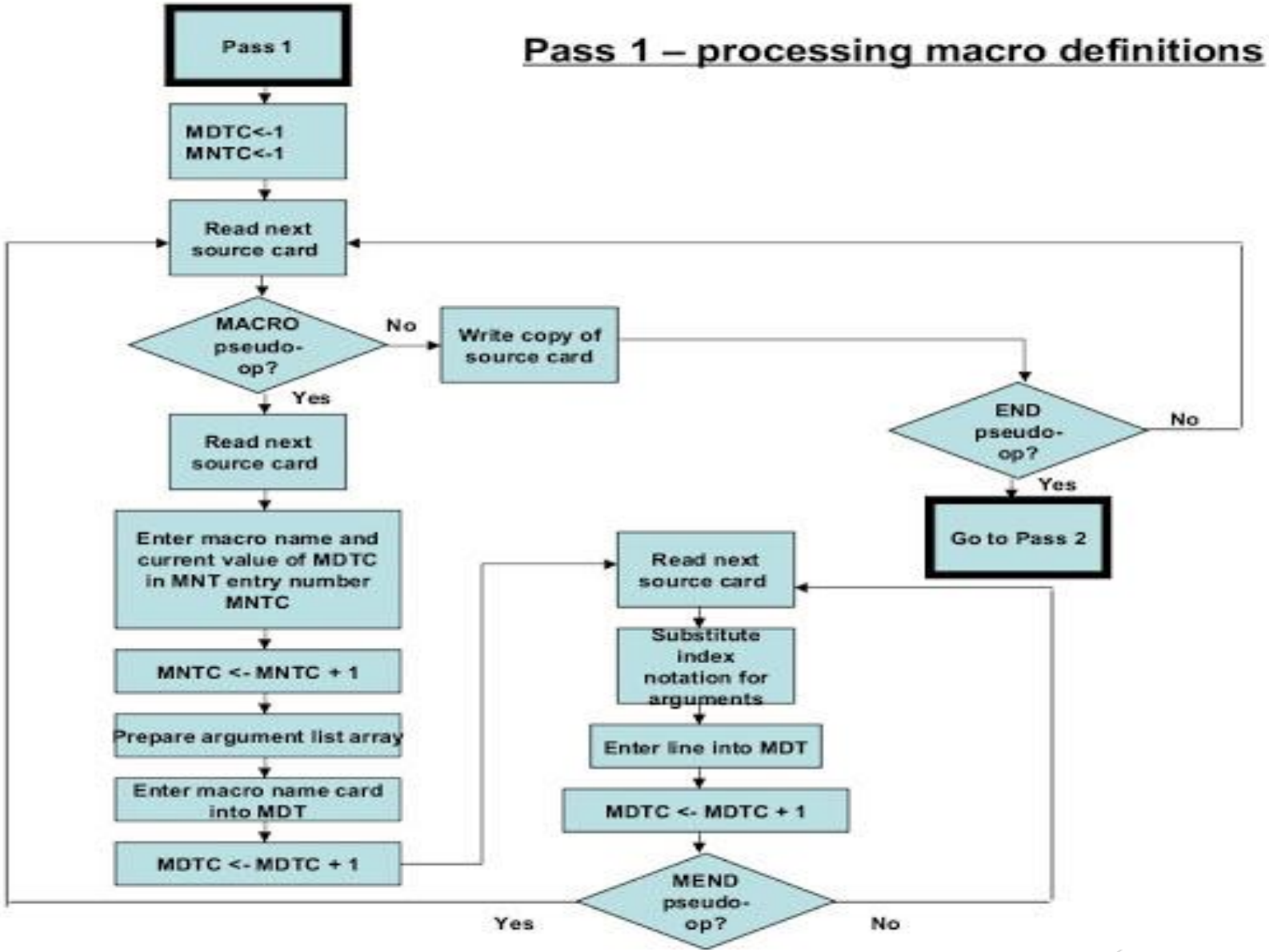
  ```
  JEQ *+11
  JLT *-14
  ```

- It is inconvenient and error-prone.

