H 5

```
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
```

```
_$2

Sym
---
CMP
DEC
DEC
DEC
DEC

DEC
EXE
EXE
EXE
MMC
MMC
MOV
PRS
SYS
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
```

FILEID**BOOSTRING

F 12

BOO
V04

```
BBBBBBBB      OOOOOO      OOOOOO      SSSSSSSS  TTTTTTTTTT  RRRRRRRR      IIIIII    NN      NN    GGGGGGGG
BBBBBBBB      OOOOOO      OOOOOO      SSSSSSSS  TTTTTTTTTT  RRRRRRRR      IIIIII    NN      NN    GGGGGGGG
BB      BB  OO      OO  OO      OO  SS              TT      RR      RR      II      NN      NN  GG
BB      BB  OO      OO  OO      OO  SS              TT      RR      RR      II      NN      NN  GG
BB      BB  OO      OO  OO      OO  SS              TT      RR      RR      II      NNNN    NN  GG
BB      BB  OO      OO  OO      OO  SS              TT      RR      RR      II      NNNN    NN  GG
BBBBBBBB    OO      OO  OO      OO    SSSSSS        TT      RRRRRRRR        II      NN  NN  NN  GG
BBBBBBBB    OO      OO  OO      OO    SSSSSS        TT      RRRRRRRR        II      NN  NN  NN  GG
BB      BB  OO      OO  OO      OO          SS      TT      RR  RR          II      NN    NNNN  GG  GGGGGG
BB      BB  OO      OO  OO      OO          SS      TT      RR  RR          II      NN    NNNN  GG  GGGGGG
BB      BB  OO      OO  OO      OO          SS      TT      RR    RR        II      NN      NN  GG      GG      ....
BB      BB  OO      OO  OO      OO          SS      TT      RR    RR        II      NN      NN  GG      GG      ....
BBBBBBBB      OOOOOO      OOOOOO    SSSSSSSS        TT      RR      RR    IIIIII    NN      NN    GGGGGG        ....
BBBBBBBB      OOOOOO      OOOOOO    SSSSSSSS        TT      RR      RR    IIIIII    NN      NN    GGGGGG        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
00000001  0000      1         BOOT_SWITCH = 1                  ; Include bootstrap emulation subset
```

```
0000          .NOSHOW CONDITIONALS
0000          .TITLE  BOO$STRING      Subset Instruction Emulation for VMB and SYSBOOT
0000          .IDENT  /V04-001/
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0C00    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility:
0000    ;
0000    ;       VAX-11 Instruction Emulator
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       The routines in this module emulate the VAX-11 string instructions.
0000    ;       These procedures can be a part of an emulator package or can be
0000    ;       called directly after the input parameters have been loaded into
0000    ;       the architectural registers.
0000    ;
0000    ;       The input parameters to these routines are the registers that
0000    ;       contain the intermediate instruction state.
0000    ;
0000    ; Environment:
0000    ;
0000    ;       These routines run at any access mode, at any IPL, and are AST
0000    ;       reentrant.
0000    ;
0000    ; Author:
0000    ;
0000    ;       Lawrence J. Kenah
0000    ;
0000    ; Creation Date:
0000    ;
0000    ;       16 August 1982
0000    ;
```

```
0000     ; Modified by:
0000     ;
0000     ;       V04-001 LJK0044         Lawrence J. Kenah       6-Sep-1984
0000     ;               The backup code for MOVTC when moving in the forward direction
0000     ;               also needs to be changed (see LJK0039) based on the relative
0000     ;               sizes of the source and destination strings.
0000     ;
0000     ;       V01-005 KDM0107         Kathleen D. Morse       21-Aug-1984
0000     ;               Fix bug in CMPC3.  Return C clear if string length is 0.
0000     ;
0000     ;       V01-004 LJK0039         Lawrence J. Kenah       20-Jul-1984
0000     ;               Mofify MOVTC backup code to reflect differences in register
0000     ;               contents when traversing strings backwards. There are two
0000     ;               cases based on the relative sizes of source and destination.
0000     ;
0000     ;       V01-003 LJK0026         Lawrence J. Kenah       19-Mar-1984
0000     ;               Final cleanup pass. Access violation handler is now called
0000     ;               STRING_ACCVIO. Set PACK_M_ACCVIO bit in R1 before passing
0000     ;               control to VAX$REFLECT_FAULT.
0000     ;
0000     ;       V01-002 LJK0011         Lawrence J. Kenah       8-Nov-1983
0000     ;               Fix three minor bugs in MOVTC and MOVTUC. Change exception
0000     ;               handling to reflect chenged implementation.
0000     ;
0000     ;       V01-001 Original        Lawrence J. Kenah       16-Aug-1982
0000     ;--
```

```
BOO
Sym

BOO
IDE
NO_
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
OP$
```

```
0000            .SUBTITLE       Miscellaneous Notes
0000    ;+
0000    ;       The following notes apply to most or all of the routines that appear in
0000    ;       this module. The comments appear here to avoid duplication in each routine.
0000    ;
0000    ;   1.  The VAX Architecture Standard (DEC STD 032) is the ultimate authority on
0000    ;       the functional behavior of these routines. A summary of each instruction
0000    ;       that is emulated appears in the Functional Description section of each
0000    ;       routine header.
0000    ;
0000    ;   2.  One design goal that affects the algorithms used is that these instructions
