```
EEEEEEEEEEEEEEEE  MMM       MMM  UUU       UUU  LLL                   AAAAAAAAA    TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE  MMM       MMM  UUU       UUU  LLL                   AAAAAAAAA    TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE  MMM       MMM  UUU       UUU  LLL                   AAAAAAAAA    TTTTTTTTTTTTTTTT
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
EEEEEEEEEEEEEEEE  MMM       MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLLL  AAA       AAA          TTT
EEEEEEEEEEEEEEEE  MMM       MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLLL  AAA       AAA          TTT
EEEEEEEEEEEEEEEE  MMM       MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLLL  AAA       AAA          TTT
```

```
VV       VV   AAAAAA   XX      XX  EEEEEEEEEE  MM      MM  UU      UU  LL               AAAAAA    TTTTTTTTTT
VV       VV   AAAAAA   XX      XX  EEEEEEEEEE  MM      MM  UU      UU  LL               AAAAAA    TTTTTTTTTT
VV       VV   AA    AA  XX    XX   EE          MMMM  MMMM  UU      UU  LL             AA      AA       TT
VV       VV   AA    AA  XX    XX   EE          MMMM  MMMM  UU      UU  LL             AA      AA       TT
VV       VV   AA    AA   XX  XX    EE          MM  MM  MM  UU      UU  LL             AA      AA       TT
VV       VV   AA    AA    XX XX    EE          MM  MM  MM  UU      UU  LL             AA      AA       TT
VV       VV   AA    AA     XX      EEEEEEE     MM      MM  UU      UU  LL             AA      AA       TT
VV       VV   AA    AA     XX      EEEEEEE     MM      MM  UU      UU  LL             AA      AA       TT
VV       VV   AAAAAAAAAA  XX XX    EE          MM      MM  UU      UU  LL             AAAAAAAAAA       TT
VV       VV   AAAAAAAAAA  XX  XX   EE          MM      MM  UU      UU  LL             AAAAAAAAAA       TT
  VV  VV     AA      AA  XX    XX   EE          MM      MM  UU      UU  LL             AA      AA       TT      ....
  VV  VV     AA      AA  XX    XX   EE          MM      MM  UU      UU  LL             AA      AA       TT      ....
    VV       AA      AA  XX      XX  EEEEEEEEEE  MM      MM  UUUUUUUUUU  LLLLLLLLLL  AA      AA       TT      ....
    VV       AA      AA  XX      XX  EEEEEEEEEE  MM      MM  UUUUUUUUUU  LLLLLLLLLL  AA      AA       TT      ....

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

I 11

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page   0   VAX
Table of contents   VAX
                                                                                                              V04

```
0000        1                  .NOSHOW CONDITIONALS
0000        3                  .TITLE  VAXSEMULATE - VAX-11 Instruction Emulator
0000        7                  .IDENT  /V04-000/
0000        8
0000        9          ;
0000       10          ;*********************************************************************
0000       11          ;*                                                                   *
0000       12          ;*    COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                        *
0000       13          ;*    DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.         *
0000       14          ;*    ALL RIGHTS RESERVED.                                           *
0000       15          ;*                                                                   *
0000       16          ;*    THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000       17          ;*    ONLY IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000       18          ;*    INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000       19          ;*    COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000       20          ;*    OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000       21          ;*    TRANSFERRED.                                                    *
0000       22          ;*                                                                   *
0000       23          ;*    THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000       24          ;*    AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000       25          ;*    CORPORATION.                                                    *
0000       26          ;*                                                                   *
0000       27          ;*    DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000       28          ;*    SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.         *
0000       29          ;*                                                                   *
0000       30          ;*                                                                   *
0000       31          ;*********************************************************************
0000       32
0000       33
0000       34          ;++
0000       35          ; Facility:
0000       36          ;
0000       37          ;       VAX-11 Instruction Emulator
0000       38          ;
0000       39          ; Abstract:
0000       40          ;
0000       41          ;       This is the main body of the instruction emulator that supports
0000       42          ;       the instructions that are not a part of the microVAX architecture.
0000       43          ;       The current design calls for support of the string instructions
0000       44          ;       (including CRC), the decimal instructions, and EDITPC.
0000       45          ;
0000       46          ;       Th s routine performs the following steps.
0000       47          ;
0000       48          ;       o Moves operands from the exception stack to registers in an
0000       49          ;         instruction-specific manner
0000       50          ;
0000       51          ;       o Calls an instruction-specific subroutine to do the actual work
0000       52          ;
0000       53          ;       If errors occur along the way, those errors are reflected to the
0000       54          ;       user as exceptions.
0000       55          ;
0000       56          ; Environment:
0000       57          ;
0000       58          ;       These routines run at any access mode, at any IPL, and are AST
0000       59          ;       reentrant. The routine starts execution in the access mode and
0000       60          ;       at the IPL at which the instruction executed.
0000       61          ;
```

```
0000      62 ; Author:
0000      63 ;
0000      64 ;       Lawrence J. Kenah
0000      65 ;
0000      66 ; Creation Date
0000      67 ;
0000      68 ;       17 August 1982
0000      69 ;
0000      70 ; Modified by:
0000      71 ;
0000      72 ;       V01-011 LJK0041          Lawrence J. Kenah          16-Jul-1984
0000      73 ;               Clear FPD in saved PSL at VAX$EMULATE_FPD entry so that
0000      74 ;               next instruction can execute correctly.
0000      75 ;
0000      76 ;       V01-010 LJK0031          Lawrence J. Kenah          5-Jul-1984
0000      77 ;               Set R2 and R4 unconditionally to zero in EDITPC routine
0000      78 ;               to allow the storage of FPD flags and similar data.
0000      79 ;
0000      80 ;       V01-009 LJK0026          Lawrence J. Kenah          19-Mar-1984
0000      81 ;               Perform final cleanup pass. Eliminate xxx_UNPACK routine
0000      82 ;               references. Add C-bit optimization to MOVP.
0000      83 ;
0000      84 ;       V01-008 LJK0010          Lawrence J. Kenah          8-Nov-1983
0000      85 ;               Eliminate code in EXIT_EMULATOR path that unconditionally
0000      86 ;               clears the T-bit and conditionally sets the TP-bit. The
0000      87 ;               TP-bit is handled by the base hardware.
0000      88 ;
0000      89 ;       V01-007 KDM0088          Kathleen D. Morse          20-Oct-1983
0000      90 ;               Make branches to VAX$REFLECT_TO_VMS into jumps, so that
0000      91 ;               the bootstrap emulator will link without truncation errors
0000      92 ;               until that routine is finished.
0000      93 ;
0000      94 ;       V01-006 KDM0003          Kathleen D. Morse          18-Apr-1983
0000      95 ;               Generate abbreviated VAX$EMULATE_FPD for the bootstrap
0000      96 ;               emulator.
0000      97 ;
0000      98 ;       V01-005 LJK0006          Lawrence J. Kenah          16-Mar-1983
0000      99 ;               Generate case tables with macros. Allow subset emulator
0000     100 ;               for bootstrap instruction emulation.
0000     101 ;
0000     102 ;       V01-004 KDM0002          Kathleen D. Morse          16-Mar-1983
0000     103 ;               Fix fourth and fifth operand fetches for SUBP6, ADDP6,
0000     104 ;               MULP and DIVP.
0000     105 ;
0000     106 ;       V01-003 KDM0001          Kathleen D. Morse          04-Mar-1983
0000     107 ;               Longword align the exception handler entry points.
0000     108 ;
0000     109 ;       V01-002 LJK0005          Lawrence J. Kenah          15-Nov-1982
0000     110 ;               Use hardware aids provided by microVAX architecture revision.
0000     111 ;               Exception is now reported in caller's mode. Operands are parsed
0000     112 ;               and placed on the exception stack as exception parameters.
0000     113 ;
0000     114 ;       V01-001 LJK0002          Lawrence J. Kenah          17-Aug-1982
0000     115 ;               Original version using kernel mode exception through OPCDEC
0000     116 ;               exception vector.
0000     117 ;--
```

L 11

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page   3
V04-000   DECLARATIONS   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (2)

```
0000   119            .SUBTITLE       DECLARATIONS
0000   120
0000   121  ; Include files:
0000   122
0000   123            $OPDEF                                  ; Values for instruction opcodes
0000   124            $PSLDEF                                 ; Define bit fields in PSL
0000   125
0000   126            .NOCROSS                                ; No cross reference for these
0000   127            .ENABLE         SUPPRESSION             ; No symbol table entries either
0000   128
0000   129            PACK_DEF                                ; Stack usage when restarting instructions
0000   130            STACK_DEF                               ; Stack usage for original exception
0000   131
0000   132            .DISABLE        SUPPRESSION             ; Turn on symbol table again
0000   133            .CROSS                                  ; Cross reference is OK now
0000   134
0000   135  ; Macro definitions
0000   136
0000   137            .MACRO    INIT_CASE_TABLE       SIZE,BASE,ERROR_EXIT
0000   138  BASE:
0000   139            .REPT    SIZE
0000   140            .WORD    ERROR_EXIT-BASE
0000   141            .ENDR
0000   142            .ENDM    INIT_CASE_TABLE
0000   143
0000   144            .MACRO    CASE_TABLE_ENTRY          OPCODE,-
0000   145                                               ROUTINE,-
0000   146                                               FPD_ROUTINE,-
0000   147                                               BOOT_FLAG
0000   148            SIGN_EXTEND      OP$ 'OPCODE , ...OPCODE
0000   149            ...OFFSET = ...OPCODE - OPCODE_BASE
0000   150            .IF      NOT_DEFINED     BOOT_SWITCH
0000   151                     INCLUDE_'OPCODE = 0
0000   152            .EXTERNAL        VAX$'OPCODE
0000   153            .EXTERNAL        FPD_ROUTINE
0000   154            . = CASE_TABLE_BASE + <2 * ...OFFSET>
0000   155            .WORD    ROUTINE - CASE_TABLE_BASE
0000   156            . = FPD_CASE_TABLE_BASE + <2 * ...OFFSET>
0000   157            .WORD    FPD_ROUTINE - FPD_CASE_TABLE_BASE
0000   158            .IF_FALSE
0000   159                     .IF      IDENTICAL       <BOOT_FLAG>,BOOT
0000   160                     INCLUDE_'OPCODE = 0
0000   161            .EXTERNAL        VAX$'OPCODE
0000   162            . = CASE_TABLE_BASE + <2 * ...OFFSET>
0000   163            .WORD    ROUTINE - CASE_TABLE_BASE
0000   164            .ENDC
0000   165            .ENDC
0000   166            .ENDM    CASE_TABLE_ENTRY
0000   167
0000   168  ; External declarations for exception handling
0000   169
0000   170            .DISABLE        GLOBAL
0000   171
0000   173            .EXTERNAL       VAX$AL_DELTA_PC_TABLE
0000   174            .EXTERNAL       VAX$REFLECT_TO_VMS
0000   176
0000   177            .EXTERNAL       VAX$_OPCDEC, -
```

