KCG College of Technology,Chennai-96
Computer Science and Engineering

CS2304 SYSTEM SOFTWARE III Sem CSE

QUESTION BANK - UNIT-III MACRO PROCESSORS

UNIT IV MACRO PROCESSORS **9**

Basic macro processor functions - Macro Definition and Expansion – Macro Processor Algorithm and data structures - Machine-independent macro processor features - Concatenation of Macro Parameters – Generation of Unique Labels – Conditional Macro Expansion – Keyword Macro Parameters-Macro within Macro-Implementation example - MASM Macro Processor – ANSI C Macro language.

1) Define Macro.

- A *macro instruction* (*macro*) is a notational convenience for the programmer.
- It allows the programmer to write a shorthand version of a program
  - A *macro* represents a commonly used group of statements in the source programming language.
  - *Expanding* the macros - the macro processor replaces each macro instruction with the corresponding group of source language statements.

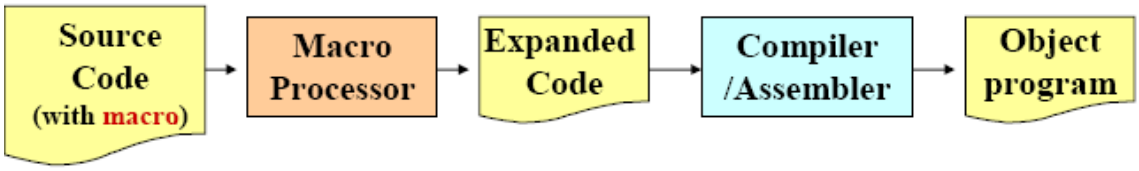
2) What is a macro processor?

A macro processor - Essentially involve the substitution of one group of characters or lines for another. Normally, it performs no analysis of the text it handles. It doesn't concern the meaning of the involved statements during macro expansion

o The design of a macro processor generally is machine independent.

Three examples of actual macro processors:

- A macro processor designed for use by assembler language programmers
- Used with a high-level programming language
- General-purpose macro processor, which is not tied to any particular language



3) What are the basic macro processor functions?

Basic Macro Processors Functions

Macro processor should processes the

Macro definitions

o Define macro name, group of instructions

Macro invocation (macro calls)

o A body is simply copied or substituted at the point of call Expansion with substitution of parameters
o ***Arguments*** are textually substituted for the ***parameters***
o The resulting procedure body is textually substituted for the call

4) What are the Assembler directives used for Macro Definition?

Macro Definition

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

o label op operands

name MACRO parameters

:

body

:

MEND

o Parameters: the entries in the operand field identify the parameters of the macro instruction

We require each parameter begins with '&'

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

o Prototype for the macro:

The *macro name* and *parameters* define a pattern or *prototype* for the macro instructions used by the programmer

5) Give an example program which uses Macro Definition.

Macro Definition

```
 5      COPY      START   0                 COPY FILE FROM INPUT TO OUTPUT
10      RDBUFF    MACRO   &INDEV,&BUFADR,&RECLTH
15      .
20      .         MACRO TO READ RECORD INTO BUFFER
25      .
30                CLEAR   X                 CLEAR LOOP COUNTER
35                CLEAR   A
40                CLEAR   S
45               +LDT     #4096            SET MAXIMUM RECORD LENGTH
50                TD      =X'&INDEV'       TEST INPUT DEVICE
55                JEQ      *-3             LOOP UNTIL READY
60                RD      =X'&INDEV'       READ CHARACTER INTO REG A
65                COMPR    A,S             TEST FOR END OF RECORD
70                JEQ      *+11            EXIT LOOP IF EOR
75                STCH     &BUFADR,X       STORE CHARACTER IN BUFFER
80                TIXR     T               LOOP UNLESS MAXIMUM LENGTH
85                JLT      *-19              HAS BEEN REACHED
90                STX      &RECLTH         SAVE RECORD LENGTH
95                MEND
```

```
 5      COPY      START   0                 COPY FILE FROM INPUT TO OUTPUT
10      RDBUFF    MACRO   &INDEV,&BUFADR,&RECLTH
15      .
20      .         MACRO TO READ RECORD INTO BUFFER
25      .
30                CLEAR   X                 CLEAR LOOP COUNTER
35                CLEAR   A
40                CLEAR   S
45               +LDT     #4096            SET MAXIMUM RECORD LENGTH
50                TD      =X'&INDEV'       TEST INPUT DEVICE
55                JEQ     *-3              LOOP UNTIL READY
60                RD      =X'&INDEV'       READ CHARACTER INTO REG A
65                COMPR   A,S              TEST FOR END OF RECORD
70                JEQ     *+11             EXIT LOOP IF EOR
75                STCH    &BUFADR,X        STORE CHARACTER IN BUFFER
80                TIXR    T                LOOP UNLESS MAXIMUM LENGTH
85                JLT     *-19               HAS BEEN REACHED
90                STX     &RECLTH          SAVE RECORD LENGTH
95                MEND
```