0000    ;       can incur exceptions such as access violations that will be reported to
0000    ;       users in such a way that the exception appears to have originated at the
0000    ;       site of the reserved instruction rather than within the emulator. This
0000    ;       constraint affects the algorithms available and dictates specific
0000    ;       implementation decisions.
0000    ;
0000    ;   3.  Each routine header contains a picture of the register usage when it is
0000    ;       necessary to store the intermediate state of an instruction (routine) while
0000    ;       servicing an exception.
0000    ;
0000    ;       The delta-PC field is used by the condition handler jacket to these
0000    ;       routines when it determines that an exception such as an access violation
0000    ;       occurred in response to an explicit use of one of the reserved
0000    ;       instructions. These routines can also be called directly with the input
0000    ;       parameters correctly placed in registers. The delta-PC field is not used in
0000    ;       this case.
0000    ;
0000    ;       Note that the input parameters to any routine are a subset of the
0000    ;       intermediate state picture.
0000    ;
0000    ;       Fields that are not used either as input parameters or to store
0000    ;       intermediate state are indicated thus, XXXXX.
0000    ;
0000    ;   4.  In the Input Parameter list for each routine, certain register fields that
0000    ;       are not used may be explicitly listed for one reason or another. These
0000    ;       unused input parameters are described as IRRELEVANT.
0000    ;
0000    ;   5.  In general, the final condition code settings are determined as the side
0000    ;       effect of one of the last instructions that executes before control is
0000    ;       passed back to the caller with an RSB. It is seldom necessary to explicitly
0000    ;       manipulate condition codes with a BIxPSW instruction or similar means.
0000    ;
0000    ;   6.  There is only a small set of exceptions that are reflected to the user in an
0000    ;       altered fashion, with the exception PC changed from within the emulator to
0000    ;       the site of the original entry into these routines. The instructions that
0000    ;       generate these exceptions are all immediately preceded by a
0000    ;
0000    ;               MARK_POINT      yyyy_N
0000    ;
0000    ;       where yyyy is the instruction name and N is a small integer. These names
0000    ;       map directly into instruction- and context-specific routines (located at
0000    ;       the end of this module) that put each instruction (routine) into a
0000    ;       consistent state before passing control to a more general exception handler
0000    ;       in a different module.
0000    ;-
```

```
    0000          .SUBTITLE       DECLARATIONS
    0000
    0000  ; Include files:
    0000
    0000          $PSLDEF                         ; Define bit fields in PSL
    0000
    0000          .NOCROSS                        ; No cross reference for these
    0000          .ENABLE         SUPPRESSION     ; No symbol table entries either
    0000
    0000          PACK_DEF                        ; Stack usage for exception handling
    0000
    0000          .DISABLE        SUPPRESSION     ; Turn on symbol table again
    0000          .CROSS                          ; Cross reference is OK now
    0000
    0000  ; Macro Definitions
    0000
    0000          .MACRO  _INCLUDE        OPCODE , BOOT_FLAG
    0000          .IF     NOT_DEFINED     BOOT_SWITCH
    0000                  OPCODE'_DEF
    0000                  INCLUDE_'OPCODE = 0
    0000          .IF_FALSE
    0000                  .IF     IDENTICAL       <BOOT_FLAG> , BOOT
    0000                          OPCODE'_DEF
    0000                          INCLUDE_'OPCODE = 0
    0000                  .ENDC
    0000          .ENDC
    0000          .ENDM   _INCLUDE
    0000
    0000  ; External declarations
    0000
    0000          .DISABLE        GLOBAL
    0000
    0000
    0000  ; PSECT Declarations:
    0000
    0000          .DEFAULT        DISPLACEMENT , WORD
    0000
00000000          .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG
    0000
    0000          BEGIN_MARK_POINT                ; Set up exception mark points
```

```
0000         .SUBTITLE      Conditional Assembly Parameters
0000   ;+
0000   ; Functional Description:
0000   ;
0000   ;      It is possible to create a subset emulator, one that emulates
0000   ;      specific reserved instructions. This capability is currently exploited
0000   ;      to create a subset emulator for use by the bootstrap programs.
0000   ;
0000   ;      An instruction is included in the full emulator by making an entry
0000   ;      in the following table. If the optional second parameter is present
0000   ;      and equal to BOOT, then that instruction is included in the subset
0000   ;      emulator used by the bootstrap code.
0000   ;-
0000
0000         .NOCROSS                              ; No cross reference for these
0000         .ENABLE        SUPPRESSION            ; No symbol table entries either
0000
0000         _INCLUDE       MOVTC
0000         _INCLUDE       MOVTUC
0000         _INCLUDE       CMPC3 , BOOT