```
                0000    178                             VAX$_OPCDEC_FPD
                0000    179
                0000    180  ; PSECT Declarations:
                0000    181
                0000    182          .DEFAULT        DISPLACEMENT , WORD
                0000    183
          00000000    184          .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, QUAD
                0000    185
                0000    186          .NOSHOW         CONDITIONALS
```

VAX$EMULATE
V04-000

N 11

- VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 5
VAX$EMULATE - Entry Path into Emulator   5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (3)

VAX
V04

```
0000   188          .SUBTITLE     VAX$EMULATE - Entry Path into Emulator
0000   189  ;+
0000   190  ; Functional Description:
0000   191  ;
0000   192  ;        There are two different entries into this module. When a reserved
0000   193  ;        instruction is first encountered, its operands are parsed by the
0000   194  ;        hardware (or microcode, if you will) and placed on the stack as
0000   195  ;        exception parameters. The code at address VAX$EMULATE is then entered
0000   196  ;        through the ^XC8(SCB) exception vector. That routine dispatches to an
0000   197  ;        instruction-specific routine called VAX$xxxxxx (xxxxxx represents the
0000   198  ;        name of the reserved instruction) after placing the operands into
0000   199  ;        registers as required by VAX$xxxxxx.
0000   200  ;
0000   201  ;        If an exception occurred during instruction emulation such that a
0000   202  ;        reserved instruction executed again, this time with FPD set, then a
0000   203  ;        different exception path is taken. The stack has a different (smaller)
0000   204  ;        set of parameters for the FPD exception. A different
0000   205  ;        instruction-specific routine executes to unpack saved intermediate
0000   206  ;        state before resuming instruction emulation.
0000   207  ;
0000   208  ;        The access mode and IPL are preserved across either exception.
0000   209  ;
0000   210  ; Input Parameters:
0000   211  ;
0000   212  ;        00(SP) - Opcode of reserved instruction
0000   213  ;        04(SP) - PC of reserved instruction (old PC)
0000   214  ;        08(SP) - First operand specifier
0000   215  ;        12(SP) - Second operand specifier
0000   216  ;        16(SP) - Third operand specifier
0000   217  ;        20(SP) - Fourth operand specifier
0000   218  ;        24(SP) - Fifth operand specifier
0000   219  ;        28(SP) - Sixth operand specifier
0000   220  ;        32(SP) - Seventh operand specifier (currently unused)
0000   221  ;        36(SP) - Eight operand specifier (currently unused)
0000   222  ;        40(SP) - PC of instruction following reserved instruction (new PC)
0000   223  ;        44(SP) - PSL at time of exception
0000   224  ;
0000   225  ; Notes on input parameters:
0000   226  ;
0000   227  ;   1.   The information that appears on the stack for each operand depends
0000   228  ;        on the nature of the operand.
0000   229  ;
0000   230  ;        .rx - Operand value
0000   231  ;        .ax - Operand address
0000   232  ;        .wx - Operand address (Register destination is stored in one's
0000   233  ;              complement form. See VAX$CVTPL for details.)
0000   234  ;
0000   235  ;   2.   The old PC value is not used unless an exception such as an access
0000   236  ;        violation occurs and the instruction has to be backed up.
0000   237  ;
0000   238  ;   3.   The seventh and eighth operands are not used for any existing VAX-11
0000   239  ;        instructions. Those slots in the exception stach frame are reserved
0000   240  ;        for future expansion.
0000   241  ;
0000   242  ;   4.   The two PC parameters and the PSL are the only data that needs to
0000   243  ;        be preserved once the instruction-specific routine is entered.
00C0   244  ;
```

B 12

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page   6   VAX
V04-000   VAX$EMULATE - Entry Path into Emulator   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (3)   V04

```
                              0000   245  ; Output Parameters:
                              0000   246  ;
                              0000   247  ;      The operands are moved from the stack to general registers in a way
                              0000   248  ;      that varies from instruction to instruction. Control is transferred
                              0000   249  ;      to a specific routine for each opcode.
                              0000   250  ;
                              0000   251  ; Notes:
                              0000   252  ;
                              0000   253  ;      There are several tables in the emulator that use the opcode as an
                              0000   254  ;      index. We choose to interpret the opcode as a signed quantity because
                              0000   255  ;      this reduces the amount of wasted space in the tables. In either case,
                              0000   256  ;      there are 27 useful entries.
                              0000   257  ;
                              0000   258  ;      Unsigned opcode
                              0000   259  ;
                              0000   260  ;           OPCODE_BASE = CVTPS (value of 8)
                              0000   261  ;           OPCODE_MAX = CVTLP (value of F9)
                              0000   262  ;
                              0000   263  ;           TABLE_SIZE = 241 decimal bytes
                              0000   264  ;
                              0000   265  ;      Signed opcode
                              0000   266  ;
                              0000   267  ;           OPCODE_BASE = ASHP (value of F8 or -8)
                              0000   268  ;           OPCODE_MAX = SKPC (value of 3B)
                              0000   269  ;
                              0000   270  ;           TABLE_SIZE = 67 decimal bytes
                              0000   271  ;
                              0000   272  ;      The savings of more than 170 entries in each table justifies all
                              0000   273  ;      of the machinations that we go through to treat opcodes as signed
                              0000   274  ;      quantities.
                              0000   275  ;-
                              0000   276
                              0000   277  ; Because the assembler does not understand sign extension of byte and
                              0000   278  ; word quantities, we must accomplish this sign extension with macros. The
                              0000   279  ; assignment statements that appear as comments illustrate the sense of the
                              0000   280  ; macro invocations that immediately follow.
                              0000   281
                              0000   282  ;      OPCODE_MAX = OP$_SKPC              ; Largest opcode in this emulator
                              0000   283
                              0000   284         SIGN_EXTEND     OP$_SKPC , OPCODE_MAX
                              0000   285
                              0000   286  ; We further restrict the table size and supported operations when we are
                              0000   287  ; building the bootstrap subset of the emulator. We only allow certain string
                              0000   288  ; instructions to contribute to the emulator.
                              0000   289
                              0000   295  ;      OPCODE_BASE = OP$_ASHP             ; Smallest (in signed sense) opcode
                              0000   296
                              0000   297         SIGN_EXTEND     OP$_ASHP , OPCODE_BASE
                              0000   299
                   00000044   0000   300  CASE_TABLE_SIZE = <OPCODE_MAX - OPCODE_BASE> + 1          ; Define table size
                              0000   301
                              0000   302         .ALIGN  LONG                      ; Alignment for exception vector
                              0000   303
                              0000   304  VAX$EMULATE::
                              0000   305
   43 8F   F8 8F   6E   8F    0000   306         CASEB   OPCODE(SP),#OPCODE_BASE,#<OPCODE_MAX-OPCODE_BASE>
                              0006   307
```

VAXSEMULATE   C 12
V04-000
- VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page   7
VAXSEMULATE - Entry Path into Emulator   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (3)

```
                        0006    308             INIT_CASE_TABLE CASE_TABLE_SIZE,CASE_TABLE_BASE,10$
                        008E    309
                        008E    310 ; If we drop through the case dispatcher, then the fault was not caused
                        008E    311 ; by executing one of the instructions supported by this emulator. Such
                        008E    312 ; exceptions will simply be passed through to VMS. (In the bootstrap emulator,
                        008E    313 ; there is no operating system to reflect the exception. We simply HALT.)
                        008E    314
   00000000'8F  DD      008E    315 10$:    PUSHL   #VAX$_OPCDEC            ; Store signal name
            0D  DD      0094    316         PUSHL   #13                    ; Total of 13 longwords in signal array
                        0096    317
         FF67'  31      0096    321         BRW     VAX$REFLECT_TO_VMS      ; Use common exit to VMS
```

VAX$EMULATE   – VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 8   VAX'
V04-000   VAX$EMULATE_FPD – Alternate Entry Path i 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (4)   V04·

D 12

```
0099   324          .SUBTITLE        VAX$EMULATE_FPD – Alternate Entry Path into Emulator
0099   325  ;+
0099   326  ; Functional Description:
0099   327  ;
0099   328  ;     This routine is entered through the ^XCC(SCB) exception vector when an
0099   329  ;     instruction that is not a part of the microVAX architecture executes
0099   330  ;     and the FPD bit is set in the PSL. The software state that was
0099   331  ;     preserved by each instruction must be restored and instruction
0099   332  ;     execution resumed. Access mode and IPL are preserved across the
0099   333  ;     exception occurrence.
0099   334  ;
0099   335  ;     Before the various VAX$xxxxxx (or VAX$xxxxxx_RESTART) routines regain
0099   336  ;     control, this dispatcher must retrieve the delta PC from wherever
0099   337  ;     it was stored and place the stack in the same state that it is in
0099   338  ;     when the normal (FPD bit not set) instruction dispatcher passes
0099   339  ;     control to the various VAX$xxxxxx routines. The pictures below explain
0099   340  ;     this.
0099   341  ;
0099   342  ; Input Parameters:
0099   343  ;
0099   344  ;     00(SP) – PC of reserved instruction
0099   345  ;     04(SP) – PSL at time of exception
0099   346  ;
0099   347  ; Output Parameters:
0099   348  ;
0099   349  ;     The following picture shows the state of the stack after the dispatcher
0099   350  ;     has executed its preliminary code but before control is passed back to
0099   351  ;     instruction-specific  execution. Note that this routine makes the
0099   352  ;     stack look like it does when a reserved instruction executes and FPD
0099   353  ;     is not yet set. This is done to make the exception exit code independent
0099   354  ;     of whether a different exception exception occurred while the emulator
0099   355  ;     was running.
0099   356  ;
0099   357  ;     00(SP) – Return PC (Address of EXIT routine in this module)
0099   358  ;     04(SP) – Unused placeholder (OPCODE)
0099   359  ;     08(SP) – PC of reserved instruction (old PC)
0099   360  ;     12(SP) – Unused placeholder (OPERAND_1)
0099   361  ;     16(SP) – Unused placeholder (OPERAND_2)
0099   362  ;     20(SP) – Unused placeholder (OPERAND_3)
0099   363  ;     24(SP) – Unused placeholder (OPERAND_4)
0099   364  ;     28(SP) – Unused placeholder (OPERAND_5)
0099   365  ;     32(SP) – Unused placeholder (OPERAND_6)
0099   366  ;     36(SP) – Unused placeholder (OPERAND_7)
0099   367  ;     40(SP) – Unused placeholder (OPERAND_8)
0099   368  ;     44(SP) – PC of instruction following reserved instruction (new PC)
0099   369  ;     48(SP) – PSL at time of exception
0099   370  ;
0099   371  ;     Before this routine dispatches to opcode-specific code, it calculates
0099   372  ;     the PC of the next instruction based on the PC of the reserved
0099   373  ;     instruction and the delta-PC quantity that was stored as part of the
0099   374  ;     instruction's intermediate state. Note that the delta PC quantity
0099   375  ;
0099   376  ;          delta PC = new PC - old PC
0099   377  ;
0099   378  ;     is stored in the upper bytes of one of the general registers, usually
0099   379  ;     bits <31:24> of R0 or R2. The registers R0 through R3 are stored on
0099   380  ;     the stack (in  he space used for the first four operands when the
```

E 12

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page   9
V04-000   VAXSEMULATE_FPD - Alternate Entry Path i 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (4)

VAX
V04

