Sri Vidya College of Engineering and Technology
Virudhunagar – 626 005

Department of Computer Science and Engineering

Class:	**III CSE, V Semester**
Subject Code:	**CS 2304**
Subject:	**System Software**
Prepared by	**Kaviya P**

Unit IV

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.

Textbook:

T1. Leland L. Beck, “System Software – An Introduction to Systems Programming”, 3rd Edition, Pearson Education Asia, 2006.

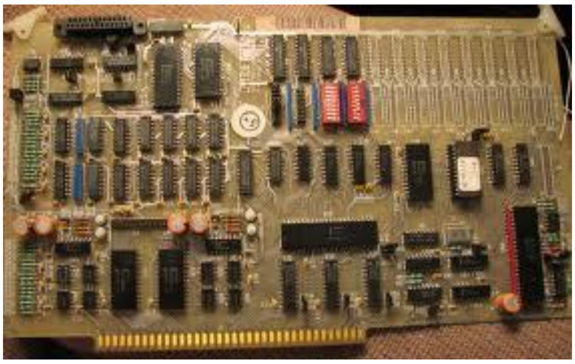
Lesson Plan 1	Basic Macro Processor Functions – Macro Definition And Expansion
Lesson Plan 2	Macro Processor Algorithm And Data Structures
Lesson Plan 3	Machine Independent Macro Processor Features – Concatenation Of Macro Parameters, Generation Of Unique Labels
Lesson Plan 4 & 5	Conditional Macro Expansion
Lesson Plan 6	Keyword Macro Parameters
Lesson Plan 7 &8	Macro Within Macro
Lesson Plan 9	Implementation Example – MASM

Staff in-charge HOD-CSE

	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**	
Class	III CSE	
Subject Code	CS2304	
Subject	System Software	
Prepared By	Kaviya.P	
Lesson Plan for	Basic Macro Processor Functions – Macro Definition And Expansion	
Time:	50 Minutes	
Lesson. No	1/9	

1. **Topics to be covered**
   - Basic Macro Processor Functions
   - Macro Definition and Expansion
2. **Skills addressed**:

   Listening
3. **Objectives of this lesson plan:**
   - To enable students to understand basic macro processor functions, macro definition and expansion
4. **Outcome (s):**
   - Able to understand basic Macro Processor Functions.
   - Able to explain Macro Definition.
   - Able to explain Macro Expansion.
5. **Link sheet:**
   - Define Macro.
6. **Evocation:**



7. **Lecture notes (attached)**

8. **Text Book**
   - Leland L. Beck, "System Software – An Introduction to Systems Programming",

     3rd Edition, Pearson Education Asia, 2006. PP 181 - 186.
9. **Application**
   - Program Development

Introduction

- A macro represents a commonly used group of statements in the source programming language
- The macro processor replaces each macro instruction with the corresponding group of source language statement, this is called expanding macros
- The functions of a macro processor essentially involve the substitution of one group of characters or lines for another

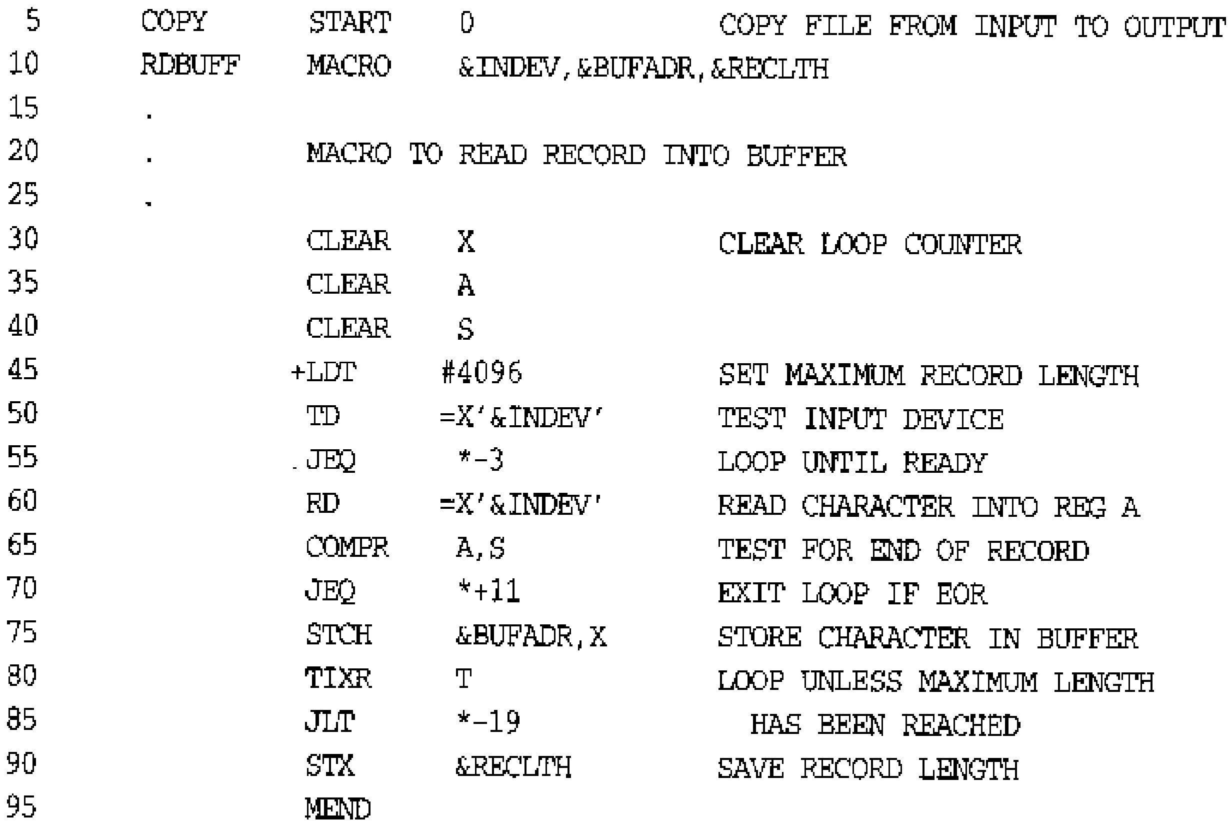
Basic Macro Processor Functions

- Macro Definition and Expansion
- Macro Processor Algorithms and Data structures