0000         _INCLUDE       CMPC5 , BOOT
0000         _INCLUDE       SCANC
0000         _INCLUDE       SPANC
0000         _INCLUDE       LOCC , BOOT
0000         _INCLUDE       SKPC
0000         _INCLUDE       MATCHC
0000         _INCLUDE       CRC
0000
0000         .DISAB E       SUPPRESSION            ; Turn on symbol table again
0000         .CROSS                                ; Cross reference is OK now
0000
0000         .NOSHOW        CONDITIONALS
0000
```

```
0000          .SUBTITLE      VAX$CMPC3 - Compare Characters (3 Operand)
0000    ;+
0000    ; Functional Description:
0000    ;
0000    ;       The bytes of string 1 specified by the length and address 1 operands are
0000    ;       compared  with the bytes of string 2 specified by the length and address
0000    ;       2 operands.  Comparison proceeds until inequality is detected or all the
0000    ;       bytes  of  the strings have been examined.  Condition codes are affected
0000    ;       by the result of the last byte  comparison.   Two  zero  length  strings
0000    ;       compare equal (i.e.  Z is set and N, V, and C are cleared).
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       R0<15:0> = len          Length of character strings
0000    ;       R1       = src1addr     Address of first character string (called S1)
0000    ;       R3       = src2addr     Address of second character string (called S2)
0000    ;
0000    ; Intermediate State:
0000    ;
0000    ;        31              23              15              07            00
0000    ;       +----------------+----------------+----------------+----------------+
0000    ;       |    delta-PC    |      XXXX      |              len                | : R0
0000    ;       +----------------+----------------+----------------+----------------+
0000    ;       |                              src1addr                             | : R1
0000    ;       +----------------+----------------+----------------+----------------+
0000    ;       |                               XXXXX                               | : R2
0000    ;       +----------------+----------------+----------------+----------------+
0000    ;       |                              src2addr                             | : R3
0000    ;       +----------------+----------------+----------------+----------------+
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       Strings are IDENTICAL
0000    ;
0000    ;               R0 = 0
0000    ;               R1 = Address of one byte beyond end of S1
0000    ;               R2 = 0 (same as R0)
0000    ;               R1 = Address of one byte beyond end of S2
0000    ;
0000    ;       Strings DO NOT MATCH
0000    ;
0000    ;               R0 = Number of bytes left in strings (including first byte
0000    ;                       that did not match)
0000    ;               R` = Address of nonmatching byte in S1
0000    ;               R2 = R0
0000    ;               R3 = Address of nonmatching byte in S2
0000    ;
0000    ; Condition Codes:
0000    ;
0000    ;       In general, the condition codes reflect whether or not the strings
0000    ;       are considered the same or different. In the case of different
0000    ;       strings, the condition codes reflect the result of the comparison
0000    ;       that indicated that the strings are not equal.
0000    ;
0000    ;       Strings are IDENTICAL
0000    ;
0000    ;               N <- 0
```

```
                 0000  ;              Z <- 1                  ; (byte in S1) EQL (byte in S2)
                 0000  ;              V <- 0
                 0000  ;              C <- 0
                 0000  ;
                 0000  ;       Strings DO NOT MATCH
                 0000  ;
                 0000  ;              N <- (byte in S1) LSS (byte in S2)
                 0000  ;              Z <- 0                  ; (byte in S1) NEQ (byte in S2)
                 0000  ;              V <- 0
                 0000  ;              C <- (byte in S1) LSSU (byte in S2)
                 0000  ;
                 0000  ;       where "byte in S1" or "byte in S2" may indicate the fill character
                 0000  ;
                 0000  ; Side Effects:
                 0000  ;
                 0000  ;       This routine uses one longword of stack.
                 0000  ;-
                 0000
                 0000  VAX$CMPC3::
50    50    3C   0000          MOVZWL  R0,R0                  ; Clear unused bits & check for zero
      0D    13   0003          BEQL    20$                    ; Simply return if zero length string
                 0005
      5A    DD   0005          PUSHL   R10                    ; Save R10 so it can hold handler
                 0007          ESTABLISH_HANDLER      -
                 0007                  STRING_ACCVIO          ; Store address of condition handler
                 0007
                 0007          MARK_POINT      CMPC3_1
81    83    91   0007  10$:    CMPB    (R3)+,(R1)+            ; Character match?
      0B    12   000A          BNEQ    30$                    ; Exit loop if different
   F8 50    F5   000C          SOBGTR  R0,10$
                 000F
                 000F  ; Exit path for strings IDENTICAL (R0 = 0, either on input or after loop)
                 000F