6) How Macro is invoked in a program? Give example.

Macro Invocation

```
165     .
170     .         MAIN PROGRAM
175     .
180     FIRST     STL     RETADR           SAVE RETURN ADDRESS
190     CLOOP     RDBUFF  F1,BUFFER,LENGTH  READ RECORD INTO BUFFER
195               LDA     LENGTH           TEST FOR END OF FILE
200               COMP    #0
205               JEQ     ENDFIL           EXIT IF EOF FOUND
210               WRBUFF  05,BUFFER,LENGTH  WRITE OUTPUT RECORD
215               J       CLOOP            LOOP
220     ENDFIL    WRBUFF  05,EOF,THREE     INSERT EOF MARKER
225               J       @RETADR
230     EOF       BYTE    C'EOF'
235     THREE     WORD    3
240     RETADR    RESW    1
245     LENGTH    RESW    1                LENGTH OF RECORD
250     BUFFER    RESB    4096             4096-BYTE BUFFER AREA
255               END     FIRST
```

Figure 4.1 Use of macros in a SIC/XE program.

7) What is Macro Expansion? Give an example.

Macro Expansion

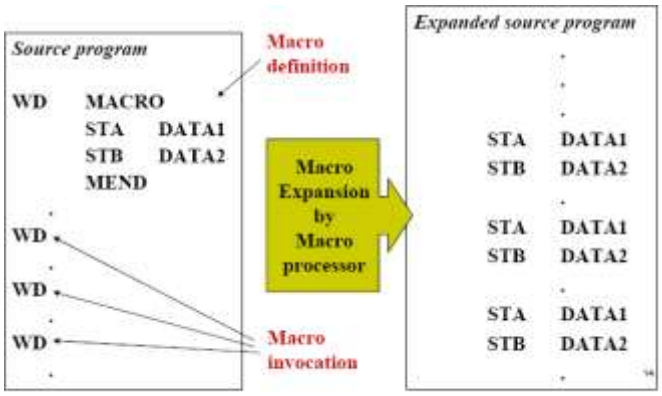
Each **macro invocation** statement will be **expanded** into the statements that form the **body** of the macro.

o **Arguments** from the macro invocation are **substituted** for the **parameters** in the **macro prototype**.

The **arguments and parameters** are associated with one another according to their **positions.**

o The **first argument** in the **macro invocation** corresponds to **the first parameter** in the **macro prototype**, etc.

Macro Expansion Example



Program From Fig. 4.1 with Macros Expanded (fig. 4.2)(Cont.)

```
215                 J       CLOOP           LOOP
220     .ENDFIL     WRBUFF  05,EOF,THREE     INSERT EOF MARKER
220a    ENDFIL      CLEAR   X               CLEAR LOOP COUNTER
220b                LDT     THREE
220c                LDCH    EOF,X           GET CHARACTER FROM BUFFER
220d                TD      =X'05'          TEST OUTPUT DEVICE
220e                JEQ     *-3             LOOP UNTIL READY
220f                WD      =X'05'          WRITE CHARACTER
220g                TIXR    T               LOOP UNTIL ALL CHARACTERS
220h                JLT     *-14              HAVE BEEN WRITTEN
225                 J       @RETADR
230     EOF         BYTE    C'EOF'
235     THREE       WORD    3
240     RETADR      RESW    1
245     LENGTH      RESW    1               LENGTH OF RECORD
250     BUFFER      RESB    4096            4096-BYTE BUFFER AREA
255                 END     FIRST
```

Figure 4.2 Program from Fig. 4.1 with macros expanded.

8)Explain the functions of two pass macro processor.