```
                              0099   381 ;         reserved instruction is first encountered) so that the same offsets
                              0099   382 ;         that were used to store the delta-PC can be used to retrieve it.
                              0099   383 ;-
                              0099   384
                              0099   385         .ALIGN  LONG                            ; Alignment for exception vector
                              009C   386
                              009C   387 VAXSEMULATE_FPD::
                              009C   388
                              009C   389
          00 04 AE    1B  E5  009C   391 5$:     BBCC    #PSL$V_FPD,4(SP),5$     ; Clear FPD in exception PSL
             5E     28  C2  00A1   392 5$:     SUBL2   #NEW_PC,SP             ; Create extra stack space
          04 AE    28 AE  D0  00A4   393         MOVL    NEW_PC(SP),OLD_PC(SP)  ; Make second copy of old PC
             08 AE    50  7D  00A9   394         MOVQ    R0,OPERAND_1(SP)       ; Save R0 and R1 in some extra space
             10 AE    52  7D  00AD   395         MOVQ    R2,OPERAND_3(SP)       ; Do the same for R2 and R3
             50    04 BE  98  00B1   396         CVTBL   aOLD_PC(SP),R0         ; Get opcode from instruction stream
       51  0000'CF40  9A  00B5   397         MOVZBL  VAX$AL_DELTA_PC_TABLE[R0],R1 ; Get offset to byte with delta-PC
       51    08 AE41  9A  00BB   398         MOVZBL  OPERAND_1(SP)[R1],R1   ; Get delta-PC
          28 AE    51  C0  00C0   399         ADDL    R1,NEW_PC(SP)          ; Convert old PC to new PC
             6E     50  D0  00C4   400         MOVL    R0,OPCODE(SP)          ; Store opcode in other than a register
          50    08 AE  7D  00C7   401         MOVQ    OPERAND_1(SP),R0       ; Restore R0 and R1
                              00CB   402                                         ; (R2 and R3 were not changed)
          0425'CF  9F  00CB   403         PUSHAB  VAX$EXIT_EMULATOR      ; Create return PC to make CASE like BSB
                              00CF   404
 43 8F  F8 8F    04 AE  8F  00CF   405         CASEB   <OPCODE+4>(SP),#OPCODE_BASE,#<OPCODE_MAX-OPCODE_BASE>
                              00D6   406
                              00D6   407         INIT_CASE_TABLE CASE_TABLE_SIZE,FPD_CASE_TABLE_BASE,10$
                              015E   408
                              015E   409 ; If we drop through the case dispatcher, then the fault was not caused
                              015E   410 ; by executing one of the instructions supported by this emulator. The
                              015E   411 ; exception will be passed to VMS with the following stack.
                              015E   412 ;
                              015E   413 ;         00(SP) - Signal array size (always 4)
                              015E   414 ;         04(SP) - Signal name (VAX$_OPCDEC_FPD)
                              015E   415 ;         08(SP) - Opcode that is not supported
                              015E   416 ;         12(SP) - PC of that opcode
                              015E   417 ;         16(SP) - PSL of exception
                              015E   418 ;
                              015E   419 ; (In the bootstrap emulator, we simply halt with the stack containing
                              015E   420 ; these data.)
                              015E   421
             5E    04  C0  015E   422 10$:    ADDL    #4,SP                 ; Discard return PC
       28 AE    04 AE  D0  0161   423         MOVL    OLD_PC(SP),NEW_PC(SP) ; Use PC of opcode and not new PC
       24 AE     6E  D0  0166   424         MOVL    OPCODE(SP),OPERAND_8(SP) ; Include opcode in signal array
             5E    24 AE  DE  016A   425         MOVAL   OPERAND_8(SP),SP      ; Discard rest of stack
                              016E   426
                              016E   428
       00000000'8F  DD  016E   429         PUSHL   #VAX$_OPCDEC_FPD      ; This is the signal name
             04  DD  0174   430         PUSHL   #4                    ; Signal array has four longwords
                              0176   431
          FE87'  31  0176   435         BRW     VAX$REFLECT_TO_VMS    ; Use common exit to VMS
```

```
0179    438              .SUBTITLE          Dispatch Tables
0179    439    ;+
0179    440    ; Functional Description:
0179    441    ;
0179    442    ;       The case tables for the two CASEB instructions are built with the
0179    443    ;       macros that are invoked here. Macros are used to guarantee that both
0179    444    ;       tables contain correct entries for a selected opcode at the same
0179    445    ;       offset.
0179    446    ;
0179    447    ; Assumptions:
0179    448    ;
0179    449    ;       The CASE_TABLE_ENTRY macro assumes that the names of the respective
0179    450    ;       case tables are CASE_TABLE_BASE and FPD_CASE_TABLE_BASE.
0179    451    ;
0179    452    ; Notes:
0179    453    ;
0179    454    ;       In the following lists, those FPD routines that do not have _FPD in
0179    455    ;       their names use the same JSB entry point for initial entry and after
0179    456    ;       restarting the instruction. In most of these cases, the register state
0179    457    ;       is the same for both starting and restarting. For the remaining cases,
0179    458    ;       there is not enough difference between the two cases to justify an
0179    459    ;       additional entry point. (See VAX$MOVTC for an example of this latter
0179    460    ;       situation.)
0179    461    ;
0179    462    ;       The FPD routines that include _RESTART in their names have to do a
0179    463    ;       certain amount of work to restore the intermediate state from the
0179    464    ;       canonical registers before they can resume instruction execution.
0179    465    ;-
0179    466
0179    467              .SAVE                              ; Remember current location counter
0179    468
0179    469    ; First generate table entries for the string instructions
0179    470
0179    471              CASE_TABLE_ENTRY           OPCODE=MOVTC,-
0179    472                                        ROUTINE=MOVTC,-
0179    473                                        FPD_ROUTINE=VAX$MOVTC
0144    474
0144    475              CASE_TABLE_ENTRY           OPCODE=MOVTUC,-
0144    476                                        ROUTINE=MOVTUC,-
0144    477                                        FPD_ROUTINE=VAX$MOVTUC
0146    478
0146    479              CASE_TABLE_ENTRY           OPCODE=CMPC3,-
0146    480                                        ROUTINE=CMPC3,-
0146    481                                        FPD_ROUTINE=VAX$CMPC3,-
0146    482                                        BOOT_FLAG=BOOT
013A    483
013A    484              CASE_TABLE_ENTRY           OPCODE=CMPC5,-
013A    485                                        ROUTINE=CMPC5,-
013A    486                                        FPD_ROUTINE=VAX$CMPC5,-
013A    487                                        BOOT_FLAG=BOOT
0142    488
0142    489              CASE_TABLE_ENTRY           OPCODE=LOCC,-
0142    490                                        ROUTINE=LOCC,-
0142    491                                        FPD_ROUTINE=VAX$LOCC,-
0142    492                                        BOOT_FLAG=BOOT
015C    493
015C    494              CASE_TABLE_ENTRY           OPCODE=SKPC,-
```

G 12

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 11   VAX
V04-000   Dispatch Tables   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (5)   V04

```
015C   495                                     ROUTINE=SKPC,-
015C   496                                     FPD_ROUTINE=VAX$SKPC
015E   497
015E   498              CASE_TABLE_ENTRY        OPCODE=SCANC,-
015E   499                                     ROUTINE=SCANC,-
015E   500                                     FPD_ROUTINE=VAX$SCANC
013C   501
013C   502              CASE_TABLE_ENTRY        OPCODE=SPANC,-
013C   503                                     ROUTINE=SPANC,-
013C   504                                     FPD_ROUTINE=VAX$SPANC
013E   505
013E   506              CASE_TABLE_ENTRY        OPCODE=MATCHC,-
013E   507                                     ROUTINE=MATCHC,-
013E   508                                     FPD_ROUTINE=VAX$MATCHC
015A   509
015A   510              CASE_TABLE_ENTRY        OPCODE=CRC,-
015A   511                                     ROUTINE=CRC,-
015A   512                                     FPD_ROUTINE=VAX$CRC
00FE   513
00FE   514 ; Now generate table entries for the decimal instructions
00FE   515
00FE   516              CASE_TABLE_ENTRY        OPCODE=ADDP4,-
00FE   517                                     ROUTINE=ADDP4,-
00FE   518                                     FPD_ROUTINE=VAX$ADDP4
0128   519
0128   520              CASE_TABLE_ENTRY        OPCODE=ADDP6,-
0128   521                                     ROUTINE=ADDP6,-
0128   522                                     FPD_ROUTINE=VAX$ADDP6
012A   523
012A   524              CASE_TABLE_ENTRY        OPCODE=ASHP,-
012A   525                                     ROUTINE=ASHP,-
012A   526                                     FPD_ROUTINE=VAX$ASHP
00D8   527
00D8   528              CASE_TABLE_ENTRY        OPCODE=CMPP3,-
00D8   529                                     ROUTINE=CMPP3,-
00D8   530                                     FPD_ROUTINE=VAX$CMPP3
0152   531
0152   532              CASE_TABLE_ENTRY        OPCODE=CMPP4,-
0152   533                                     ROUTINE=CMPP4,-
0152   534                                     FPD_ROUTINE=VAX$CMPP4
0156   535
0156   536              CASE_TABLE_ENTRY        OPCODE=CVTLP,-
0156   537                                     ROUTINE=CVTLP,-
0156   538                                     FPD_ROUTINE=VAX$CVTLP_RESTART
00DA   539
00DA   540              CASE_TABLE_ENTRY        OPCODE=CVTPL,-
00DA   541                                     ROUTINE=CVTPL,-
00DA   542                                     FPD_ROUTINE=VAX$CVTPL_RESTART
0154   543
0154   544              CASE_TABLE_ENTRY        OPCODE=CVTPS,-
0154   545                                     ROUTINE=CVTPS,-
0154   546                                     FPD_ROUTINE=VAX$CVTPS
00F8   547
00F8   548              CASE_TABLE_ENTRY        OPCODE=CVTPT,-
00F8   549                                     ROUTINE=CVTPT,-
00F8   550                                     FPD_ROUTINE=VAX$CVTPT_RESTART
0130   551
```