5A    8E    D0   000F          MOVL    (SP)+,R10              ; Restore saved R10
      52    D4   0012  20$:    CLRL    R2                     ; Set R2 for output value of 0
      50    D5   0014          TSTL    R0                     ; Set condition codes
            05   0016          RSB                            ; Return point for IDENTICAL strings
                 0017
                 0017  ; Exit path when strings DO NOT MATCH
                 0017
5A    8E    D0   0017  30$:    MOVL    (SP)+,R10              ; Restore saved R10
52    50    D0   001A          MOVL    R0,R2                  ; R0 and R2 are the same on exit
73    71    91   001D          CMPB    -(R1),-(R3)            ; Reset R1 and R3 and set condition codes
            05   0020          RSB                            ; Return point when strings DO NOT MATCH
```

```
0021            .SUBTITLE       VAX$CMPC5 - Compare Characters (5 Operand)
0021    ;+
0021    ; Functional Description:
0021    ;
0021    ;       The bytes of the string 1 specified  by  the  length  1  and  address  1
0021    ;       operands  are  compared  with the bytes of the string 2 specified by the
0021    ;       length 2 and address 2 operands.  If  one  string  is  longer  than  the
0021    ;       other, the shorter string is conceptually extended to the length of the
0021    ;       longer by appending (at  higher  addresses)  bytes  equal  to  the  fill
0021    ;       operand.  Comparison  proceeds  until inequality is detected or all the
0021    ;       bytes of the strings have been examined.  Condition codes  are  affected
0021    ;       by  the  result  of  the  last byte comparison.  Two zero length strings
0021    ;       compare equal (i.e.  Z is set and N, V, and C are cleared).
0021    ;
0021    ; Input Parameters:
0021    ;
0021    ;       R0<15:0>  = len         Length of first character string (called S1)
0021    ;       R0<23:16> = fill        Fill character that is used when strings have
0021    ;                                  different lengths
0021    ;       R1        = addr        Address of first character string
0021    ;       R2<15:0>  = len         Length of second character string (called S2)
0021    ;       R3        = addr        Address of second character string
0021    ;
002'    ; Intermediate State:
0021    ;
0021    ;        31               23               15              07              00
0021    ;       +----------------+----------------+----------------+----------------+
0021    ;       :    delta-PC    :      fill      :             src1len             : : R0
0021    ;       +----------------+----------------+----------------+----------------+
0021    ;       :                             src1addr                              : : R1
0021    ;       +----------------+----------------+----------------+----------------+
0021    ;       :              XXXXX              :             src2len             : : R2
0021    ;       +----------------+----------------+----------------+----------------+
0021    ;       :                             src2addr                              : : R3
0021    ;       +----------------+----------------+----------------+----------------+
0021    ;
0021    ; Output Parameters:
0021    ;
0021    ;       Strings are IDENTICAL
0021    ;
0021    ;               R0 = 0
0021    ;               R1 = Address of one byte beyond end of S1
0021    ;               R2 = 0 (same as R0)
0021    ;               R1 = Address of one byte beyond end of S2
0021    ;
0021    ;       Strings DO NOT MATCH
0021    ;
0021    ;               R0 = Number of bytes remaining in S1 when mismatch detected
0021    ;                       (or zero if S1 exhausted before mismatch detected)
0021    ;               R1 = Address of nonmatching byte in S1
0021    ;               R2 = Number of bytes remaining in S2 when mismatch detected
0021    ;                       (or zero if S2 exhausted before mismatch detected)
0021    ;               R3 = Address of nonmatching byte in S2
0021    ;
0021    ; Condition Codes:
0021    ;
0021    ;       In general, the condition codes reflect whether or not the strings
```

```
                              0021  ;       are considered the same or different. In the case of different
                              0021  ;       strings, the condition codes reflect the result of the comparison
                              0021  ;       that indicated that the strings are not equal.
                              0021  ;
                              0021  ;       Strings are IDENTICAL
                              0021  ;
                              0021  ;               N <- 0
                              0021  ;               Z <- 1                  ; (byte in S1) EQL (byte in S2)
                              0021  ;               V <- 0
                              0021  ;               C <- 0
                              0021  ;
                              0021  ;       Strings DO NOT MATCH
                              0021  ;
                              0021  ;               N <- (byte in S1) LSS (byte in S2)
                              0021  ;               Z <- 0                  ; (byte in S1) NEQ (byte in S2)
                              0021  ;               V <- 0