Two-pass macro processor

- **Two-pass macro processor**
  - Pass1: process all *macro definitions*
  - Pass2: expand all *macro invocation* statements
- **Problem**
  - Does not allow *nested macro definitions*
- **Nested macro definitions**
- The body of a macro contains definitions of other macros
  - Because all macros would *have to be defined during the*
  - *first pass* before any macro invocations were expanded
- **Solution**
  - One-pass macro processor

9) **What is the characteristic of one pass macro processor?**

One-pass macro processor

Every macro must be defined before it is called

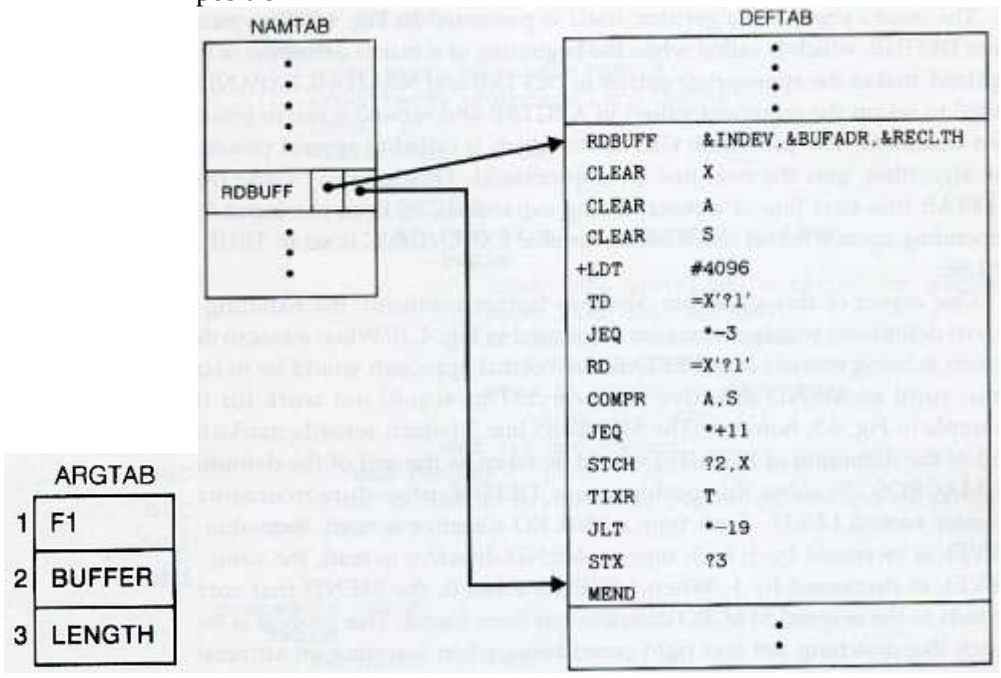
One-pass processor can alternate between ***macrodefinition*** and ***macro expansion***

Nested macro definitions are allowed

10) What are the data structures used by the macro processor?

➢ Data Structures

- **MACRO DEFINITION TABLE (DEFTAB)**
  - Macro prototype
  - States that make up the macro body
  - Reference to parameters are converted to a positional notation.
- **MACRO NAME TABLE (NAMTAB)**
  - Role: an index to DEFTAB
  - Pointers to the beginning and end of the definition in DEFTAB
- **MACRO ARGUMENT TABLE (ARGTAB)**
  - Used during the expansion
  - Invocation \ Arguments are stored in ARGTAB according to their position



11) Write an algorithm for the Macro processor and explain.

Algorithm and Data Structure (4)

```
begin {macro processor}
    EXPANDING := FALSE
    while OPCODE ≠ 'END' do
        begin
            GETLINE
            PROCESSLINE
        end {while}
end {macro processor}
```

```
procedure GETLINE
    begin
        if EXPANDING then
            begin
                get next line of macro definition from DEFTAB
                substitute arguments from ARGTAB for positional notation
            end {if}
        else
            read next line from input file
    end {GETLINE}
procedure PROCESSLINE
    begin
        search NAMTAB for OPCODE
        if found then
            EXPAND
        else if OPCODE = 'MACRO' then
            DEFINE
        else write source line to expanded file
    end {PROCESSLINE}
```

```
procedure EXPAND
    begin
        EXPANDING  := TRUE
        get first line of macro definition {prototype} from DEFTAB
        set up arguments from macro invocation in ARGTAB
        write macro invocation to expanded file as a comment
        while not end of macro definition do
            begin
                GETLINE
                PROCESSLINE
            end {while}
        EXPANDING := FALSE
    end {EXPAND}
procedure DEFINE
    begin
        enter macro name into NAMTAB
        enter macro prototype into DEFTAB
        LEVEL  := 1
        while LEVEL > 0 do
            begin
                GETLINE
                if this is not a comment line then
                    begin
                        substitute positional notation for parameters
                        enter line into DEFTAB
                        if OPCODE = 'MACRO' then
                            LEVEL := LEVEL + 1
                        else if OPCODE = 'MEND' then
                            LEVEL  := LEVEL - 1
                    end {if not comment}
            end {while}
        store in NAMTAB pointers to beginning and end of definition
    end {DEFINE}
```