```
0130    552              CASE_TABLE_ENTRY          OPCODE=CVTSP,-
0130    553                                        ROUTINE=CVTSP,-
0130    554                                        FPD_ROUTINE=VAX$CVTSP
00FA    555
00FA    556              CASE_TABLE_ENTRY          OPCODE=CVTTP,-
00FA    557                                        ROUTINE=CVTTP,-
00FA    558                                        FPD_ROUTINE=VAX$CVTTP_RESTART
0134    559
0134    560              CASE_TABLE_ENTRY          OPCODE=DIVP,-
0134    561                                        ROUTINE=DIVP,-
0134    562                                        FPD_ROUTINE=VAX$DIVP
0136    563
0136    564              CASE_TABLE_ENTRY          OPCODE=MOVP,-
0136    565                                        ROUTINE=MOVP,-
0136    566                                        FPD_ROUTINE=VAX$MOVP
0150    567
0150    568              CASE_TABLE_ENTRY          OPCODE=MULP,-
0150    569                                        ROUTINE=MULP,-
0150    570                                        FPD_ROUTINE=VAX$MULP
0132    571
0132    572              CASE_TABLE_ENTRY          OPCODE=SUBP4,-
0132    573                                        ROUTINE=SUBP4,-
0132    574                                        FPD_ROUTINE=VAX$SUBP4
012C    575
012C    576              CASE_TABLE_ENTRY          OPCODE=SUBP6,-
012C    577                                        ROUTINE=SUBP6,-
012C    578                                        FPD_ROUTINE=VAX$SUBP6
012E    579
012E    580  ; EDITPC always seems to find itself in last place
012E    581
012E    582              CASE_TABLE_ENTRY          OPCODE=EDITPC,-
012E    583                                        ROUTINE=EDITPC,-
012E    584                                        FPD_ROUTINE=VAX$EDITPC_RESTART
0158    585
00000179 586             .RESTORE                              ; Reset current location counter
```

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 13   VAX
V04-000   Description of instruction-specific rout 5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (6)   V04

I 12

```
0179   588              .SUBTITLE        Description of instruction-specific routines
0179   589
0179   590    ;++
0179   591    ; The instruction-specific routines do similar things. Rather than clutter up
0179   592    ; each routine with the same comments, we will describe the steps that each
0179   593    ; routine takes in this section.
0179   594    ;
0179   595    ; The input parameters to each routine are identical.
0179   596    ;
0179   597    ;                    Contents of exception stack
0179   598    ;                    ---------------------------
0179   599    ;
0179   600    ;        OPCODE(SP)      - Opcode of reserved instruction
0179   601    ;        OLD_PC(SP)      - PC of reserved instruction
0179   602    ;        OPERAND_1(SP) - First operand specifier
0179   603    ;        OPERAND_2(SP) - Second operand specifier
0179   604    ;        OPERAND_3(SP) - Third operand specifier
0179   605    ;        OPERAND_4(SP) - Fourth operand specifier
0179   606    ;        OPERAND_5(SP) - Fifth operand specifier
0179   607    ;        OPERAND_6(SP) - Sixth operand specifier
0179   608    ;        OPERAND_7(SP) - Seventh operand specifier (currently unused)
0179   609    ;        OPERAND_8(SP) - Eight operand specifier (currently unused)
0179   610    ;        NEW_PC(SP)      - PC of instruction following reserved instruction
0179   611    ;        EXCEPTION_PSL(SP) - PSL at time of exception
0179   612    ;
0179   613    ;        The routine headers for the instruction-specific routines in this
0179   614    ;        module will list the input and output parameters in symbolic form
0179   615    ;        only. The VAX$xxxxxx routines in other modules in the emulator contain
0179   616    ;        the exact meanings of the various operands (parameters) to the
0179   617    ;        routines.
0179   618    ;
0179   619    ; Outline of execution:
0179   620    ;
0179   621    ;        The operands are loaded into registers as required by the instruction
0179   622    ;        specific routines. Routine headers for each routine contain detailed
0179   623    ;        descriptions.
0179   624    ;
0179   625    ;        A routine of the form VAX$xxxxxx (where xxxxxx is the instruction
0179   626    ;        name) is called to perform the actual work indicated by each
0179   627    ;        instruction.
0179   628    ;
0179   629    ;        Common exit code executes to allow the condition codes returned by the
0179   630    ;        VAX$xxxxxx routines to be passed back to the code that generated the
0179   631    ;        original exception.
0179   632    ;
0179   633    ; Notes:
0179   634    ;
0179   635    ;        The following routines are constructed to be reasonably fast. In
0179   636    ;        particular, each instruction has its own separate routine, even though
0179   637    ;        several instructions differ only in the instruction-specific routine
0179   638    ;        to which final control is passed. Rather than share this common code
0179   639    ;        at the expense of another dispatch on opcode, we shoose to duplicate
0179   640    ;        the common code.
0179   641    ;--
0179   642
```

```
                              0179   645              .SUBTITLE        MOVTC - Exception handler for MOVTC instruction
                              0179   646   ;+
                              0179   647   ; Input Parameters:
                              0179   648   ;
                              0179   649   ;        OPCODE(SP)
                              0179   650   ;        OLD_PC(SP)
                              0179   651   ;        OPERAND_1(SP) - srclen.rw
                              0179   652   ;        OPERAND_2(SP) - srcaddr.ab
                              0179   653   ;        OPERAND_3(SP) - fill.rb
                              0179   654   ;        OPERAND_4(SP) - tbladdr.ab
                              0179   655   ;        OPERAND_5(SP) - dstlen.rw
                              0179   656   ;        OPERAND_6(SP) - dstaddr.ab
                              0179   657   ;        OPERAND_7(SP)
                              0179   658   ;        OPERAND_8(SP)
                              0179   659   ;        NEW_PC(SP)
                              0179   660   ;        EXCEPTION_PSL(SP)
                              0179   661   ;
                              0179   662   ; Output Parameters:
                              0179   663   ;
                              0179   664   ;        R0<15:0> - srclen.rw
                              0179   665   ;        R1       - srcaddr.ab
                              0179   666   ;        R2<7:0>  - fill.rb
                              0179   667   ;        R3       - tbladdr.ab
                              0179   668   ;        R4<15:0> - dstlen.rw
                              0179   669   ;        R5       - dstaddr.ab
                              0179   670   ;
                              0179   671   ; Implicit Output:
                              0179   672   ;
                              0179   673   ;        R0<31:16> - 0
                              0179   674   ;        R2<31:8>  - 0
                              0179   675   ;        R4<31:16> - 0
                              0179   676   ;-
                              0179   677
                              0179   678   MOVTC:
                              0179   679
    50    08 AE   3C          0179   680              MOVZWL   OPERAND_1(SP),R0       ; R0<15:0> <- srclen.rw
    51    0C AE   D0          017D   681              MOVL     OPERAND_2(SP),R1       ; R1       <- srcaddr.ab
    52    10 AE   9A          0181   682              MOVZBL   OPERAND_3(SP),R2       ; R2<7:0>  <- fill.rb
    53    14 AE   D0          0185   683              MOVL     OPERAND_4(SP),R3       ; R3       <- tbladdr.ab
    54    18 AE   3C          0189   684              MOVZWL   OPERAND_5(SP),R4       ; R4<15:0> <- dstlen.rw
    55    1C AE   D0          018D   685              MOVL     OPERAND_6(SP),R5       ; R5       <- dstaddr.ab
                              0191   686
                              0191   687   ; Now that the operands have been loaded, the only exception parameter
                              0191   688   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              0191   689   ; there is no reason why the state of the stack needs to be altered and we
                              0191   690   ; save two instructions if we leave the stack alone.
                              0191   691
        0425'CF   9F          0191   692              PUSHAB   VAX$EXIT_EMULATOR      ; Store the return PC
         FE68'    31          0195   693              BRW      VAX$MOVTC              ; Do the actual work
```

K 12

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 15
V04-000   MOVTUC - Exception handler for MOVTUC in   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (8)

```
                         0198   697            .SUBTITLE        MOVTUC - Exception handler for MOVTUC instruction
                         0198   698   ;+
                         0198   699   ; Input Parameters:
                         0198   700   ;
                         0198   701   ;          OPCODE(SP)
                         0198   702   ;          OLD_PC(SP)
                         0198   703   ;          OPERAND_1(SP) - srclen.rw
                         0198   704   ;          OPERAND_2(SP) - srcaddr.ab
                         0198   705   ;          OPERAND_3(SP) - esc.rb
                         0198   706   ;          OPERAND_4(SP) - tbladdr.ab
                         0198   707   ;          OPERAND_5(SP) - dstlen.rw
                         0198   708   ;          OPERAND_6(SP) - dstaddr.ab
                         0198   709   ;          OPERAND_7(SP)
                         0198   710   ;          OPERAND_8(SP)
                         0198   711   ;          NEW_PC(SP)
                         0198   712   ;          EXCEPTION_PSL(SP)
                         0198   713   ;
                         0198   714   ; Output Parameters:
                         0198   715   ;
                         0198   716   ;          R0<15:0> - srclen.rw
                         0198   717   ;          R1       - srcaddr.ab
                         0198   718   ;          R2<7:0>  - esc.rb
                         0198   719   ;          R3       - tbladdr.ab
                         0198   720   ;          R4<15:0> - dstlen.rw
                         0198   721   ;          R5       - dstaddr.ab
                         0198   722   ;
                         0198   723   ; Implicit Output:
                         0198   724   ;
                         0198   725   ;          R0<31:16> - 0
                         0198   726   ;          R2<31:8>  - 0
                         0198   727   ;          R4<31:16> - 0
                         0198   728   ;-
                         0198   729
                         0198   730   MOVTUC:
                         0198   731
    50   08 AE   3C      0198   732            MOVZWL   OPERAND_1(SP),R0          ; R0<15:0> <- srclen.rw
    51   0C AE   D0      019C   733            MOVL     OPERAND_2(SP),R1          ; R1       <- srcaddr.ab
    52   10 AE   9A      01A0   734            MOVZBL   OPERAND_3(SP),R2          ; R2<7:0>  <- esc.rb
    53   14 AE   D0      01A4   735            MOVL     OPERAND_4(SP),R3          ; R3       <- tbladdr.ab
    54   18 AE   3C      01A8   736            MOVZWL   OPERAND_5(SP),R4          ; R4<15:0> <- dstlen.rw
    55   1C AE   D0      01AC   737            MOVL     OPERAND_6(SP),R5          ; R5       <- dstaddr.ab
                         01B0   738
                         01B0   739   ; Now that the operands have been loaded, the only exception parameter
                         01B0   740   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         01B0   741   ; there is no reason why the state of the stack needs to be altered and we
                         01B0   742   ; save two instructions if we leave the stack alone.
                         01B0   743
        0425'CF   9F     01B0   744            PUSHAB   VAX$EXIT_EMULATOR         ; Store the return PC
          FE49'   31     01B4   745            BRW      VAX$MOVTUC                ; Do the actual work
```