                              0021  ;               C <- (byte in S1) LSSU (byte in S2)
                              0021  ;
                              0021  ;       where "byte in S1" or "byte in S2" may indicate the fill character
                              0021  ;
                              0021  ; Side Effects:
                              0021  ;
                              0021  ;       This routine uses two longwords of stack.
                              0021  ;-
                              0021
                              0021          .ENABLE LOCAL_BLOCK
                              0021
                              0021  VAX$CMPC5::
                     5A   DD  0021          PUSHL   R10                     ; Save R10 so it can hold handler
                              0023          ESTABLISH_HANDLER       -
                              0023                  STRING_ACCVIO           ; Store address of condition handler
                     54   DD  0023          PUSHL   R4                      ; Save register
      54    50    F0 8F   78  0025          ASHL    #-16,R0,R4              ; Get escape character
              50     50   3C  002A          MOVZWL  R0,R0                   ; Clear unused bits & is S1 length zero?
                     28   13  002D          BEQL    50$                     ; Branch if yes
              52     52   3C  002F          MOVZWL  R2,R2                   ; Clear unused bits & is S2 length zero?
                     14   13  0032          BEQL    30$
                              0034
                              0034  ; Main loop. The following loop executes when both strings have characters
                              0034  ; remaining and inequality has not yet been detected.
                              0034
                              0034  ; THE FOLLOWING LOOP IS A TARGET FOR FURTHER OPTIMIZATION IN THAT THE
                              0034  ; LOOP SHOULD NOT REQUIRE TWO SOBGTR INSTRUCTIONS. NOTE, THOUGH, THAT
                              0034  ; THE CURRENT UNOPTIMIZED LOOP IS EASIER TO BACK UP.
                              0034
                              0034          MARK_POINT      CMPC5_1
              83     81   91  0034  10$:    CMPB    (R1)+,(R3)+             ; Characters match?
                     32   12  0037          BNEQ    80$                     ; Exit loop if bytes different
                  09 50   F5  0039          SOBGTR  R0,20$                  ; Check for S1 exhausted
                              003C
                              003C  ; The next test determines whether S2 is also exhausted.
                              003C
                     52   D7  003C          DECL    R2                      ; Put R2 in step with R0
                     1C   12  003E          BNEQ    60$                     ; Branch if bytes remaining in S2
                              0040
                              0040  ; This is the exit path for identical strings. If we get here, then both
                              0040  ; R0 and R2 are zero.  The condition codes are correctly set (by the ASHL
                   0040   ; instruction) so the registers are restored with a POPR to avoid changing
                   0040   ; the condition codes.
                   0040
                   0040   IDENTICAL:
0410 8F        BA  0040          POPR    #^M<R4,R10>              ; Re,tore saved registers
               05  0044          RSB                              ; Exit indicating IDENTICAL strings
                   0045
  EC 52        F5  0045   20$:   SOBGTR  R2,10$                   ; Check for S2 exhausted
                   0048
                   0048   ; The following loop is entered when all of S2 has been processed but
                   0048   ; there are characters remaining in S1. In other words,
                   0048   ;
                   0048   ;      R0 GTRU 0
                   0048   ;      R2 EQL 0
                   0048   ;
                   0048   ; The remaining characters in S1 are compared to the fill character.
                   0048
                   0048          MARK_POINT      CMPC5_2
54    81       91  0048   30$:   CMPB    (R1)+,R4                 ; Characters match?
      05       12  004B          BNEQ    40$                      ; Exit loop if no match
  F8 50        F5  004D          SOBGTR  R0,30$                   ; Any more bytes in S1?
                   0050
      EE       11  0050          BRB     IDENTICAL                ; Exit indicating IDENTICAL strings
                   0052
54    71       91  0052   40$:   CMPB    -(R1),R4                 ; Reset R1 and set condition codes
      17       11  0055          BRB     NO_MATCH                 ; Exit indicating strings DO NOT MATCH
                   0057
                   0057   ; The following code executes if S1 has zero length on input. If S2 also
                   0057   ; has zero length, the routine smply returns, indicating equal strings.
                   0057
52    52       3C  0057   50$:   MOVZWL  R2,R2                    ; Clear unused bits. Is S2 len also zero?
      E4       13  005A          BEQL    IDENTICAL                ; Exit indicating IDENTICAL strings
                   005C