ONE-PASS MACRO PROCESSOR

- A one-pass macro processor that alternate between *macro definition* and *macro expansion* in a recursive way is able to handle recursive macro definition.
- Restriction:
  - The definition of a macro must appear in the source program before any statements that invoke that macro.
  - This restriction does not create any real inconvenience

FLOWCHART OF PASS-1 MACRO PROCESSOR



TWO-PASS MACRO PROCESSOR

- You may design a two-pass macro processor

  Pass 1:

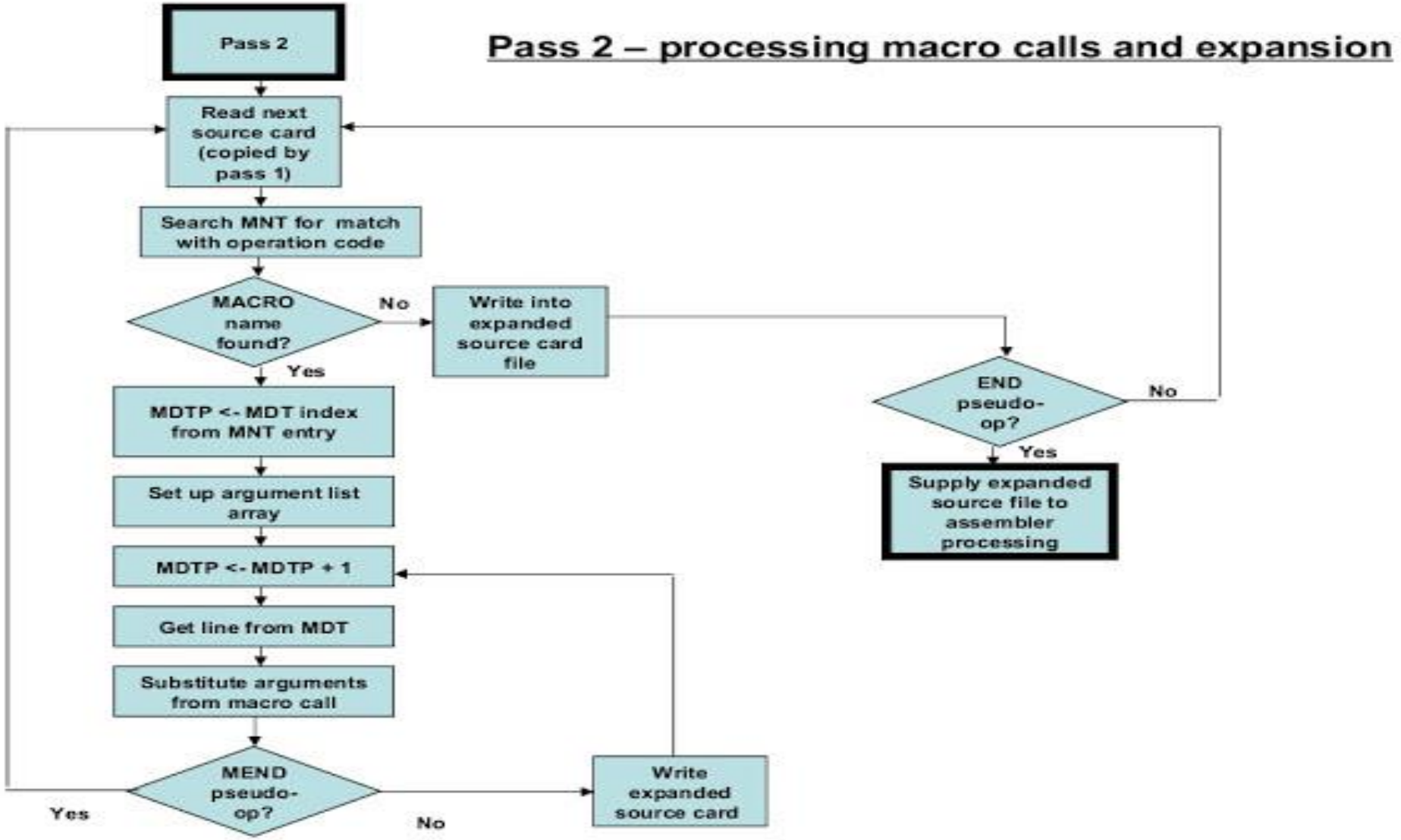
  Process all macro definitions

  Pass 2:

  Expand all macro invocation statements

- However, one-pass may be enough
- Because all macros would have to be defined during the first pass before any macro invocations were expanded.
- The definition of a macro must appear before any statements that invoke that macro.
- Moreover, the body of one macro can contain definitions of other macros.