```
                         01B7      749              .SUBTITLE      CMPC3 - Exception handler for CMPC3 instruction
                         01B7      750      ;+
                         01B7      751      ; Input Parameters:
                         01B7      752      ;
                         01B7      753      ;         OPCODE(SP)
                         01B7      754      ;         OLD_PC(SP)
                         01B7      755      ;         OPERAND_1(SP) - len.rw
                         01B7      756      ;         OPERAND_2(SP) - src1addr.ab
                         01B7      757      ;         OPERAND_3(SP) - src2addr.ab
                         01B7      758      ;         OPERAND_4(SP)
                         01B7      759      ;         OPERAND_5(SP)
                         01B7      760      ;         OPERAND_6(SP)
                         01B7      761      ;         OPERAND_7(SP)
                         01B7      762      ;         OPERAND_8(SP)
                         01B7      763      ;         NEW_PC(SP)
                         01B7      764      ;         EXCEPTION_PSL(SP)
                         01B7      765      ;
                         01B7      766      ; Output Parameters:
                         01B7      767      ;
                         01B7      768      ;         R0<15:0> - len.rw
                         01B7      769      ;         R1       - src1addr.ab
                         01B7      770      ;         R3       - src2addr.ab
                         01B7      771      ;
                         01B7      772      ; Implicit Output:
                         01B7      773      ;
                         01B7      774      ;         R0<31:16> - 0
                         01B7      775      ;         R2        - UNPREDICTABLE
                         01B7      776      ;-
                         01B7      777
                         01B7      778  CMPC3:
                         01B7      779
50     08 AE    3C       01B7      780              MOVZWL    OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
51     0C AE    D0       01BB      781              MOVL      OPERAND_2(SP),R1        ; R1       <- src1addr.ab
53     10 AE    D0       01BF      782              MOVL      OPERAND_3(SP),R3        ; R3       <- src2addr.ab
                         01C3      783
                         01C3      784      ; Now that the operands have been loaded, the only exception parameter
                         01C3      785      ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         01C3      786      ; there is no reason why the state of the stack needs to be altered and we
                         01C3      787      ; save two instructions if we leave the stack alone.
                         01C3      788
       0425'CF   9F      01C3      789              PUSHAB    VAX$EXIT_EMULATOR       ; Store the return PC
        FE36'    31      01C7      790              BRW       VAX$CMPC3               ; Do the actual work
```

M 12

```
                              01CA    794            .SUBTITLE          CMPC5 - Exception handler for CMPC5 instruction
                              01CA    795  ;+
                              01CA    796  ; Input Parameters:
                              01CA    797  ;
                              01CA    798  ;        OPCODE(SP)
                              01CA    799  ;        OLD_PC(SP)
                              01CA    800  ;        OPERAND_1(SP) - src1len.rw
                              01CA    801  ;        OPERAND_2(SP) - src1addr.ab
                              01CA    802  ;        OPERAND_3(SP) - fill.rb
                              01CA    803  ;        OPERAND_4(SP) - src2len.rw
                              01CA    804  ;        OPERAND_5(SP) - src2addr.ab
                              01CA    805  ;        OPERAND_6(SP)
                              01CA    806  ;        OPERAND_7(SP)
                              01CA    807  ;        OPERAND_8(SP)
                              01CA    808  ;        NEW_PC(SP)
                              01CA    809  ;        EXCEPTION_PSL(SP)
                              01CA    810  ;
                              01CA    811  ; Output Parameters:
                              01CA    812  ;
                              01CA    813  ;        R0<15:0>   - srclen.rw
                              01CA    814  ;        R0<23:16>  - fill.rb
                              01CA    815  ;        R1         - srcaddr.ab
                              01CA    816  ;        R2<15:0>   - src2len.rw
                              01CA    817  ;        R3         - src2addr.ab
                              01CA    818  ;
                              01CA    819  ; Implicit Output:
                              01CA    820  ;
                              01CA    821  ;        R0<31:24> - UNPREDICTABLE
                              01CA    822  ;        R2<31:16> - 0
                              01CA    823  ;-
                              01CA    824
                              01CA    825  CMPC5:
                              01CA    826
50   10 AE    10    9C        01CA    827            ROTL     #16,OPERAND_3(SP),R0    ; R0<23:16> <- fill.rb
50      08 AE       B0        01CF    828            MOVW     OPERAND_1(SP),R0        ; R0<15:0>  <- src1len.rw
51      0C AE       D0        01D3    829            MOVL     OPERAND_2(SP),R1        ; R1        <- src1addr.ab
52      14 AE       3C        01D7    830            MOVZWL   OPERAND_4(SP),R2        ; R2<15:0>  <- src2len.rw
53      18 AE       D0        01DB    831            MOVL     OPERAND_5(SP),R3        ; R3        <sca- src2addr.ab
                              01DF    832
                              01DF    833  ; Now that the operands have been loaded, the only exception parameter
                              01DF    834  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              01DF    835  ; there is no reason why the state of the stack needs to be altered and we
                              01DF    836  ; save two instructions if we leave the stack alone.
                              01DF    837
        0425'CF    9F        01DF    838            PUSHAB   VAX$EXIT_EMULATOR        ; Store the return PC
           FE1A'    31        01E3    839            BRW      VAX$CMPC5                ; Do the actual work
```

N 12

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 18
V04-000   SCANC - Exception handler for SCANC inst 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (11)

```
                                01E6   843            .SUBTITLE        SCANC - Exception handler for SCANC instruction
                                01E6   844  ;+
                                01E6   845  ; Input Parameters:
                                01E6   846  ;
                                01E6   847  ;         OPCODE(SP)
                                01E6   848  ;         OLD_PC(SP)
                                01E6   849  ;         OPERAND_1(SP) - len.rw
                                01E6   850  ;         OPERAND_2(SP) - addr.ab
                                01E6   851  ;         OPERAND_3(SP) - tbladdr.ab
                                01E6   852  ;         OPERAND_4(SP) - mask.ab
                                01E6   853  ;         OPERAND_5(SP)
                                01E6   854  ;         OPERAND_6(SP)
                                01E6   855  ;         OPERAND_7(SP)
                                01E6   856  ;         OPERAND_8(SP)
                                01E6   857  ;         NEW_PC(SP)
                                01E6   858  ;         EXCEPTION_PSL(SP)
                                01E6   859  ;
                                01E6   860  ; Output Parameters:
                                01E6   861  ;
                                01E6   862  ;         R0<15:0> - len.rw
                                01E6   863  ;         R1       - addr.ab
                                01E6   864  ;         R2<7:0>  - mask.rb
                                01E6   865  ;         R3       - tbladdr.ab
                                01E6   866  ;
                                01E6   867  ; Implicit Output:
                                01E6   868  ;
                                01E6   869  ;         R0<31:16> - 0
                                01E6   870  ;         R2<31:8>  - 0
                                01E6   871  ;-
                                01E6   872
                                01E6   873  SCANC:
                                01E6   874
50      08 AE   3C              01E6   875            MOVZWL   OPERAND_1(SP),R0     ; R0<15:0> <- len.rw
51      0C AE   D0              01EA   876            MOVL     OPERAND_2(SP),R1     ; R1       <- addr.ab
53      10 AE   D0              01EE   877            MOVL     OPERAND_3(SP),R3     ; R3       <- tbladdr.ab
52      14 AE   9A              01F2   878            MOVZBL   OPERAND_4(SP),R2     ; R2<7:0>  <- mask.ab
                                01F6   879
                                01F6   880  ; Now that the operands have been loaded, the only exception parameter
                                01F6   881  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                01F6   882  ; there is no reason why the state of the stack needs to be altered and we
                                01F6   883  ; save two instructions if we leave the stack alone.
                                01F6   884
        0425'CF  9F             01F6   885            PUSHAB   VAX$EXIT_EMULATOR    ; Store the return PC
          FE03'  31             01FA   886            BRW      VAX$SCANC            ; Do the actual work
```

B 13

VAX$EMU_ATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 19   VAX$
V04-000   SPANC - Exception handler for SPANC inst 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (12)   V04-