                   005C   ; The following loop is entered when all of S1 has been processed but
                   005C   ; there are characters remaining in S2. In other words,
                   005C   ;
                   005C   ;      R0 EQL 0
                   005C   ;      R2 GTRU 0
                   005C   ;
                   005C   ; The remaining characters in S2 are compared to the fill character.
                   005C
                   005C          MARK_POINT      CMPC5_3
83    54       91  005C   60$:   CMPB    R4,(R3)+                 ; Characters match?
      05       12  005F          BNEQ    70$                      ; Exit loop if no match
  F8 52        F5  0061          SOBGTR  R2,60$                   ; Any more bytes in S2?
                   0064
      DA       11  0064          BRB     IDENTICAL                ; Exit indicating IDENTICAL strings
                   0066
73    54       91  0066   70$:   CMPB    R4,-(R3)                 ; Reset R3 and set condition codes
      03       11  0069          BRB     NO_MATCH                 ; Exit indicating strings DO NOT MATCH
                   006B
                   006B   ; The following exit path is taken if both strings have characters
                   006B   ; remaining and a character pair that did not match was detected.
                   006B
73    71       91  006B   80$:   CMPB    -(R1),-(R3)              ; Reset R1 and R3 and set condition codes
                   006E   NO_MATCH:                               ; Restore R4 and R10
0410 8F        BA  006E          POPR    #^M<R4,R10>              ;  without changing condition codes
```

```
05  0072    958         RSB                                     ; Exit indicating strings DO NOT MATCH
    0073    959
    0073    960         .DISABLE        LOCAL_BLOCK
```

```
            0073          .SUBTITLE       VAX$LOCC - Locate Character
            0073    ;+
            0073    ; Functional Description:
            0073    ;
            0073    ;       The character operand is compared with the bytes of the string specified
            0073    ;       by the length and address operands.  Comparison continues until equality
            0073    ;       is detected or all bytes of the string have been compared.  If  equality
            0073    ;       is  detected;  the condition code Z-bit is cleared;  otherwise the Z-bit
            0073    ;       is set.
            0073    ;
            0073    ; Input Parameters:
            0073    ;
            0073    ;       R0<15:0>  = len         Length of character string
            0073    ;       R0<23:16> = char        Character to be located
            0073    ;       R1        = addr        Address of character string
            0073    ;
            0073    ; Intermediate State:
            0073    ;
            0073    ;    31              23              15              07             00
            0073    ;   +---------------+---------------+---------------+---------------+
            0073    ;   |   delta-PC    |     char      |              len              | : R0
            0073    ;   +---------------+---------------+---------------+---------------+
            0073    ;   |                             addr                              | : R1
            0073    ;   +---------------+---------------+---------------+---------------+
            0073    ;
            0073    ; Output Parameters:
            0073    ;
            0073    ;       Character Found
            0073    ;
            0073    ;               R0 = Number of bytes remaining in the string (including located one)
            0073    ;               R1 = Address of the located byte
            0073    ;
            0073    ;       Character NOT Found
            0073    ;
            0073    ;               R0 = 0
            0073    ;               R1 = Address of one byte beyond end of string
            0073    ;
            0073    ; Condition Codes:
            0073    ;
            0073    ;       N <- 0
            0073    ;       Z <- R0 EQL 0
            0073    ;       V <- 0
            0073    ;       C <- 0
            0073    ;
            0073    ;       The Z bit is clear if the character is located.
            0073    ;       The Z bit is set if the character is NOT located.
            0073    ;
            0073    ; Side Effects:
            0073    ;
            0073    ;       This routine uses two longwords of stack
            0073    ;-
            0073
            0073    VAX$LOCC::
 5A   DD    0073            PUSHL   R10                     ; Save R10 so it can hold handler
            0075            ESTABLISH_HANDLER       -
            0075                    STRING_ACCVIO           ; Store address of condition handler
 52   DD    0075            PUSHL   R2                      ; Save register
 52   50   F0 8F   78  0077  1209          ASHL    #-16,R0,R2                 ; Get character to be located
           50 50   3C  007C  1210          MOVZWL  R0,R0                      ; Clear unused bits & check for 0 length
              08   13  007F  1211          BEQL    20$                        ; Simply return if length is 0
                       0081  1212
                       0081  1213          MARK_POINT      LOCC_1