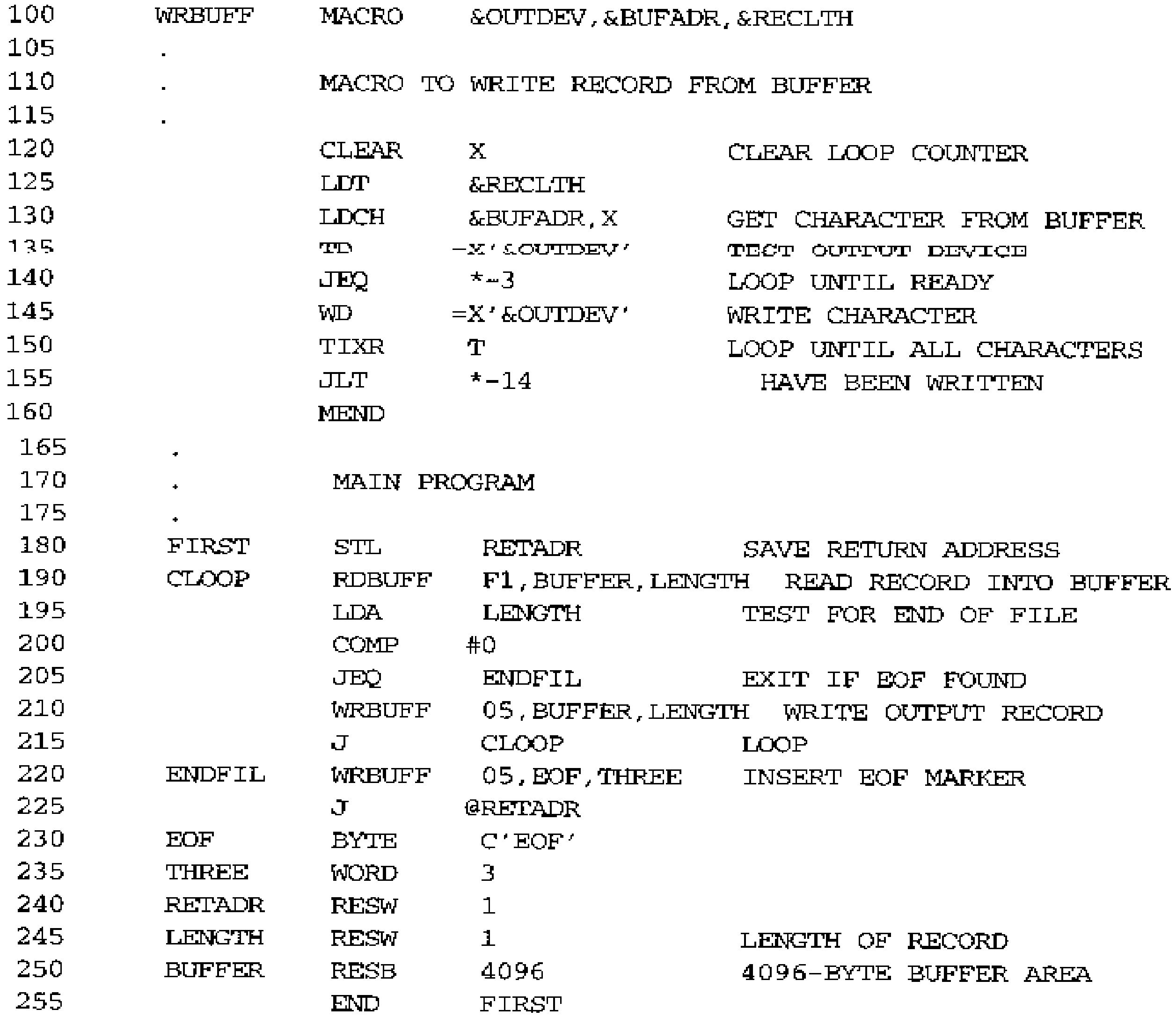
Macro Definition and Expansion

- The MACRO statement identifies the beginning of a macro definition
- The symbol in the label field is the name of the instruction
- The entries in the operand field identify the parameter of the macro instruction
- Each parameter begins with the character &
- The MEND assembler directive marks the end of the macro definition
- A macro invocation statement gives the name of the macro instruction being invoked and the arguments to be used in expanding the macro

Use of macros in a SIC/XE

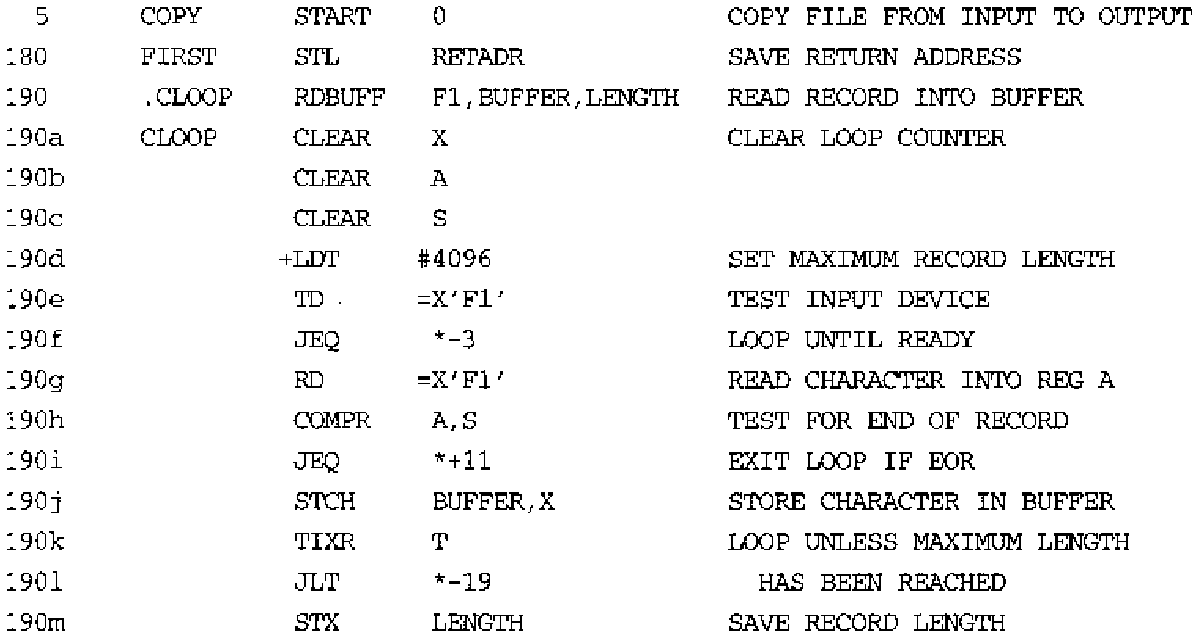


```
 5    COPY      START   0              COPY FILE FROM INPUT TO OUTPUT
10    RDBUFF    MACRO   &INDEV,&BUFADR,&RECLTH
15    .
20    .         MACRO TO READ RECORD INTO BUFFER
25    .
30              CLEAR   X              CLEAR LOOP COUNTER
35              CLEAR   A
40              CLEAR   S
45             +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
```



```
100     WRBUFF    MACRO     &OUTDEV,&BUFADR,&RECLTH
105     .
110     .         MACRO TO WRITE RECORD FROM BUFFER
115     .
120               CLEAR     X               CLEAR LOOP COUNTER
125               LDT       &RECLTH
130               LDCH      &BUFADR,X       GET CHARACTER FROM BUFFER
135               TD       =X'&OUTDEV'      TEST OUTPUT DEVICE
140               JEQ       *-3             LOOP UNTIL READY
145               WD       =X'&OUTDEV'      WRITE CHARACTER
150               TIXR      T               LOOP UNTIL ALL CHARACTERS
155               JLT       *-14              HAVE BEEN WRITTEN
160               MEND
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
```

Program with Macro Expanded



```
  5     COPY      START     0                   COPY FILE FROM INPUT TO OUTPUT
180     FIRST     STL       RETADR              SAVE RETURN ADDRESS
190     .CLOOP    RDBUFF    F1,BUFFER,LENGTH    READ RECORD INTO BUFFER
190a    CLOOP     CLEAR     X                   CLEAR LOOP COUNTER
190b              CLEAR     A
190c              CLEAR     S
190d             +LDT      #4096                SET MAXIMUM RECORD LENGTH
190e              TD       =X'F1'               TEST INPUT DEVICE
190f              JEQ       *-3                 LOOP UNTIL READY
190g              RD       =X'F1'               READ CHARACTER INTO REG A
190h              COMPR     A,S                 TEST FOR END OF RECORD
190i              JEQ       *+11                EXIT LOOP IF EOR
190j              STCH      BUFFER,X            STORE CHARACTER IN BUFFER
190k              TIXR      T                   LOOP UNLESS MAXIMUM LENGTH
190l              JLT       *-19                  HAS BEEN REACHED
190m              STX       LENGTH              SAVE RECORD LENGTH
```

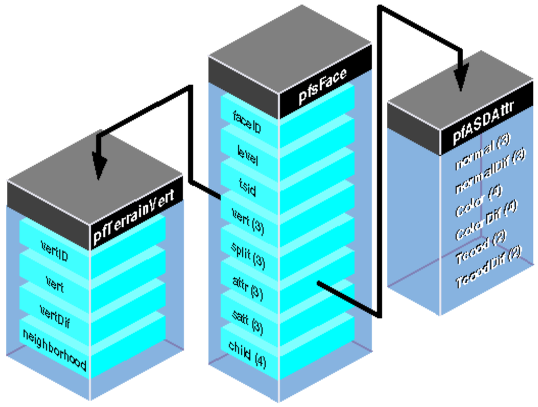
```
195                LDA      LENGTH             TEST FOR END OF FILE
200                COMP    #0
205                JEQ      ENDFIL             EXIT IF EOF FOUND
210                WRBUFF   05,BUFFER,LENGTH   WRITE OUTPUT RECORD
210a               CLEAR    X                  CLEAR LOOP COUNTER
210b               LDT      LENGTH
210c               LDCH     BUFFER,X           GET CHARACTER FROM BUFFER
210d               TD      =X'05'              TEST OUTPUT DEVICE
210e               JEQ      *-3                LOOP UNTIL READY
210f               WD      =X'05'              WRITE CHARACTER
210g               TIXR     T                  LOOP UNTIL ALL CHARACTERS
210h               JLT      *-14                 HAVE BEEN WRITTEN
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
```

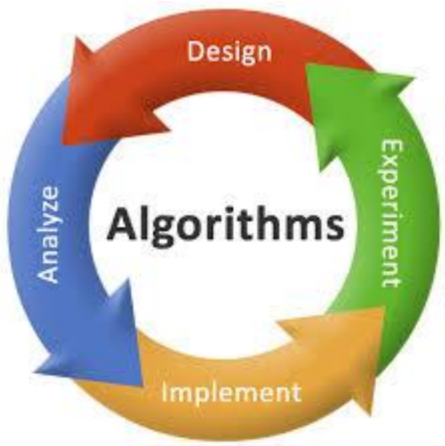
	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**	

Class	III CSE
Subject Code	CS2304
Subject	System Software
Prepared By	Kaviya.P
Lesson Plan for	Macro Processor Algorithm And Data Structures
Time:	50 Minutes
Lesson. No	2/9