```
                          01FD    890              .SUBTITLE        SPANC - Exception handler for SPANC instruction
                          01FD    891    ;+
                          01FD    892    ; Input Parameters:
                          01FD    893    ;
                          01FD    894    ;        OPCODE(SP)
                          01FD    895    ;        OLD_PC(SP)
                          01FD    896    ;        OPERAND_1(SP) - len.rw
                          01FD    897    ;        OPERAND_2(SP) - addr.ab
                          01FD    898    ;        OPERAND_3(SP) - tbladdr.ab
                          01FD    899    ;        OPERAND_4(SP) - mask.ab
                          01FD    900    ;        OPERAND_5(SP)
                          01FD    901    ;        OPERAND_6(SP)
                          01FD    902    ;        OPERAND_7(SP)
                          01FD    903    ;        OPERAND_8(SP)
                          01FD    904    ;        NEW_PC(SP)
                          01FD    905    ;        EXCEPTION_PSL(SP)
                          01FD    906    ;
                          01FD    907    ; Output Parameters:
                          01FD    908    ;
                          01FD    909    ;        R0<15:0> - len.rw
                          01FD    910    ;        R1       - addr.ab
                          01FD    911    ;        R2<7:0>  - mask.rb
                          01FD    912    ;        R3       - tbladdr.ab
                          01FD    913    ;
                          01FD    914    ; Implicit Output:
                          01FD    915    ;
                          01FD    916    ;        R0<31:16> - 0
                          01FD    917    ;        R2<31:8>  - 0
                          01FD    918    ;-
                          01FD    919
                          01FD    920    SPANC:
                          01FD    921
50    08 AE  3C           01FD    922              MOVZWL    OPERAND_1(SP),R0     ; R0<15:0> <- len.rw
51    0C AE  D0           0201    923              MOVL      OPERAND_2(SP),R1     ; R1       <- addr.ab
53    10 AE  D0           0205    924              MOVL      OPERAND_3(SP),R3     ; R3       <- tbladdr.ab
52    14 AE  9A           0209    925              MOVZBL    OPERAND_4(SP),R2     ; R2<7:0>  <- mask.ab
                          020D    926
                          020D    927    ; Now that the operands have been loaded, the only exception parameter
                          020D    928    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          020D    929    ; there is no reason why the state of the stack needs to be altered and we
                          020D    930    ; save two instructions if we leave the stack alone.
                          020D    931
       0425'CF  9F        020D    932              PUSHAB    VAX$EXIT_EMULATOR    ; Store the return PC
         FDEC'  31        0211    933              BRW       VAX$SPANC            ; Do the actual work
```

C 13

```
                                0214   937              .SUBTITLE        LOCC - Exception handler for LOCC instruction
                                0214   938  ;+
                                0214   939  ; Input Parameters:
                                0214   940  ;
                                0214   941  ;        OPCODE(SP)
                                0214   942  ;        OLD_PC(SP)
                                0214   943  ;        OPERAND_1(SP) - char.rb
                                0214   944  ;        OPERAND_2(SP) - len.rw
                                0214   945  ;        OPERAND_3(SP) - addr.ab
                                0214   946  ;        OPERAND_4(SP)
                                0214   947  ;        OPERAND_5(SP)
                                0214   948  ;        OPERAND_6(SP)
                                0214   949  ;        OPERAND_7(SP)
                                0214   950  ;        OPERAND_8(SP)
                                0214   951  ;        NEW_PC(SP)
                                0214   952  ;        EXCEPTION_PSL(SP)
                                0214   953  ;
                                0214   954  ; Output Parameters:
                                0214   955  ;
                                0214   956  ;        R0<15:0>  - len.rw
                                0214   957  ;        R0<23:16> - char.rb
                                0214   958  ;        R1        - addr.ab
                                0214   959  ;
                                0214   960  ; Implicit Output:
                                0214   961  ;
                                0214   962  ;        R0<31:24> - UNPREDICTABLE
                                0214   963  ;-
                                0214   964
                                0214   965  LOCC:
                                0214   966
50   08 AE   10   9C           0214   967              ROTL     #16,OPERAND_1(SP),R0     ; R0<23:16> <- char.ab
     50   0C AE   B0           0219   968              MOVW     OPERAND_2(SP),R0         ; R0<15:0>  <- len.rw
     51   10 AE   D0           021D   969              MOVL     OPERAND_3(SP),R1         ; R1        <- addr.ab
                                0221   970
                                0221   971  ; Now that the operands have been loaded, the only exception parameter
                                0221   972  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                0221   973  ; there is no reason why the state of the stack needs to be altered and we
                                0221   974  ; save two instructions if we leave the stack alone.
                                0221   975
     0425'CF   9F              0221   976              PUSHAB   VAX$EXIT_EMULATOR        ; Store the return PC
         FDD8'  31             0225   977              BRW      VAX$LOCC                 ; Do the actual work
```

D 13

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 21
V04-000   SKPC - Exception handler for SKPC instru   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (14)

```
                           0228    981            .SUBTITLE        SKPC - Exception handler for SKPC instruction
                           0228    982  ;+
                           0228    983  ; Input Parameters:
                           0228    984  ;
                           0228    985  ;       OPCODE(SP)
                           0228    986  ;       OLD_PC(SP)
                           0228    987  ;       OPERAND_1(SP) - char.rb
                           0228    988  ;       OPERAND_2(SP) - len.rw
                           0228    989  ;       OPERAND_3(SP) - addr.ab
                           0228    990  ;       OPERAND_4(SP)
                           0228    991  ;       OPERAND_5(SP)
                           0228    992  ;       OPERAND_6(SP)
                           0228    993  ;       OPERAND_7(SP)
                           0228    994  ;       OPERAND_8(SP)
                           0228    995  ;       NEW_PC(SP)
                           0228    996  ;       EXCEPTION_PSL(SP)
                           0228    997  ;
                           0228    998  ; Output Parameters:
                           0228    999  ;
                           0228   1000  ;       R0<15:0>  - len.rw
                           0228   1001  ;       R0<23:16> - char.rb
                           0228   1002  ;       R1        - addr.ab
                           0228   1003  ;
                           0228   1004  ; Implicit Output:
                           0228   1005  ;
                           0228   1006  ;       R0<31:24> - UNPREDICTABLE
                           0228   1007  ;-
                           0228   1008
                           0228   1009  SKPC:
                           0228   1010
50      08 AE  10   9C     0228   1011            ROTL    #16,OPERAND_1(SP),R0    ; R0<23:16> <- char.ab
        50    0C AE   B0   022D   1012            MOVW    OPERAND_2(SP),R0        ; R0<15:0>  <- len.rw
        51    10 AE   D0   0231   1013            MOVL    OPERAND_3(SP),R1        ; R1        <- addr.ab
                           0235   1014
                           0235   1015  ; Now that the operands have been loaded, the only exception parameter
                           0235   1016  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                           0235   1017  ; there is no reason why the state of the stack needs to be altered and we
                           0235   1018  ; save two instructions if we leave the stack alone.
                           0235   1019
        0425'CF    9F      0235   1020            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
          FDC4'    31      0239   1021            BRW     VAX$SKPC                ; Do the actual work
```

E 13

VAX$EMULATE   - \AX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 22   VAX1
V04-000   MATCHC - Exception handler for MATCHC in   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (15)   V04-

```
                        023C  1025          .SUBTITLE       MATCHC - Exception handler for MATCHC instruction
                        023C  1026  ;+
                        023C  1027  ; Input Parameters:
                        023C  1028  ;
                        023C  1029  ;       OPCODE(SP)
                        023C  1030  ;       OLD_PC(SP)
                        023C  1031  ;       OPERAND_1(SP) - objlen.rw
                        023C  1032  ;       OPERAND_2(SP) - objaddr.ab
                        023C  1033  ;       OPERAND_3(SP) - srclen.rw
                        023C  1034  ;       OPERAND_4(SP) - srcaddr.ab
                        023C  1035  ;       OPERAND_5(SP)
                        023C  1036  ;       OPERAND_6(SP)
                        023C  1037  ;       OPERAND_7(SP)
                        023C  1038  ;       OPERAND_8(SP)
                        023C  1039  ;       NEW_PC(SP)
                        023C  1040  ;       EXCEPTION_PSL(SP)
                        023C  1041  ;
                        023C  1042  ; Output Parameters:
                        023C  1043  ;
                        023C  1044  ;       R0<15:0>  - objlen.rw
                        023C  1045  ;       R1        - objaddr.ab
                        023C  1046  ;       R2<15:0>  - srclen.rw
                        023C  1047  ;       R3        - srcaddr.ab
                        023C  1048  ;
                        023C  1049  ; Implicit Output:
                        023C  1050  ;
                        023C  1051  ;       R0<31:16> - 0
                        023C  1052  ;       R2<31:16> - 0
                        023C  1053  ;-
                        023C  1054
                        023C  1055  MATCHC:
                        023C  1056
50    08 AE    3C       023C  1057          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0>  <- objlen.rw
51    0C AE    D0       0240  1058          MOVL    OPERAND_2(SP),R1        ; R1        <- objaddr.ab
52    10 AE    3C       0244  1059          MOVZWL  OPERAND_3(SP),R2        ; R2<15:0>  <- srclen.rw
53    14 AE    D0       0248  1060          MOVL    OPERAND_4(SP),R3        ; R3        <- srcaddr.ab
                        024C  1061
                        024C  1062  ; Now that the operands have been loaded, the only exception parameter
                        024C  1063  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        024C  1064  ; there is no reason why the state of the stack needs to be altered and we
                        024C  1065  ; save two instructions if we leave the stack alone.
                        024C  1066
   0425'CF    9F        024C  1067          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
      FDAD'    31       0250  1068          BRW     VAX$MATCHC              ; Do the actual work
```

F 13

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 23   VAX
V04-000   CRC - Exception handler for CRC instruct   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (16)   V04

```
                            0253  1072          .SUBTITLE       CRC - Exception handler for CRC instruction
                            0253  1073   ;+
                            0253  1074   ; Input Parameters:
                            0253  1075   ;
                            0253  1076   ;         OPCODE(SP)
                            0253  1077   ;         OLD_PC(SP)
                            0253  1078   ;         OPERAND_1(SP) - tbl.ab
                            0253  1079   ;         OPERAND_2(SP) - inicrc.rl
                            0253  1080   ;         OPERAND_3(SP) - strlen.rw
                            0253  1081   ;         OPERAND_4(SP) - stream.ab
                            0253  1082   ;         OPERAND_5(SP)
                            0253  1083   ;         OPERAND_6(SP)
                            0253  1084   ;         OPERAND_7(SP)
                            0253  1085   ;         OPERAND_8(SP)
                            0253  1086   ;         NEW_PC(SP)
                            0253  1087   ;         EXCEPTION_PSL(SP)
                            0253  1088   ;
                            0253  1089   ; Output Parameters:
                            0253  1090   ;
                            0253  1091   ;         R0         - inicrc.rl
                            0253  1092   ;         R1         - tbl.ab
                            0253  1093   ;         R2<15:0>   - strlen.rw
                            0253  1094   ;         R3         - stream.ab
                            0253  1095   ;
                            0253  1096   ; Implicit Output:
                            0253  1097   ;
                            0253  1098   ;         R2<31:16> - 0
                            0253  1099   ;-
                            0253  1100
                            0253  1101   CRC:
                            0253  1102
   51   08 AE   D0          0253  1103          MOVL    OPERAND_1(SP),R1        ; R1         <- tbl.ab
   50   0C AE   D0          0257  1104          MOVL    OPERAND_2(SP),R0        ; R0         <- inicrc.rl
   52   10 AE   3C          025B  1105          MOVZWL  OPERAND_3(SP),R2        ; R2<15:0>   <- strlen.rw
   53   14 AE   D0          025F  1106          MOVL    OPERAND_4(SP),R3        ; R3         <- stream.ab
                            0263  1107
                            0263  1108   ; Now that the operands have been loaded, the only exception parameter
                            0263  1109   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                            0263  1110   ; there is no reason why the state of the stack needs to be altered and we
                            0263  1111   ; save two instructions if we leave the stack alone.
                            0263  1112
      0425'CF   9F          0263  1113          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
        FD96'   31          0267  1114          BRW     VAX$CRC                 ; Do the actual work
```

G 13

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 24
V04-000   ADDP4 - Exception handler for ADDP4 inst   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (17)