           81 52   91  0081  1214 10$:     CMPB    R2,(R1)+                   ; Character match?
              0A   13  0084  1215          BEQL    30$                        ; Exit loop if yes
           F8 50   F5  0086  1216          SOBGTR  R0,10$
                       0089  1217
                       0089  1218 ; If we drop through the end of the loop into the following code, then
                       0089  1219 ; the input string was exhausted with the character NOT found.
                       0089  1220
         0404 8F   BA  0089  1221 20$:     POPR    #^M<R2,R10>                ; Restore saved R2 and R10
              50   D5  008D  1222          TSTL    R0                         ; Insure that C-bit is clear
                   05  008F  1223          RSB                                ; Return with Z-bit set
                       0090  1224
                       0090  1225 ; Exit path when character located
                       0090  1226
              51   D7  0090  1227 30$:     DECL    R1                         ; Point R1 to located character
              F5   11  0092  1228          BRB     20$                        ; Join common code
```

```
0094  2168          END_MARK_POINT
0094  2169
0094  2170          .END
```

```
BOOT SWITCH                      = 00000001                     OP$_CVTLG                        = 00004EFD
IDENTICAL                          00000040 R      02           OP$_CVTLH                        = 00006EFD
NO MATCH                           0000006E R      02           OP$_CVTLP                        = 000000F9
OP$_ACBD                         = 0000006F                     OP$_CVTPL                        = 00000036
OP$_ACBF                         = 0000004F                     OP$_CVTPS                        = 00000008
OP$_ACBG                         = 00004FFD                     OP$_CVTPT                        = 00000024
OP$_ACBH                         = 00006FFD                     OP$_CVTRDL                       = 0000006B
OP$_ADDD2                        = 00000060                     OP$_CVTRFL                       = 0000004B
OP$_ADDD3                        = 00000061                     OP$_CVTRGL                       = 00004BFD
OP$_ADDF2                        = 00000040                     OP$_CVTRHL                       = 00006BFD
OP$_ADDF3                        = 00000041                     OP$_CVTSP                        = 00000009
OP$_ADDG2                        = 000040FD                     OP$_CVTTP                        = 00000026
OP$_ADDG3                        = 000041FD                     OP$_CVTWD                        = 0000006D
OP$_ADDH2                        = 000060FD                     OP$_CVTWF                        = 0000004D
OP$_ADDH3                        = 000061FD                     OP$_CVTWG                        = 00004DFD
OP$_ADDP4                        = 00000020                     OP$_CVTWH                        = 00006DFD
OP$_ADDP6                        = 00000021                     OP$_DIVD2                        = 00000066
OP$_ASHP                         = 000000F8                     OP$_DIVD3                        = 00000067
OP$_CLRD                         = 0000007C                     OP$_DIVF2                        = 00000046
OP$_CLRF                         = 000000D4                     OP$_DIVF3                        = 00000047
OP$_CLRG                         = 0000007C                     OP$_DIVG2                        = 000046FD
OP$_CLRH                         = 00007CFD                     OP$_DIVG3                        = 000047FD
OP$_CMPD                         = 00000071                     OP$_DIVH2                        = 000066FD
OP$_CMPF                         = 00000051                     OP$_DIVH3                        = 000067FD
OP$_CMPG                         = 000051FD                     OP$_DIVP                         = 00000027
OP$_CMPH                         = 000071FD                     OP$_EDITPC                       = 00000038
OP$_CMPP3                        = 00000035                     OP$_EMODD                        = 00000074
OP$_CMPP4                        = 00000037                     OP$_EMODF                        = 00000054
OP$_CRC                          = 0000000B                     OP$_EMODG                        = 000054FD
OP$_CVTBD                        = 0000006C                     OP$_EMODH                        = 000074FD
OP$_CVTBF                        = 0000004C                     OP$_MATCHC                       = 00000039
OP$_CVTBG                        = 00004CFD                     OP$_MNEGD                        = 00000072
OP$_CVTBH                        = 00006CFD                     OP$_MNEGF                        = 00000052
OP$_CVTDB                        = 00000068                     OP$_MNEGG                        = 000052FD
OP$_CVTDF                        = 00000076                     OP$_MNEGH                        = 000072FD
OP$_CVTDH                        = 000032FD                     OP$_MOVD                         = 00000070
OP$_CVTDL                        = 0000006A                     OP$_MOVF                         = 00000050