1. **Topics to be covered**
   - Macro Processor Algorithm and Data Structures
2. **Skills addressed**:

   Listening
3. **Objectives of this lesson plan:**
   - To enable students to understand macro processor algorithm and data structures
4. **Outcome (s):**
   - Able to explain macro processor algorithm and data structures.
5. **Link sheet:**
   - Define algorithm.
   - Define data structure.
6. **Evocation:**





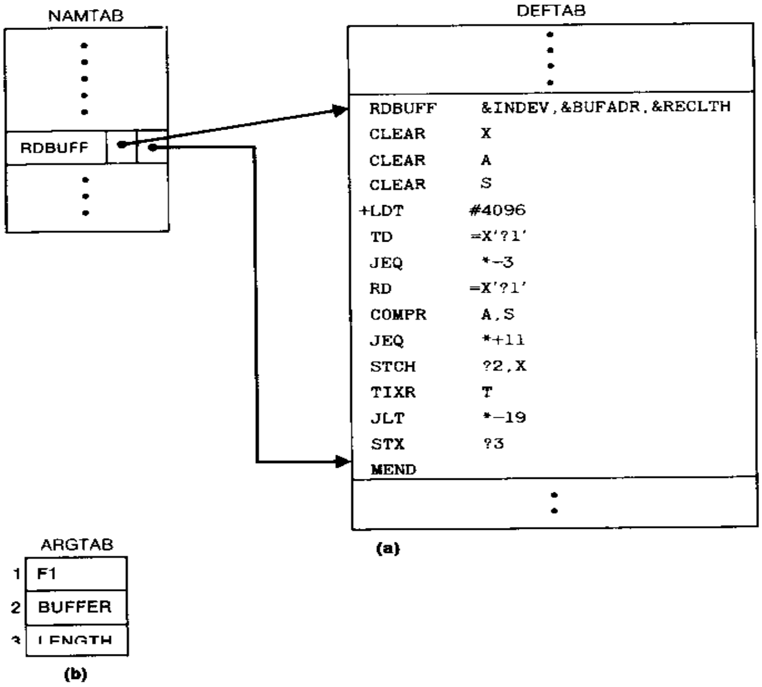
7. **Lecture notes (attached)**

8. **Text Book**
   - Leland L. Beck, “System Software – An Introduction to Systems Programming”,

     3rd Edition, Pearson Education Asia, 2006. PP 186 - 191.
9. **Application**
   - ROM

Macro Processor Data Structures

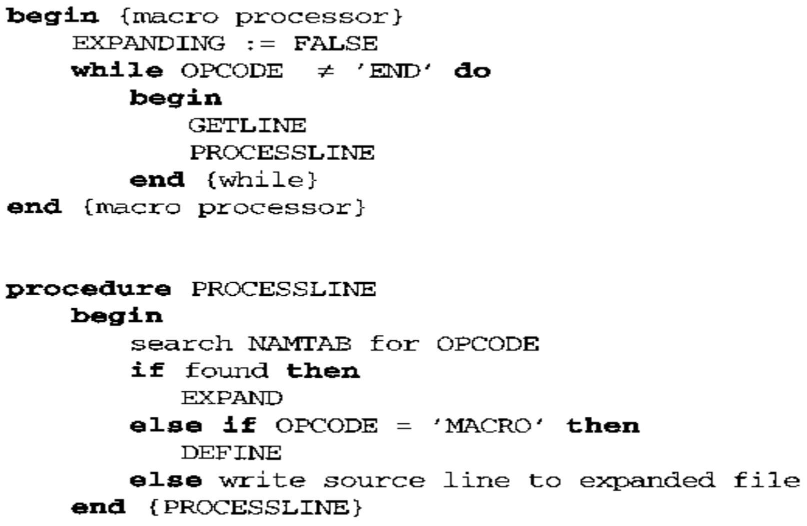
- The macro definitions themselves are stored in definition table (DEFTAB), which contains the macro prototype and the statements that make up the macro body
- The macro names are entered into NAMTAB, which serves as an index to DEFTAB
- For each macro instruction defined NAMTAB contains pointers to the beginning and end of the definition in DEFTAB
- The third data structure is an argument table (ARGTAB), which is used during the expansion of macro invocations
- When a macro invocation statement is recognized, the arguments are stored in ARGTAB according to their position in the argument list



```
RDBUFF    &INDEV,&BUFADR,&RECLTH
CLEAR     X
CLEAR     A
CLEAR     S
+LDT      #4096
TD        =X'?1'
JEQ       *-3
RD        =X'?1'
COMPR     A,S
JEQ       *+11
STCH      ?2,X
TIXR      T
JLT       *-19
STX       ?3
MEND
```

	ARGTAB
1	F1
2	BUFFER
3	LENGTH

Algorithm for a One-pass Macro Processor



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


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}
```

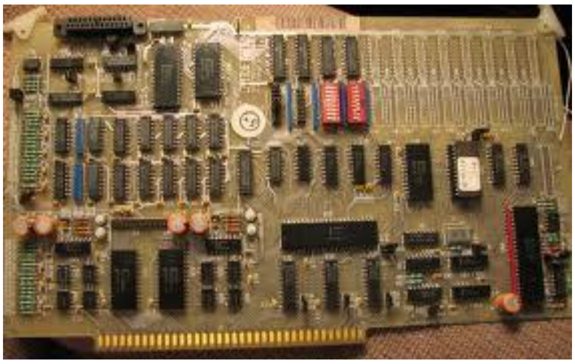
Figure 4.5 Algorithm for a one-pass macro processor.

```
 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}
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 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}
```

	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**	
Class	III CSE	
Subject Code	CS2304	
Subject	System Software	
Prepared By	Kaviya.P	
Lesson Plan for	Machine Independent Macro Processor Features – Concatenation of Macro Parameters, Generation of Unique Labels	
Time:	50 Minutes	
Lesson. No	3/9	

1. **Topics to be covered**
   - Machine Independent Macro Processor – Concatenation of Macro Parameters
   - Generation of Unique Labels
2. **Skills addressed**:

   Listening
3. **Objectives of this lesson plan:**
   - To enable students to understand machine independent macro processor features – Concatenation of macro parameters & Generation of unique labels.
4. **Outcome (s):**
   - Able to explain concatenation of macro parameters.
   - Able to explain generation of unique labels.
5. **Link sheet:**
   - Define macro processor.
6. **Evocation:**



7. **Lecture notes (attached)**

8. **Text Book**
   - Leland L. Beck, "System Software – An Introduction to Systems Programming",

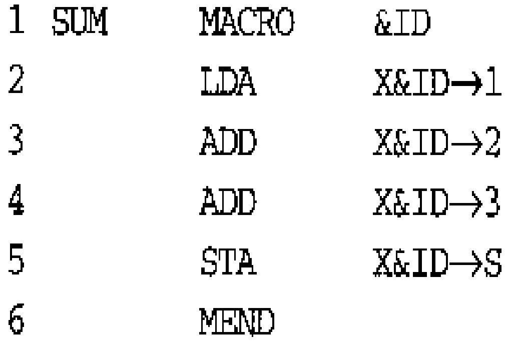
     3rd Edition, Pearson Education Asia, 2006. PP 192 - 195.
9. **Application**
   - Program Development

Machine-Independent Macro Processor Features

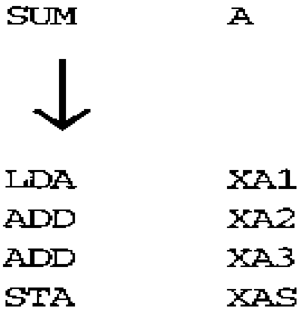
- Concatenation of Macro Parameters
- Generation of Unique Labels
- Conditional Macro Expansion
- Keyword Macro Parameters

Concatenation of Macro Parameters

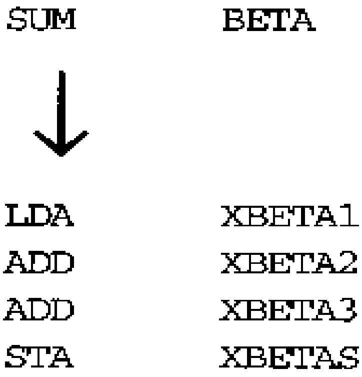
- Most macro processors allow parameters to concatenated with other character strings
- If similar processing is to be performed on each series of variables, the programmer might want to incorporate this processing in to a macro instruction
- The body of the macro definition might contain a statement like “LDA X&ID1” in which the parameter &ID is concatenated after the character string X and before the character string 1
- If the macro definition contained both &ID and &ID1 as parameters, the situation would be ambiguous
- Most macro processors deal with this problem by providing a special concatenation operator (e.g. →)
- LDA X&ID→1



```
SUM     MACRO   &ID
        LDA     X&ID→1
        ADD     X&ID→2
        ADD     X&ID→3
        STA     X&ID→S
        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