```
                         026A  1118              .SUBTITLE      ADDP4 - Exception handler for ADDP4 instruction
                         026A  1119  ;+
                         026A  1120  ; Input Parameters:
                         026A  1121  ;
                         026A  1122  ;         OPCODE(SP)
                         026A  1123  ;         OLD_PC(SP)
                         026A  1124  ;         OPERAND_1(SP) - addlen.rw
                         026A  1125  ;         OPERAND_2(SP) - addaddr.ab
                         026A  1126  ;         OPERAND_3(SP) - sumlen.rw
                         026A  1127  ;         OPERAND_4(SP) - sumaddr.ab
                         026A  1128  ;         OPERAND_5(SP)
                         026A  1129  ;         OPERAND_6(SP)
                         026A  1130  ;         OPERAND_7(SP)
                         026A  1131  ;         OPERAND_8(SP)
                         026A  1132  ;         NEW_PC(SP)
                         026A  1133  ;         EXCEPTION_PSL(SP)
                         026A  1134  ;
                         026A  1135  ; Output Parameters:
                         026A  1136  ;
                         026A  1137  ;         R0<15:0> - addlen.rw
                         026A  1138  ;         R1       - addaddr.ab
                         026A  1139  ;         R2<15:0> - sumlen.rw
                         026A  1140  ;         R3       - sumaddr.ab
                         026A  1141  ;
                         026A  1142  ; Implicit Output:
                         026A  1143  ;
                         026A  1144  ;         R0<31:16> - 0
                         026A  1145  ;         R2<31:16> - 0
                         026A  1146  ;-
                         026A  1147
                         026A  1148  ADDP4:
                         026A  1149
50    08 AE   3C         026A  1150              MOVZWL   OPERAND_1(SP),R0      ; R0<15:0> <- addlen.rw
51    0C AE   D0         026E  1151              MOVL     OPERAND_2(SP),R1      ; R1       <- addaddr.ab
52    10 AE   3C         0272  1152              MOVZWL   OPERAND_3(SP),R2      ; R2<15:0> <- sumlen.rw
53    14 AE   D0         0276  1153              MOVL     OPERAND_4(SP),R3      ; R3       <- sumaddr.ab
                         027A  1154
                         027A  1155  ; Now that the operands have been loaded, the only exception parameter
                         027A  1156  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         027A  1157  ; there is no reason why the state of the stack needs to be altered and we
                         027A  1158  ; save two instructions if we leave the stack alone.
                         027A  1159
    0425'CF   9F         027A  1160              PUSHAB   VAX$EXIT_EMULATOR     ; Store the return PC
      FD7F'   31         027E  1161              BRW      VAX$ADDP4             ; Do the actual work
```

H 13

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 25   VAX$
V04-000   ADDP6 - Exception handler for ADDP6 inst 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (18)   V04-

```
                    0281   1165            .SUBTITLE        ADDP6 - Exception handler for ADDP6 instruction
                    0281   1166   ;+
                    0281   1167   ; Input Parameters:
                    0281   1168   ;
                    0281   1169   ;         OPCODE(SP)
                    0281   1170   ;         OLD_PC(SP)
                    0281   1171   ;         OPERAND_1(SP) - add1len.rw
                    0281   1172   ;         OPERAND_2(SP) - add1addr.ab
                    0281   1173   ;         OPERAND_3(SP) - add2len.rw
                    0281   1174   ;         OPERAND_4(SP) - add2addr.ab
                    0281   1175   ;         OPERAND_5(SP) - sumlen.rw
                    0281   1176   ;         OPERAND_6(SP) - sumaddr.ab
                    0281   1177   ;         OPERAND_7(SP)
                    0281   1178   ;         OPERAND_8(SP)
                    0281   1179   ;         NEW_PC(SP)
                    0281   1180   ;         EXCEPTION_PSL(SP)
                    0281   1181   ;
                    0281   1182   ; Output Parameters:
                    0281   1183   ;
                    0281   1184   ;         R0<15:0> - add1len.rw
                    0281   1185   ;         R1       - add1addr.ab
                    0281   1186   ;         R2<15:0> - add2len.rw
                    0281   1187   ;         R3       - add2addr.ab
                    0281   1188   ;         R4<15:0> - sumlen.rw
                    0281   1189   ;         R5       - sumaddr.ab
                    0281   1190   ;
                    0281   1191   ; Implicit Output:
                    0281   1192   ;
                    0281   1193   ;         R0<31:16> - 0
                    0281   1194   ;         R2<31:16> - 0
                    0281   1195   ;         R4<31:16> - 0
                    0281   1196   ;-
                    0281   1197
                    0281   1198   ADDP6:
                    0281   1199
50   08 AE    3C    0281   1200            MOVZWL   OPERAND_1(SP),R0         ; R0<15:0> <- add1len.rw
51   0C AE    D0    0285   1201            MOVL     OPERAND_2(SP),R1         ; R1       <- add1addr.ab
52   10 AE    3C    0289   1202            MOVZWL   OPERAND_3(SP),R2         ; R2<15:0> <- add2len.rw
53   14 AE    D0    028D   1203            MOVL     OPERAND_4(SP),R3         ; R3       <- add2addr.ab
54   18 AE    3C    0291   1204            MOVZWL   OPERAND_5(SP),R4         ; R4<15:0> <- sumlen.rw
55   1C AE    D0    0295   1205            MOVL     OPERAND_6(SP),R5         ; R5       <- sumaddr.ab
                    0299   1206
                    0299   1207   ; Now that the operands have been loaded, the only exception parameter
                    0299   1208   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                    0299   1209   ; there is no reason why the state of the stack needs to be altered and we
                    0299   1210   ; save two instructions if we leave the stack alone.
                    0299   1211
     0425'CF   9F   0299   1212            PUSHAB   VAX$EXIT_EMULATOR        ; Store the return PC
       FD60'   31   029D   1213            BRW      VAX$ADDP6                ; Do the actual work
```

I 13

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 26   VAX$
V04-000   ASHP - Exception handler for ASHP instru 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (19)   V04-

```
                        02A0  1217              .SUBTITLE         ASHP - Exception handler for ASHP instruction
                        02A0  1218  ;+
                        02A0  1219  ; Input Parameters:
                        02A0  1220  ;
                        02A0  1221  ;           OPCODE(SP)
                        02A0  1222  ;           OLD_PC(SP)
                        02A0  1223  ;           OPERAND_1(SP) - cnt.rb
                        02A0  1224  ;           OPERAND_2(SP) - srclen.rw
                        02A0  1225  ;           OPERAND_3(SP) - srcaddr.ab
                        02A0  1226  ;           OPERAND_4(SP) - round.rb
                        02A0  1227  ;           OPERAND_5(SP) - dstlen.rw
                        02A0  1228  ;           OPERAND_6(SP) - dstaddr.ab
                        02A0  1229  ;           OPERAND_7(SP)
                        02A0  1230  ;           OPERAND_8(SP)
                        02A0  1231  ;           NEW_PC(SP)
                        02A0  1232  ;           EXCEPTION_PSL(SP)
                        02A0  1233  ;
                        02A0  1234  ; Output Parameters:
                        02A0  1235  ;
                        02A0  1236  ;           R0<15:0>  - srclen.rw
                        02A0  1237  ;           R0<31:16> - count.rb
                        02A0  1238  ;           R1        - srcaddr.ab
                        02A0  1239  ;           R2<15:0>  - dstlen.rw
                        02A0  1240  ;           R2<31:16> - round.rb
                        02A0  1241  ;           R3        - dstaddr.ab
                        02A0  1242  ;
                        02A0  1243  ; Implicit Output:
                        02A0  1244  ;
                        02A0  1245  ;           R0<31:24> - UNPREDICTABLE
                        02A0  1246  ;           R2<31:24> - UNPREDICTABLE
                        02A0  1247  ;-
                        02A0  1248
                        02A0  1249  ASHP:
                        02A0  1250
50    08 AE   10  9C    02A0  1251              ROTL      #16,OPERAND_1(SP),R0    ; R0<31:16> <- count.rb
      50  0C AE   B0    02A5  1252              MOVW      OPERAND_2(SP),R0        ; R0<15:0>  <- srclen.rw
      51    10 AE   D0  02A9  1253              MOVL      OPERAND_3(SP),R1        ; R1        <- srcaddr.ab
52    14 AE   10  9C    02AD  1254              ROTL      #16,OPERAND_4(SP),R2    ; R2<31:16> <- round.rb
      52  18 AE   B0    02B2  1255              MOVW      OPERAND_5(SP),R2        ; R2<15:0>  <- dstlen.rw
      53  1C AE   D0    02B6  1256              MOVL      OPERAND_6(SP),R3        ; R3        <- dstaddr.ab
                        02BA  1257
                        02BA  1258  ; Now that the operands have been loaded, the only exception parameter
                        02BA  1259  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        02BA  1260  ; there is no reason why the state of the stack needs to be altered and we
                        02BA  1261  ; save two instructions if we leave the stack alone.
                        02BA  1262
       0425'CF  9F      02BA  1263              PUSHAB    VAX$EXIT_EMULATOR       ; Store the return PC
         FD3F'  31      02BE  1264              BRW       VAX$ASHP                ; Do the actual work
```

VAX$EMULATE
V04-000

J 13

- VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 27
CMPP3 - Exception handler for CMPP3 inst 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (20)

VAX'
V04