FLOWCHART OF PASS-2 MACRO PROCESSOR

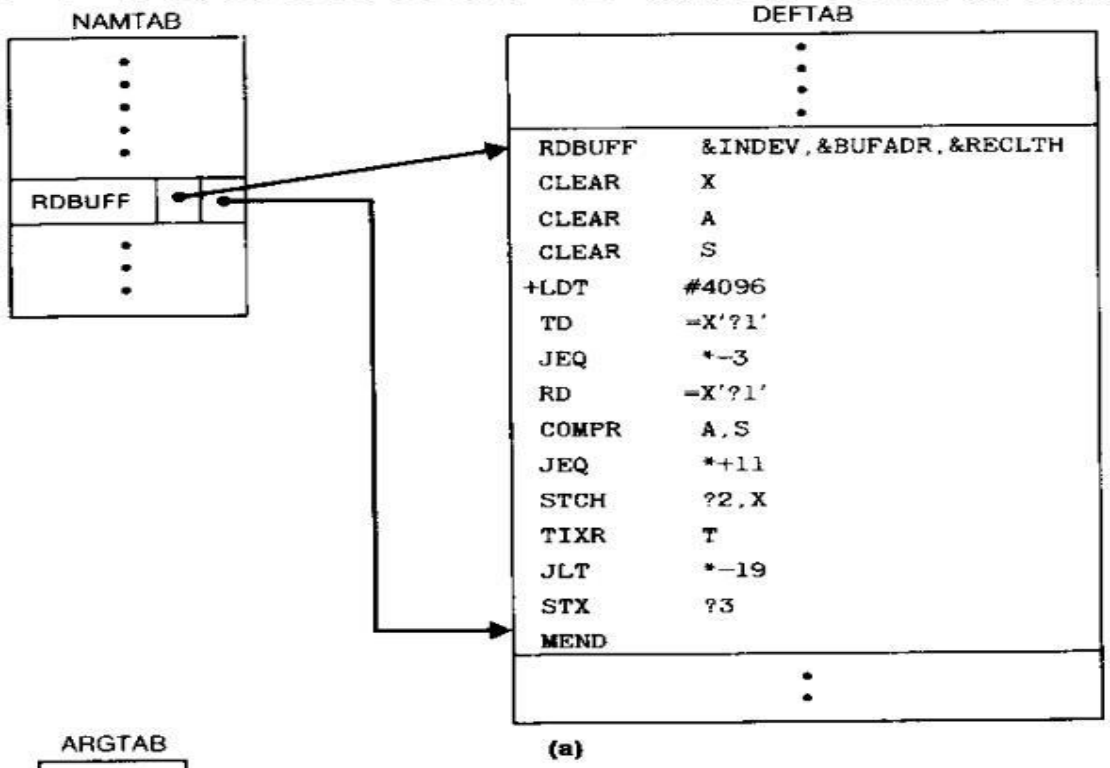


DATA STRUCTURES FOR ONE-PASS MACRO PROCESSOR

DEFTAB (definition table)	NAMTAB	ARGTAB
❑ Stores the macro definition including *macro prototype* and *macro body* ❑ Comment lines are omitted. ❑ References to the macro instruction parameters are converted to a positional notation for efficiency in substituting arguments.	❑ Stores macro names Serves as an index to DEFTAB. ❑ The begin Pointers to begining and the end of the macro definition (DEFTAB)	❑ Stores the arguments of macro invocation according to their positions in the argument list ❑ As the macro is expanded, arguments from ARGTAB are substituted for the corresponding parameters in the macro body.

DATA STRUCTURES

Macro Processor Data Structures



(a)

(b)

ALGORITHM

MAIN program

- Iterations of
- GETLINE
- PROCESSLINE

Procedure GETLINE

If EXPANDING **then**
get the next line to be processed from DEFTAB
Else
read next line from input file

Procedure EXPAND

Set up the argument values in ARGTAB
Expand a macro invocation statement (like in MAIN procedure)
- Iterations of
- GETLINE
- PROCESSLINE

Procedure PROCESSLINE

- DEFINE
- EXPAND
- Output source line

Procedure DEFINE

Make appropriate entries in DEFTAB and NAMTAB

COMPARISON OF MACRO PROCESSORS DESIGN

One-pass algorithm	Two-pass algorithm
➢ Every macro must be defined before it is called ➢ One-pass processor can alternate between macro definition and macro expansion ➢ Nested macro definitions are allowed but nested calls are not	➢ Pass1: Recognize macro definitions ➢ Pass2: Recognize macro calls Nested macro definitions are not allowed

THANK YOU

Prof. SURYAVANSHI ARTI PRASHANT
E-mail:- artips15@gmail.com
Prof. SURYAVANSHI PRASHANT MAHARUDRA
E-mail:- sprashant1234@gmail.com
HSBPVT's PARIKRAMA GOI COE KASHTI