- Relative addressing in a source statement may be acceptable for short jumps such as “JEQ *-3*
- For longer jumps spanning several instructions, such notation is very inconvenient, error-prone and difficult to read
- Allow the creation of special types of labels
- Each symbol beginning with $ has been modified by replacing $ with $xx, where xx is a two character alphanumeric counter of the number of macro instructions expanded
- For the first macro expansions, xx will have the value AA
- For succeeding macro expansions, xx will be set to AB, AC, etc

```
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
```

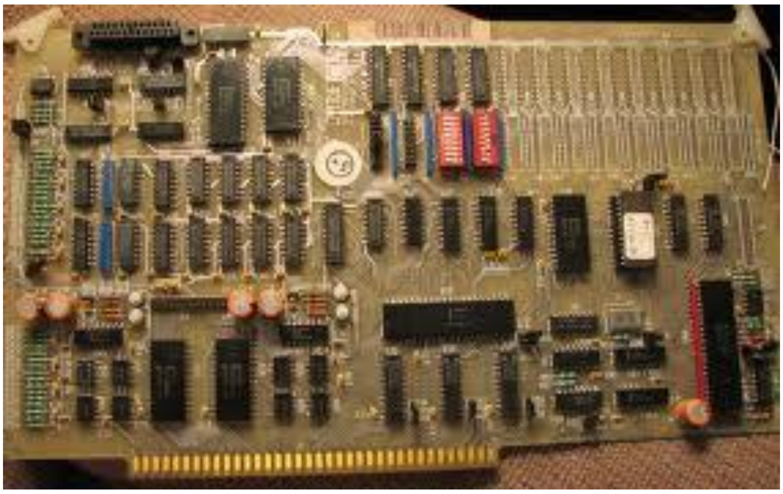
```
              RDBUFF   F1,BUFFER,LENGTH
```

```
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
```

	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**
Class	III CSE
Subject Code	CS2304
Subject	System Software
Prepared By	Kaviya.P
Lesson Plan for	Conditional Macro Expansion
Time:	50 Minutes
Lesson. No	4 & 5/9

1. **Topics to be covered**
   - Conditional Macro Expansion
2. **Skills addressed**:

   Listening
3. **Objectives of this lesson plan:**
   - To enable students to understand conditional macro expansion
4. **Outcome (s):**
   - Able to explain conditional macro expansion.
5. **Link sheet:**
   - Define macro.
6. **Evocation:**



7. **Lecture notes (attached)**

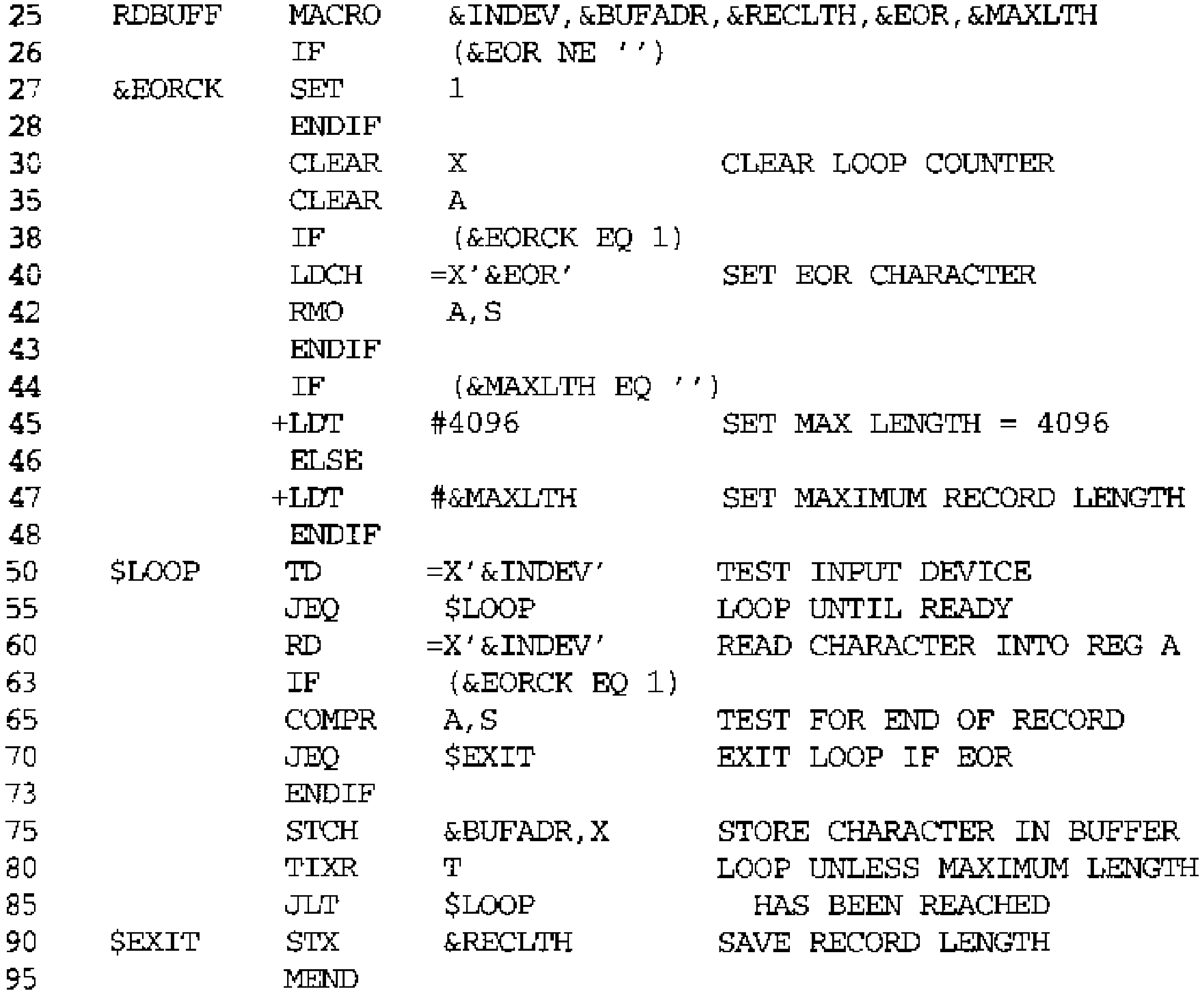
8. **Text Book**
   - Leland L. Beck, “System Software – An Introduction to Systems Programming”,

     3rd Edition, Pearson Education Asia, 2006. PP 195 - 201.
9. **Application**
   - Program Development

Conditional Macro Expansion

- Most macro processors can modify the sequence of statements generated for a macro expansion, depending on the arguments supplied in the macro invocation
- The IF statement evaluates a Boolean expression that is its operand
- If the value of this expression is TRUE, the statements following the IF are generated until an ELSE is encountered
- Otherwise, these statements are skipped, and the statements following the ELSE are generated
- The ENDIF statement terminates the conditional expression that was begun by the IF statement
- The macro processor must maintain a symbol table that contains the values of all macro-time variables used
- Entries in this table are made or modified when SET statements are processed
- The implementation outlined above does not allow for nested IF structures
- WHILE: a macro-time looping statement
- The WHILE statement specifies that the following lines, until the next ENDW statement, are to be generated repeatedly as long as a particular condition is true
- The macro-time variable &CTR is used to count the number of times the lines following the WHILE statement have been generated

Use of Macro-time Conditional Statements



```
RDBUFF   MACRO   &INDEV,&BUFADR,&RECLTH,&EOR,&MAXLTH
         IF      (&EOR NE '')
&EORCK   SET     1
         ENDIF
         CLEAR   X                CLEAR LOOP COUNTER
         CLEAR   A
         IF      (&EORCK EQ 1)
         LDCH    =X'&EOR'         SET EOR CHARACTER
         RMO     A,S
         ENDIF
         IF      (&MAXLTH EQ '')
        +LDT     #4096            SET MAX LENGTH = 4096
         ELSE
        +LDT     #&MAXLTH         SET MAXIMUM RECORD LENGTH
         ENDIF
$LOOP    TD      =X'&INDEV'       TEST INPUT DEVICE
         JEQ     $LOOP            LOOP UNTIL READY
         RD      =X'&INDEV'       READ CHARACTER INTO REG A
         IF      (&EORCK EQ 1)
         COMPR   A,S              TEST FOR END OF RECORD
         JEQ     $EXIT            EXIT LOOP IF EOR
         ENDIF
         STCH    &BUFADR,X        STORE CHARACTER IN BUFFER
         TIXR    T                LOOP UNLESS MAXIMUM LENGTH
         JLT     $LOOP              HAS BEEN REACHED
$EXIT    STX     &RECLTH          SAVE RECORD LENGTH
         MEND
```

```
         RDBUFF   F3, BUF, RECL, 04, 2048
```

```
30              CLEAR    X              CLEAR LOOP COUNTER
35              CLEAR    A
40              LDCH    =X'04'          SET EOR CHARACTER
42              RMO      A,S
47             +LDT     #2048           SET MAXIMUM RECORD LENGTH
50    $AALOOP   TD      =X'F3'          TEST INPUT DEVICE
55              JEQ      $AALOOP        LOOP UNTIL READY
60              RD      =X'F3'          READ CHARACTER INTO REG A
65              COMPR    A,S            TEST FOR END OF RECORD
70              JEQ      $AAEXIT        EXIT LOOP IF EOR
75              STCH     BUF,X          STORE CHARACTER IN BUFFER
80              TIXR     T              LOOP UNLESS MAXIMUM LENGTH
85              JLT      $AALOOP          HAS BEEN REACHED
90    $AAEXIT   STX      RECL           SAVE RECORD LENGTH
```