OP$_CVTDW                        = 00000069                     OP$_MOVG                         = 000050FD
OP$_CVTFB                        = 00000048                     OP$_MOVH                         = 000070FD
OP$_CVTFD                        = 00000056                     OP$_MOVP                         = 00000034
OP$_CVTFG                        = 000099FD                     OP$_MOVTC                        = 0000002E
OP$_CVTFH                        = 000098FD                     OP$_MOVTUC                       = 0000002F
OP$_CVTFL                        = 0000004A                     OP$_MULD2                        = 00000064
OP$_CVTFW                        = 00000049                     OP$_MULD3                        = 00000065
OP$_CVTGB                        = 000048FD                     OP$_MULF2                        = 00000044
OP$_CVTGF                        = 000033FD                     OP$_MULF3                        = 00000045
OP$_CVTGH                        = 000056FD                     OP$_MULG2                        = 000044FD
OP$_CVTGL                        = 00004AFD                     OP$_MULG3                        = 000045FD
OP$_CVTGW                        = 000049FD                     OP$_MULH2                        = 000064FD
OP$_CVTHB                        = 000068FD                     OP$_MULH3                        = 000065FD
OP$_CVTHD                        = 0000F7FD                     OP$_MULP                         = 00000025
OP$_CVTHF                        = 0000F6FD                     OP$_POLYD                        = 00000075
OP$_CVTHG                        = 000076FD                     OP$_POLYF                        = 00000055
OP$_CVTHL                        = 00006AFD                     OP$_POLYG                        = 000055FD
OP$_CVTHW                        = 000069FD                     OP$_POLYH                        = 000075FD
OP$_CVTLD                        = 0000006E                     OP$_SCANC                        = 0000002A
OP$_CVTLF                        = 0000004E                     OP$_SKPC                         = 0000003B
```

```
OP$_SPANC                  = 0000002B
OP$_SUBD2                  = 00000062
OP$_SUBD3                  = 00000063
OP$_SUBF2                  = 00000042
OP$_SUBF3                  = 00000043
OP$_SUBG2                  = 000042FD
OP$_SUBG3                  = 000043FD
OP$_SUBH2                  = 000062FD
OP$_SUBH3                  = 000063FD
OP$_SUBP4                  = 00000022
OP$_SUBP6                  = 00000023
OP$_TSTD                   = 00000073
OP$_TSTF                   = 00000053
OP$_TSTG                   = 000053FD
OP$_TSTH                   = 000073FD
VAX$CMPC3                    00000000 RG    02
VAX$CMPC5                    00000021 RG    02
VAX$LOCC                     00000073 RG    02

                                         +----------------+
                                         ! Psect synopsis !
                                         +----------------+

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                       00000000  (     0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
_VAX$CODE                   00000094  (   148.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG

                                      +------------------------+
                                      ! Performance indicators !
                                      +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   15    00:00:00.06    00:00:01.22
Command processing               74    00:00:00.73    00:00:05.99
Pass 1                          390    00:00:11.56    00:00:41.58
Symbol table sort                 0    00:00:00.58    00:00:01.86
Pass 2                          102    00:00:05.40    00:00:15.24
Symbol table output              16    00:00:00.11    00:00:00.40
Psect synopsis output             2    00:00:00.01    00:00:00.02
Cross-reference output            0    00:00:00.00    00:00:00.00
Assembler run totals            599    00:00:18.45    00:01:06.31

The working set limit was 1500 pages.
70465 bytes (138 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 447 non-local and 14 local symbols.
4923 source lines were read in Pass 1, producing 13 object records in Pass 2.
145 pages of virtual memory were used to define 143 macros.
```

```
                                             +--------------------------+
                                             ! Macro library statistics !
                                             +--------------------------+

Macro library name                             Macros defined
------------------                             --------------
_$255$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1                8
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      5
TOTALS (all libraries)                                13

584 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:BOOSTRING/OBJ=OBJ$:BOOSTRING MSRC$:BOOTSWT/UPDATE=(ENH$:BOOTSWT)+MSRC$:MISSING/UPDATE (ENH$:MISSING)+MSRC$:VAXSTRING/
```

0142 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XLATE
LIS

VAXEMUL
MAP

WIOHNDLR
LIS

WORKIO
LIS

BOOSTRING
LIS

EMULAT

FPEMUL
MAP

BO0EMULAT
LIS

WORDWRAP
LIS

VAXMACROS
MAR

WRIEDTMSG
LIS

FPEMULATE
LIS

VAXREGDEF
SDL