12) What is the difference between a Macro and a subroutine?

Differences between Macro and Subroutine

- After macro processing, the expanded file can be used as input to the assembler.
- The statements generated from the macro expansions will be assembled exactly as though they had been written directly by the programmer.
- The **differences** between **macro invocation and subroutine call**
    - The statements that form the body of the macro are generated each time a macro is expanded.
    - Statements in a subroutine appear only once, regardless of how many times the subroutine is called.

13) **What are the machine independent features of macro processors?**

Other Macro Features

- Concatenation of macro parameters
- Generation of unique labels
- Conditional macro expansion
- Keyword Macro Parameters

Concatenation of Macro Parameters

- Pre-concatenation
  - LDA X&ID1
- Post-concatenation
  - LDA X&ID→1
- Example: Figure 4.6

Concatenation Example

```
1  SUM     MACRO   &ID
2          LDA     X&ID→1
3          ADD     X&ID→2
4          ADD     X&ID→3
5          STA     X&ID→S
6          MEND
```

```
SUM     A
        ↓
LDA     XA1
ADD     XA2
ADD     XA3
STA     XAS
```

```
SUM     BETA
        ↓
LDA     XBETA1
ADD     XBETA2
ADD     XBETA3
STA     XBETAS
```

Generation of Unique Labels

- Example
  - JEQ *-3
  - inconvenient, error-prone, difficult to read
- Example Figure 4.7
  - ✓ $LOOP TD =X'&INDEV'
  - 1st call:
    - ✓ $AALOOP TD =X'F1'

- 2nd call:
  - ✓ $ABLOOP TD =X'F1'

```
25    RDBUFF   MACRO   &INDEV,&BUFADR,&RECLTH
30             CLEAR   X             CLEAR LOOP COUNTER
35             CLEAR   A
40             CLEAR   S
45            +LDT     #4096         SET MAXIMUM RECORD LENGTH
50    $LOOP    TD      =X'&INDEV'    TEST INPUT DEVICE
55             JEQ      $LOOP        LOOP UNTIL READY
60             RD      =X'&INDEV'    READ CHARACTER INTO REG A
65             COMPR    A,S          TEST FOR END OF RECORD
70             JEQ      $EXIT        EXIT LOOP IF EOR
75             STCH     &BUFADR,X    STORE CHARACTER IN BUFFER
80             TIXR     T            LOOP UNLESS MAXIMUM LENGTH
85             JLT      $LOOP          HAS BEEN REACHED
90    $EXIT    STX      &RECLTH      SAVE RECORD LENGTH
95             MEND
```

(a)

RDBUFF F1, BUFFER, LENGTH

```
30             CLEAR   X             CLEAR LOOP COUNTER
35             CLEAR   A
40             CLEAR   S
45            +LDT     #4096         SET MAXIMUM RECORD LENGTH
50    $AALOOP  TD      =X'F1'        TEST INPUT DEVICE
55             JEQ      $AALOOP      LOOP UNTIL READY
60             RD      =X'F1'        READ CHARACTER INTO REG A
65             COMPR    A,S          TEST FOR END OF RECORD
70             JEQ      $AAEXIT      EXIT LOOP IF EOR
75             STCH     BUFFER,X     STORE CHARACTER IN BUFFER
80             TIXR     T            LOOP UNLESS MAXIMUM LENGTH
85             JLT      $AALOOP        HAS BEEN REACHED
90    $AAEXIT  STX      LENGTH       SAVE RECORD LENGTH
```

(b)