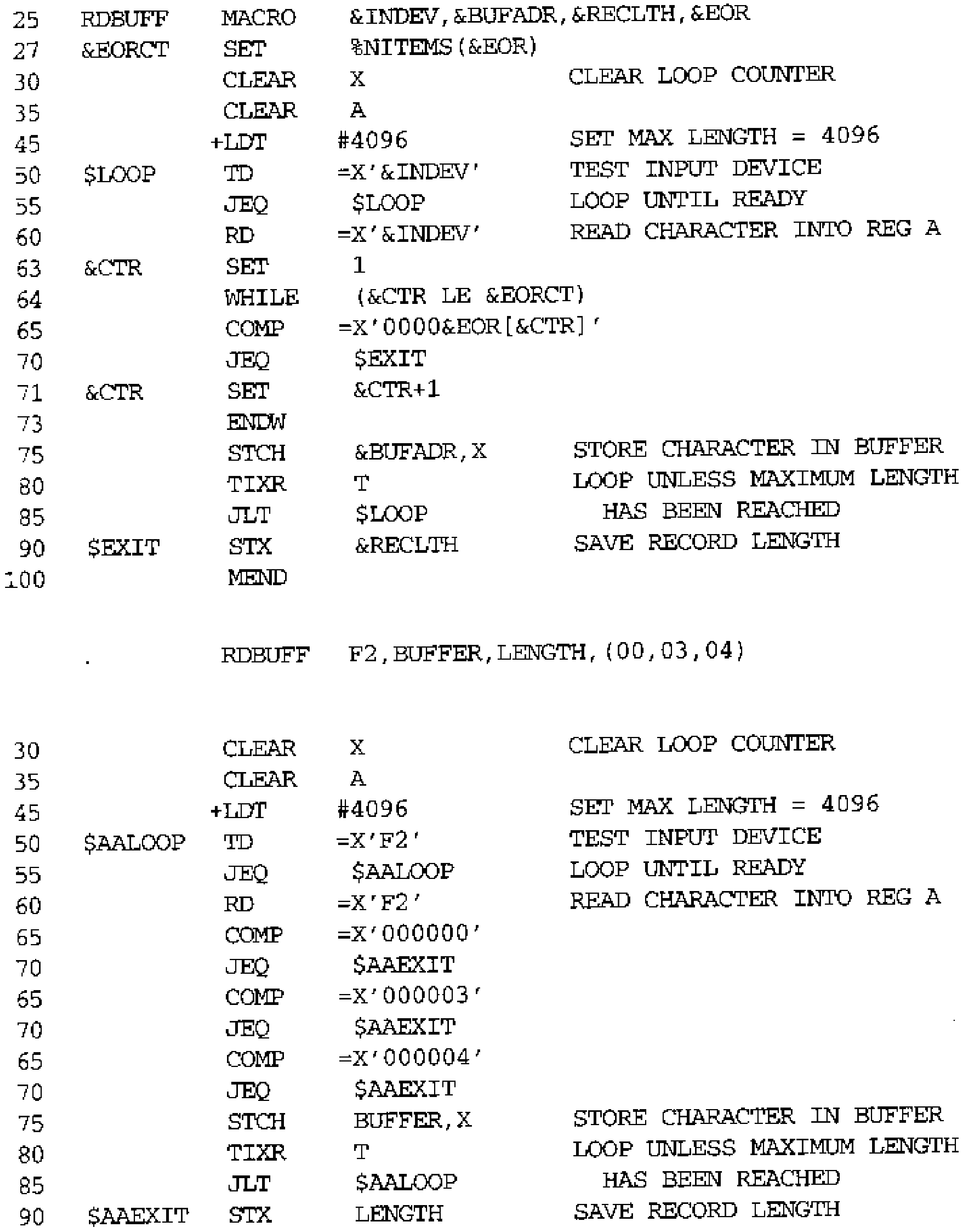
```
      .         RDBUFF   0E,BUFFER,LENGTH,,80
```

```
30              CLEAR    X              CLEAR LOOP COUNTER
35              CLEAR    A
47             +LDT     #80             SET MAXIMUM RECORD LENGTH
50    $ABLOOP   TD      =X'0E'          TEST INPUT DEVICE
55              JEQ      $ABLOOP        LOOP UNTIL READY
60              RD      =X'0E'          READ CHARACTER INTO REG A
75              STCH     BUFFER,X       STORE CHARACTER IN BUFFER
80              TIXR     T              LOOP UNLESS MAXIMUM LENGTH
87              JLT      $ABLOOP          HAS BEEN REACHED
90    $ABEXIT   STX      LENGTH         SAVE RECORD LENGTH
```

```
      .         RDBUFF   F1,BUFF,RLENG,04
```

```
30              CLEAR    X              CLEAR LOOP COUNTER
35              CLEAR    A
40              LDCH    =X'04'          SET EOR CHARACTER
42              RMO      A,S
45             +LDT     #4096           SET MAX LENGTH = 4096
50    $ACLOOP   TD      =X'F1'          TEST INPUT DEVICE
55              JEQ      $ACLOOP        LOOP UNTIL READY
60              RD      =X'F1'          READ CHARACTER INTO REG A
65              COMPR    A,S            TEST FOR END OF RECORD
70              JEQ      $ACEXIT        EXIT LOOP IF EOR
75              STCH     BUFF,X         STORE CHARACTER IN BUFFER
80              TIXR     T              LOOP UNLESS MAXIMUM LENGTH
85              JLT      $ACLOOP          HAS BEEN REACHED
90    $ACEXIT   STX      RLENG          SAVE RECORD LENGTH
```

Use of Macro-time looping Statements

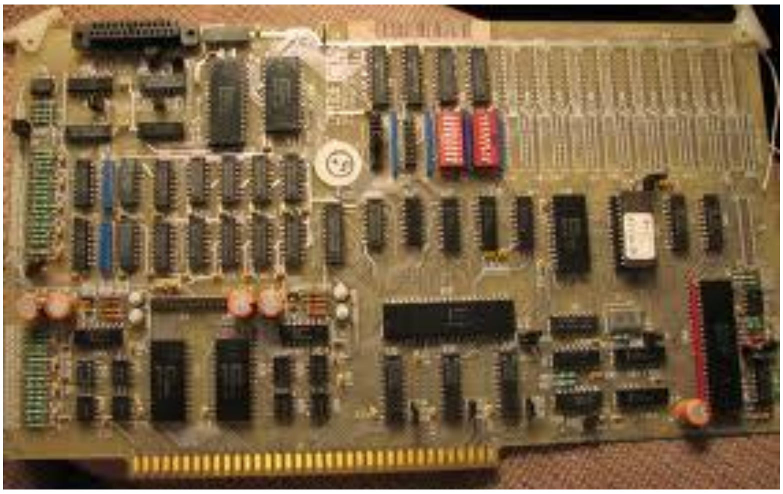


```
25    RDBUFF   MACRO   &INDEV,&BUFADR,&RECLTH,&EOR
27    &EORCT   SET     %NITEMS(&EOR)
30             CLEAR   X                  CLEAR LOOP COUNTER
35             CLEAR   A
45            +LDT     #4096              SET MAX LENGTH = 4096
50    $LOOP    TD      =X'&INDEV'         TEST INPUT DEVICE
55             JEQ     $LOOP              LOOP UNTIL READY
60             RD      =X'&INDEV'         READ CHARACTER INTO REG A
63    &CTR     SET     1
64             WHILE   (&CTR LE &EORCT)
65             COMP    =X'0000&EOR[&CTR]'
70             JEQ     $EXIT
71    &CTR     SET     &CTR+1
73             ENDW
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
100            MEND

      .        RDBUFF  F2,BUFFER,LENGTH,(00,03,04)


30             CLEAR   X                  CLEAR LOOP COUNTER
35             CLEAR   A
45            +LDT     #4096              SET MAX LENGTH = 4096
50    $AALOOP  TD      =X'F2'             TEST INPUT DEVICE
55             JEQ     $AALOOP            LOOP UNTIL READY
60             RD      =X'F2'             READ CHARACTER INTO REG A
65             COMP    =X'000000'
70             JEQ     $AAEXIT
65             COMP    =X'000003'
70             JEQ     $AAEXIT
65             COMP    =X'000004'
70             JEQ     $AAEXIT
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
```

	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**	
Class	III CSE	
Subject Code	CS2304	
Subject	System Software	
Prepared By	Kaviya.P	
Lesson Plan for	Keyword Macro Parameters	
Time:	50 Minutes	
Lesson. No	6/9	

1. **Topics to be covered**
   - Keyword Macro Parameters
2. **Skills addressed**:

   Listening
3. **Objectives of this lesson plan:**
   - To enable students to understand keyword macro parameters
4. **Outcome (s):**
   - Able to explain keyword macro parameters.
5. **Link sheet:**
   - Define macro.
6. **Evocation:**



7. **Lecture notes (attached)**

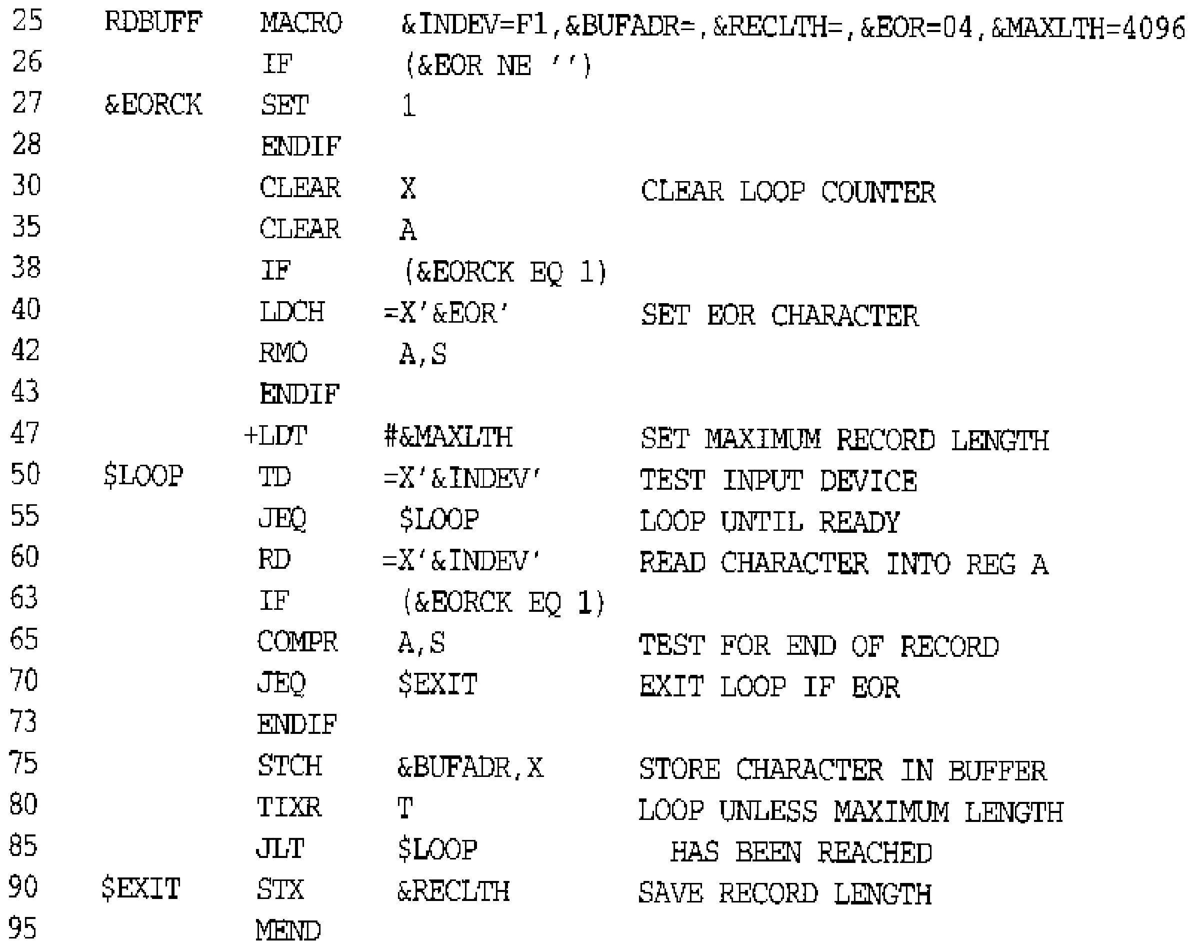
8. **Text Book**
   - Leland L. Beck, “System Software – An Introduction to Systems Programming”,

     3rd Edition, Pearson Education Asia, 2006. PP 202 - 204.
9. **Application**
   - Program Development

Keyword Macro Parameters

- Positional parameter: parameters and arguments were associated with each other according to their positions in the macro prototype and the macro invocation statement
- Keyword parameters: each argument value is written with a keyword that named the corresponding parameter
- Each parameter name is followed by an equal sign, which identifies a keyword parameter
- The parameter is assumed to have the default value if its name does not appear in the macro invocation statement

Use of Keyword Parameters in Macro Instructions



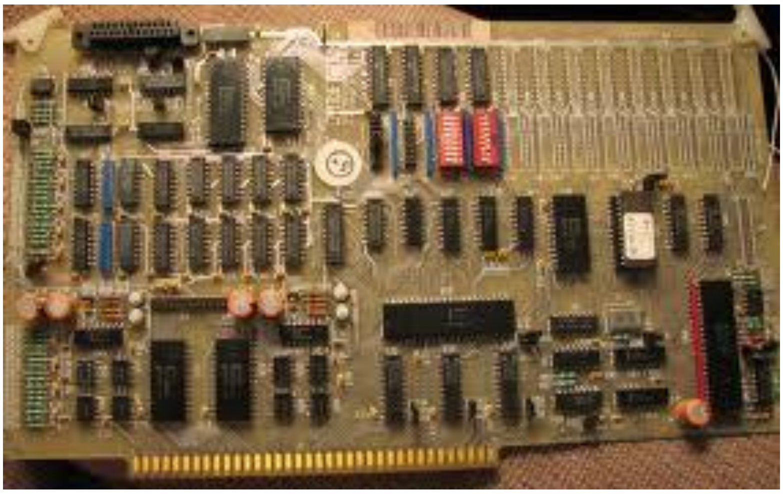
```
RDBUFF   MACRO   &INDEV=F1,&BUFADR=,&RECLTH=,&EOR=04,&MAXLTH=4096
         IF      (&EOR NE '')
&EORCK   SET     1
         ENDIF
         CLEAR   X               CLEAR LOOP COUNTER
         CLEAR   A
         IF      (&EORCK EQ 1)
         LDCH    =X'&EOR'        SET EOR CHARACTER
         RMO     A,S
         ENDIF
        +LDT     #&MAXLTH        SET MAXIMUM RECORD LENGTH
$LOOP    TD      =X'&INDEV'      TEST INPUT DEVICE
         JEQ     $LOOP           LOOP UNTIL READY
         RD      =X'&INDEV'      READ CHARACTER INTO REG A
         IF      (&EORCK EQ 1)
         COMPR   A,S             TEST FOR END OF RECORD
         JEQ     $EXIT           EXIT LOOP IF EOR
         ENDIF
         STCH    &BUFADR,X       STORE CHARACTER IN BUFFER
         TIXR    T               LOOP UNLESS MAXIMUM LENGTH
         JLT     $LOOP             HAS BEEN REACHED
$EXIT    STX     &RECLTH         SAVE RECORD LENGTH
         MEND
```

Sri Vidya College of Engineering and Technology
Department of Computer Science & Engineering

Class	III CSE
Subject Code	CS2304
Subject	System Software
Prepared By	Kaviya.P
Lesson Plan for	Macro Within Macro
Time:	50 Minutes
Lesson. No	7&8/9

1. **Topics to be covered**
    - Macro within Macro
2. **Skills addressed**:

    Listening
3. **Objectives of this lesson plan:**
    - To enable students to understand macro within macro
4. **Outcome (s):**
    - Able to explain macro within macro.
5. **Link sheet:**
    - Define macro.
6. **Evocation:**



7. **Lecture notes (attached)**

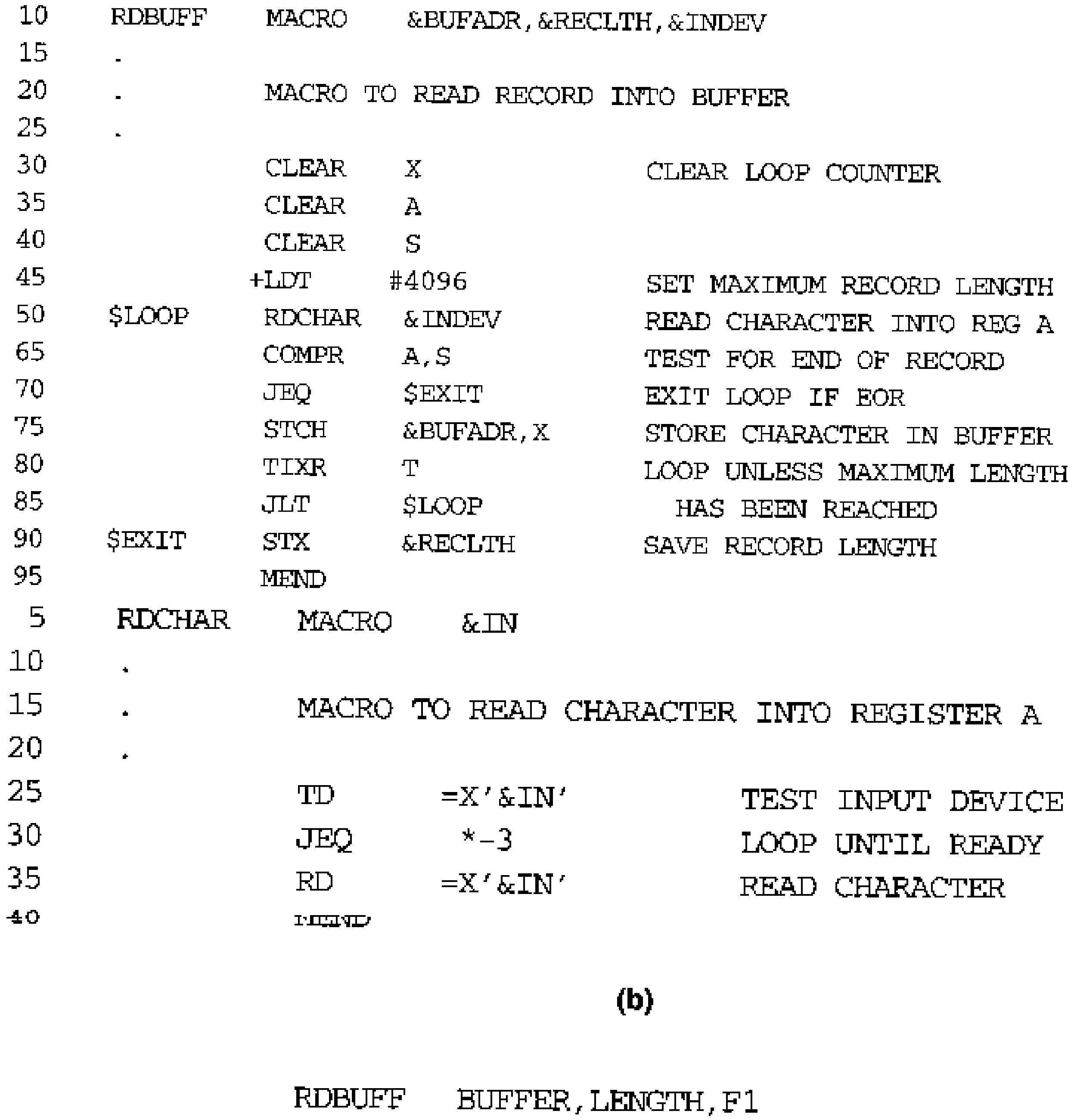
8. **Text Book**
    - Leland L. Beck, “System Software – An Introduction to Systems Programming”,

      3rd Edition, Pearson Education Asia, 2006. PP 204 - 209.
9. **Application**
    - Program Development

Macro within Macro

- Macro within macro can be solved if the macro processor is being written in a programming language that allows recursive calls
- The compiler would be sure that previous value of any variables declared within a procedure were saved when that procedure was called recursively
- If would take care of other details involving return from the procedure

Example of Nested Macro Invocation



```
10   RDBUFF   MACRO    &BUFADR,&RECLTH,&INDEV
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   $LOOP    RDCHAR   &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
 5   RDCHAR   MACRO    &IN
10   .
15   .        MACRO TO READ CHARACTER INTO REGISTER A
20   .
25            TD       =X'&IN'            TEST INPUT DEVICE
30            JEQ       *-3               LOOP UNTIL READY
35            RD       =X'&IN'            READ CHARACTER
40            MEND
```

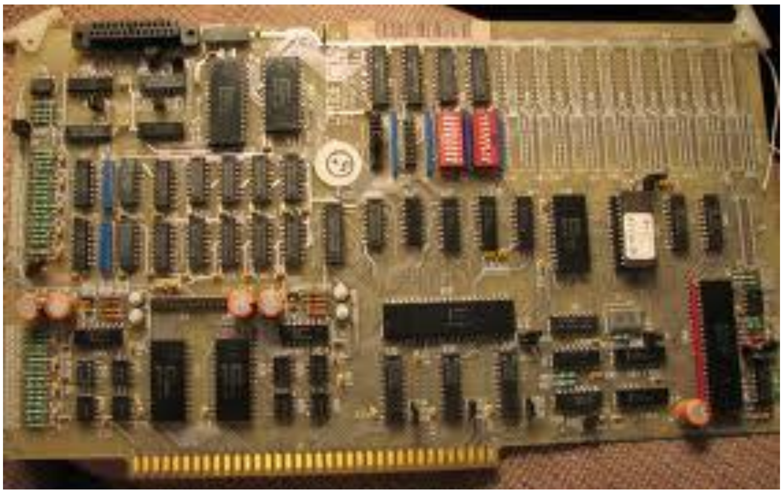
(b)

```
              RDBUFF   BUFFER,LENGTH,F1
```

	Sri Vidya College of Engineering and Technology **Department of Computer Science & Engineering**	
Class	III CSE	
Subject Code	CS2304	
Subject	System Software	
Prepared By	Kaviya.P	
Lesson Plan for	Implementation Example – MASM	
Time:	50 Minutes	
Lesson. No	9/9	

1. **Topics to be covered**
    - Implementation Example – MASM
2. **Skills addressed**:

    Listening
3. **Objectives of this lesson plan:**
    - To enable students to understand Implementation Example – MASM
4. **Outcome (s):**
    - Able to explain MASM macro processor.
5. **Link sheet:**
    - Define macro processor.
6. **Evocation:**



7. **Lecture notes (attached)**

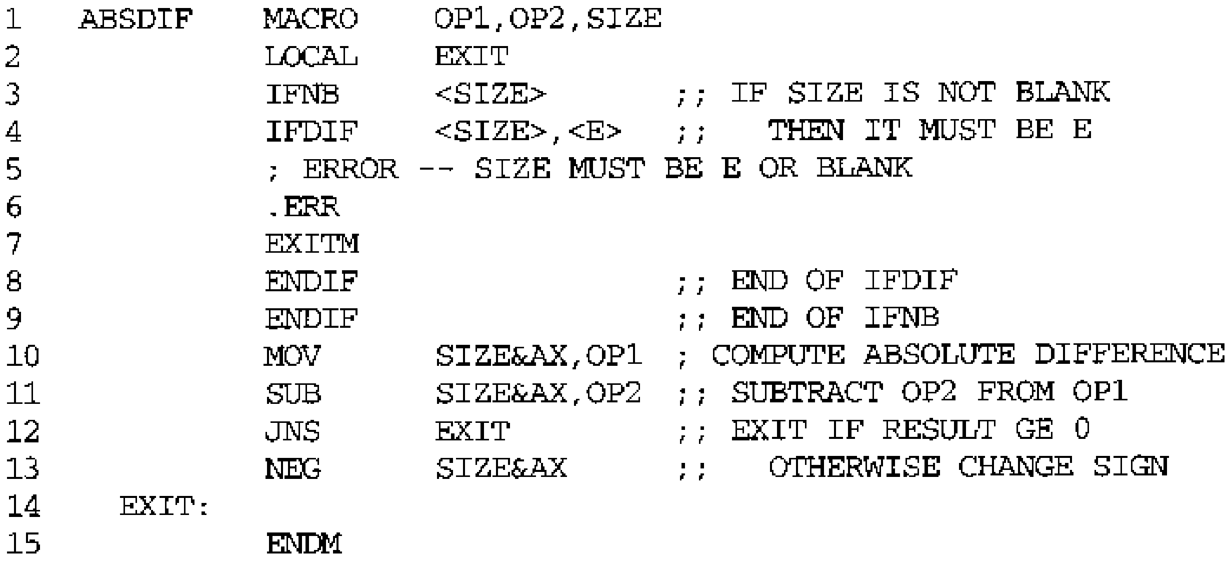
8. **Text Book**
    - Leland L. Beck, “System Software – An Introduction to Systems Programming”,

      3rd Edition, Pearson Education Asia, 2006. PP 213 - 216.
9. **Application**
    - Program Development

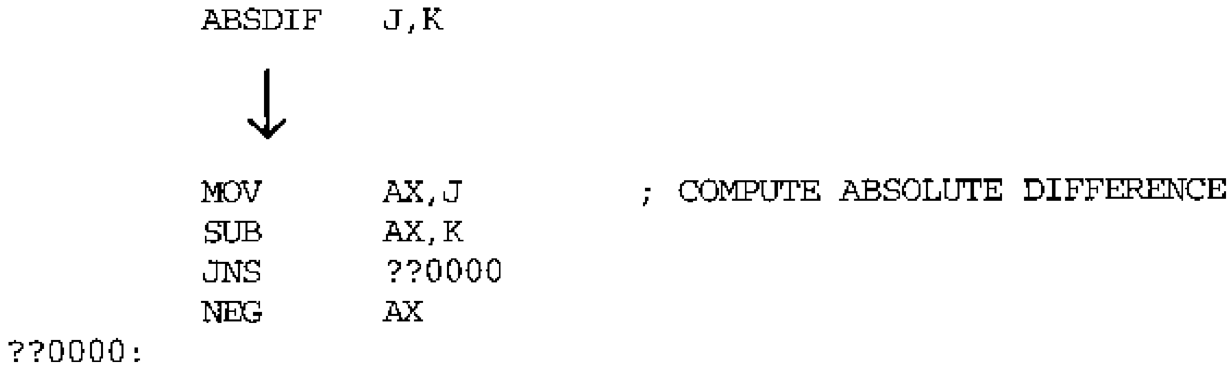
MASM Macro Processor

- The macro processor of MASM is integrated with Pass 1 of the assembler
- MASM generates the unique names of local labels in the form ??n, where n is a hexadecimal number in the range 0000 to FFFF
- .ERR: signals to MASM that an error has been detected
- EXITM: directs MASM to terminate the expansion of the macro
- &: is a concatenation operator
- ;; is a macro comment, serves only as documentation for the macro definition
- ; is an ordinary assembler language comment, included as part of the macro expansion
- IRP: sets the macro-time variable to a sequence of values specified in <…>
- The statements between the TRP and the matching ENDM are generated once for each value of the variable

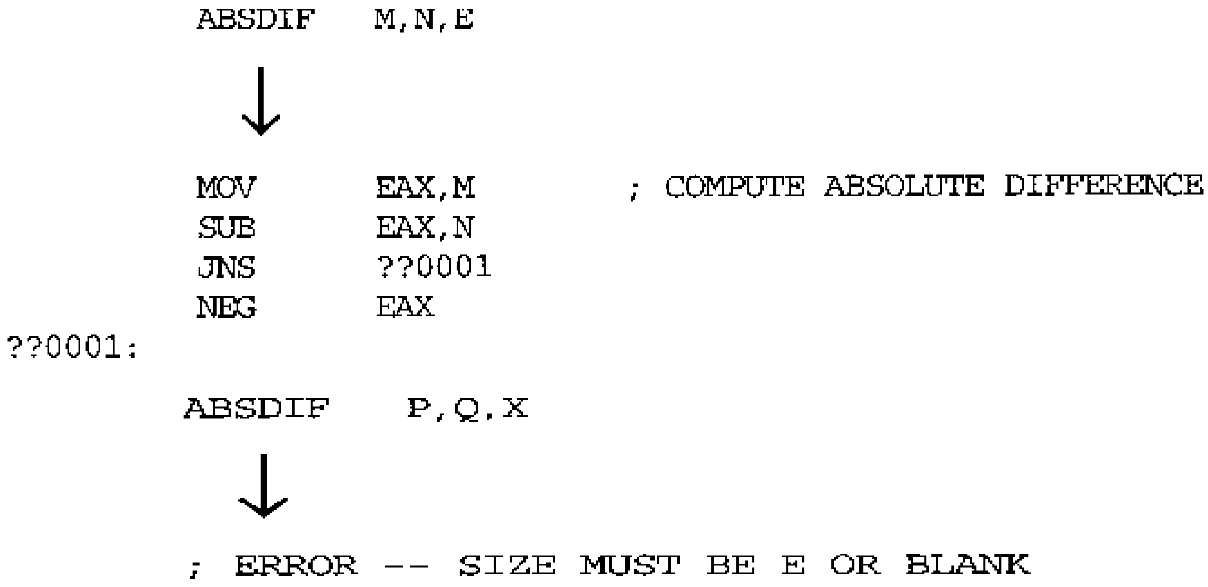
Examples of MASM Macro and Conditional Statements



```
ABSDIF     MACRO     OP1,OP2,SIZE
           LOCAL     EXIT
           IFNB      <SIZE>          ;; IF SIZE IS NOT BLANK
           IFDIF     <SIZE>,<E>      ;;   THEN IT MUST BE E
           ; ERROR -- SIZE MUST BE E OR BLANK
           .ERR
           EXITM
           ENDIF                     ;; END OF IFDIF
           ENDIF                     ;; END OF IFNB
           MOV       SIZE&AX,OP1    ; COMPUTE ABSOLUTE DIFFERENCE
           SUB       SIZE&AX,OP2    ;; SUBTRACT OP2 FROM OP1
           JNS       EXIT            ;; EXIT IF RESULT GE 0
           NEG       SIZE&AX         ;;   OTHERWISE CHANGE SIGN
  EXIT:
           ENDM
```

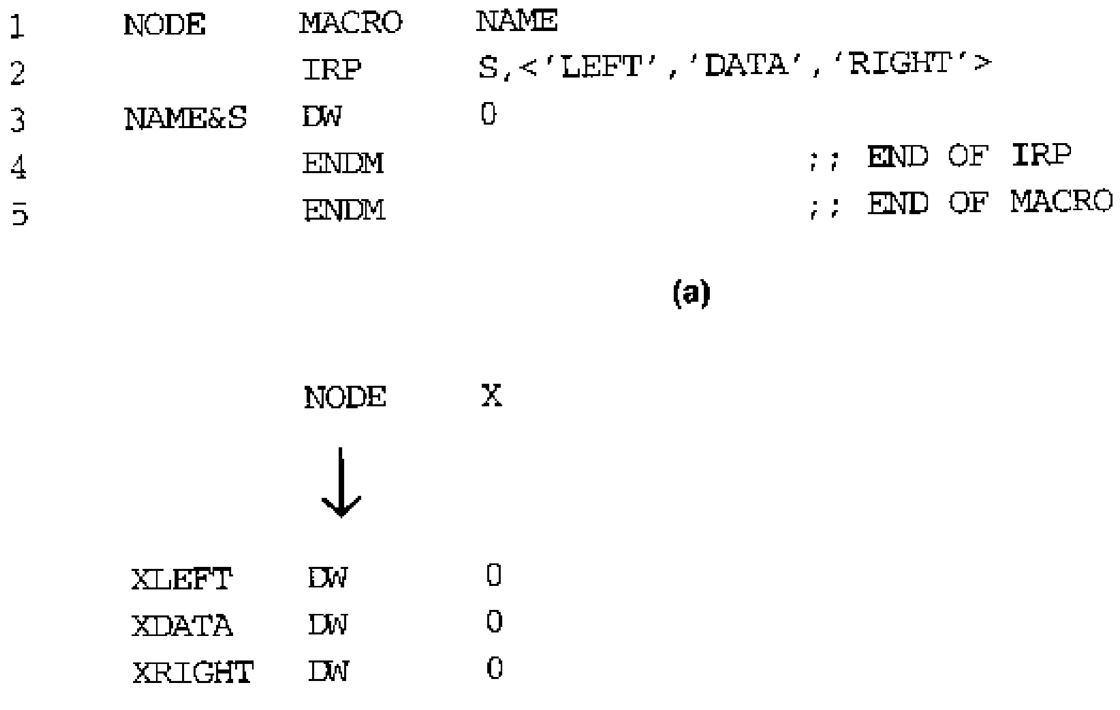


```
        ABSDIF    J,K

          ↓

        MOV       AX,J          ; COMPUTE ABSOLUTE DIFFERENCE
        SUB       AX,K
        JNS       ??0000
        NEG       AX
??0000:
```



```
          ABSDIF    M,N,E

             ↓

          MOV       EAX,M         ; COMPUTE ABSOLUTE DIFFERENCE
          SUB       EAX,N
          JNS       ??0001
          NEG       EAX
??0001:

          ABSDIF    P,Q,X

             ↓

          ; ERROR -- SIZE MUST BE E OR BLANK
```

Example of MASM Iteration Statement



```
1     NODE      MACRO    NAME
2               IRP      S,<'LEFT','DATA','RIGHT'>
3     NAME&S    DW       0
4               ENDM                        ;; END OF IRP
5               ENDM                        ;; END OF MACRO
```

(a)

```
                NODE     X

                  ↓

      XLEFT     DW       0
      XDATA     DW       0
      XRIGHT    DW       0
```

(b)

Figure 4.13 Example of MASM iteration statement.