```
                              02C1   1268              .SUBTITLE        CMPP3 - Exception handler for CMPP3 instruction
                              02C1   1269      ;+
                              02C1   1270      ; Input Parameters:
                              02C1   1271      ;
                              02C1   1272      ;          OPCODE(SP)
                              02C1   1273      ;          OLD_PC(SP)
                              02C1   1274      ;          OPERAND_1(SP) - len.rw
                              02C1   1275      ;          OPERAND_2(SP) - src1addr.ab
                              02C1   1276      ;          OPERAND_3(SP) - src2addr.ab
                              02C1   1277      ;          OPERAND_4(SP)
                              02C1   1278      ;          OPERAND_5(SP)
                              02C1   1279      ;          OPERAND_6(SP)
                              02C1   1280      ;          OPERAND_7(SP)
                              02C1   1281      ;          OPERAND_8(SP)
                              02C1   1282      ;          NEW_PC(SP)
                              02C1   1283      ;          EXCEPTION_PSL(SP)
                              02C1   1284      ;
                              02C1   1285      ; Output Parameters:
                              02C1   1286      ;
                              02C1   1287      ;          R0<15:0> - len.rw
                              02C1   1288      ;          R1        - src1addr.ab
                              02C1   1289      ;          R3        - src2addr.ab
                              02C1   1290      ;
                              02C1   1291      ; Implicit Output:
                              02C1   1292      ;
                              02C1   1293      ;          R0<31:16> - 0
                              02C1   1294      ;          R2        - UNPREDICTABLE
                              02C1   1295      ;-
                              02C1   1296
                              02C1   1297  CMPP3:
                              02C1   1298
50   08 AE   3C   02C1   1299              MOVZWL   OPERAND_1(SP),R0         ; R0<15:0> <- len.rw
51   0C AE   D0   02C5   1300              MOVL     OPERAND_2(SP),R1         ; R1        <- src1addr.ab
53   10 AE   D0   02C9   1301              MOVL     OPERAND_3(SP),R3         ; R3        <- src2addr.ab
                              02CD   1302
                              02CD   1303      ; Now that the operands have been loaded, the only exception parameter
                              02CD   1304      ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              02CD   1305      ; there is no reason why the state of tne stack needs to be altered and we
                              02CD   1306      ; save two instructions if we leave the stack alone.
                              02CD   1307
     0425'CF   9F   02CD   1308              PUSHAB   VAX$EXIT_EMULATOR        ; Store the return PC
        FD2C'   31   02D1   1309              BRW      VAX$CMPP3                ; Do the actual work
```

K 13

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 28
V04-000   CMPP4 - Exception handler for CMPP4 inst 5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (21)

```
                        02D4  1313           .SUBTITLE        CMPP4 - Exception handler for CMPP4 instruction
                        02D4  1314  ;+
                        02D4  1315  ; Input Parameters:
                        02D4  1316  ;
                        02D4  1317  ;        OPCODE(SP)
                        02D4  1318  ;        OLD_PC(SP)
                        02D4  1319  ;        OPERAND_1(SP) - src1len.rw
                        02D4  1320  ;        OPERAND_2(SP) - src1addr.ab
                        02D4  1321  ;        OPERAND_3(SP) - src2len.rw
                        02D4  1322  ;        OPERAND_4(SP) - src2addr.ab
                        02D4  1323  ;        OPERAND_5(SP)
                        02D4  1324  ;        OPERAND_6(SP)
                        02D4  1325  ;        OPERAND_7(SP)
                        02D4  1326  ;        OPERAND_8(SP)
                        02D4  1327  ;        NEW_PC(SP)
                        02D4  1328  ;        EXCEPTION_PSL(SP)
                        02D4  1329  ;
                        02D4  1330  ; Output Parameters:
                        02D4  1331  ;
                        02D4  1332  ;        R0<15:0> - src1len.rw
                        02D4  1333  ;        R1       - src1addr.ab
                        02D4  1334  ;        R2<15:0> - src2len.rw
                        02D4  1335  ;        R3       - src2addr.ab
                        02D4  1336  ;
                        02D4  1337  ; Implicit Output:
                        02D4  1338  ;
                        02D4  1339  ;        R0<31:16> - 0
                        02D4  1340  ;        R2<31:16> - 0
                        02D4  1341  ;-
                        02D4  1342
                        02D4  1343  CMFP4:
                        02D4  1344
50   08 AE    3C        02D4  1345           MOVZWL   OPERAND_1(SP),R0         ; R0<15:0> <- src1len.rw
51   0C AE    D0        02D8  1346           MOVL     OPERAND_2(SP),R1         ; R1       <- src1addr.ab
52   10 AE    3C        02DC  1347           MOVZWL   OPERAND_3(SP),R2         ; R2<15:0> <- src2len.rw
53   14 AE    D0        02E0  1348           MOVL     OPERAND_4(SP),R3         ; R3       <- src2addr.ab
                        02E4  1349
                        02E4  1350  ; Now that the operands have been loaded, the only exception parameter
                        02E4  1351  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        02E4  1352  ; there is no reason why the state of the stack needs to be altered and we
                        02E4  1353  ; save two instructions if we leave the stack alone.
                        02E4  1354
     0425'CF   9F       02E4  1355           PUSHAB   VAX$EXIT_EMULATOR        ; Store the return PC
        FD15'  31       02E8  1356           BRW      VAX$CMPP4                ; Do the actual work
```

```
                        02EB  1360          .SUBTITLE       CVTLP - Exception handler for CVTLP instruction
                        02EB  1361  ;+
                        02EB  1362  ; Input Parameters:
                        02EB  1363  ;
                        02EB  1364  ;         OPCODE(SP)
                        02EB  1365  ;         OLD_PC(SP)
                        02EB  1366  ;         OPERAND_1(SP) - src.rl
                        02EB  1367  ;         OPERAND_2(SP) - dstlen.rw
                        02EB  1368  ;         OPERAND_3(SP) - dstaddr.ab
                        02EB  1369  ;         OPERAND_4(SP)
                        02EB  1370  ;         OPERAND_5(SP)
                        02EB  1371  ;         OPERAND_6(SP)
                        02EB  1372  ;         OPERAND_7(SP)
                        02EB  1373  ;         OPERAND_8(SP)
                        02EB  1374  ;         NEW_PC(SP)
                        02EB  1375  ;         EXCEPTION_PSL(SP)
                        02EB  1376  ;
                        02EB  1377  ; Output Parameters:
                        02EB  1378  ;
                        02EB  1379  ;         R0         - src.rl
                        02EB  1380  ;         R2<15:0> - dstlen.rw
                        02EB  1381  ;         R3         - dstaddr.ab
                        02EB  1382  ;
                        02EB  1383  ; Implicit Output:
                        02EB  1384  ;
                        02EB  1385  ;         R1         - explicitly set to zero
                        02EB  1386  ;         R2<31:16> - 0
                        02EB  1387  ;-
                        02EB  1388
                        02EB  1389  CVTLP:
                        02EB  1390
50    08 AE    D0        02EB  1391          MOVL    OPERAND_1(SP),R0        ; R0       <- src.rl
         51    D4        02EF  1392          CLRL    R1                      ; R1       <- 0
52    0C AE    3C        02F1  1393          MOVZWL  OPERAND_2(SP),R2        ; R2<15:0> <- dstlen.rw
53    10 AE    D0        02F5  1394          MOVL    OPERAND_3(SP),R3        ; R3       <- dstaddr.ab
                        02F9  1395
                        02F9  1396  ; Now that the operands have been loaded, the only exception parameter
                        02F9  1397  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        02F9  1398  ; there is no reason why the state of the stack needs to be altered and we
                        02F9  1399  ; save two instructions if we leave the stack alone.
                        02F9  1400
     0425'CF    9F       02F9  1401          PUSHAB  VAXSEXIT_EMULATOR       ; Store the return PC
         FD00'  31       02FD  1402          BRW     VAXSCVTLP               ; Do the actual work
```

```
                                0300    1406              .SUBTITLE        CVTPL - Exception handler for CVTPL instruction
                                0300    1407    ;+
                                0300    1408    ; Input Parameters:
                                0300    1409    ;
                                0300    1410    ;        OPCODE(SP)
                                0300    1411    ;        OLD_PC(SP)
                                0300    1412    ;        OPERAND_1(SP) - srclen.rw
                                0300    1413    ;        OPERAND_2(SP) - srcaddr.ab
                                0300    1414    ;        OPERAND_3(SP) - dst.wl
                                0300    1415    ;        OPERAND_4(SP)
                                0300    1416    ;        OPERAND_5(SP)
                                0300    1417    ;        OPERAND_6(SP)
                                0300    1418    ;        OPERAND_7(SP)
                                0300    1419    ;        OPERAND_8(SP)
                                0300    1420    ;        NEW_PC(SP)
                                0300    1421    ;        EXCEPTION_PSL(SP)
                                0300    1422    ;
                                0300    1423    ; Output Parameters:
                                0300    1424    ;
                                0300    1425    ;        R0<15:0> - srclen.rw
                                0300    1426    ;        R1       - srcaddr.ab
                                0300    1427    ;        R3       - dst.wl
                                0300    1428    ;
                                0300    1429    ; Notes:
                                0300    1430    ;
                                0300    1431    ;        The routine header for VAX$CVTPL describes how the destination is
                                0300    1432    ;        encoded in a register. Basically, OPERAND_3 contains the effective
                                0300    1433    ;        address of the operand. If the destination is a general register, then
                                0300    1434    ;        OPERAND_3 contains the ones complement of the register number.
                                0300    1435    ;
                                0300    1436    ; Implicit Output:
                                0300    1437    ;
                                0300    1438    ;        R0<31:16> - 0
                                0300    1439    ;        R2        - explicitly set to zero
                                0300    1440    ;-
                                0300    1441
                                0300    1442    CVTPL:
                                0300    1443
         50    08 AE    3C      0300    1444              MOVZWL   OPERAND_1(SP),R0       ; R0<15:0> <- srclen.rw
         51    0C AE    D0      0304    1445              MOVL     OPERAND_2(SP),R1       ; R1       <- srcaddr.ab
               52       D4      0308    1446              CLRL     R2                     ; R2       <- 0
         53    10 AE    D0      030A    1447              MOVL     OPERAND_3(SP),R3       ; R3       <- dst.wl
                                030E    1448
                                030E    1449    ; Now that the operands have been loaded, the only exception parameter
                                030E    1450    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                030E    1451    ; there is no reason why the state of the stack needs to be altered and we
                                030E    1452    ; save two instructions if we leave the stack alone.
                                030E    1453
         0425'CF    9F          030E    1454              PUSHAB   VAX$EXIT_EMULATOR      ; Store the return PC
            FCEB'    31         0312    1455              BRW      VAX$CVTPL              ; Do the actual work
```

N 13

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 31
V04-000   CVTPS - Exception handler for CVTPS inst   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (24)

```
                        0315  1459              .SUBTITLE        CVTPS - Exception handler for CVTPS instruction
                        0315  1460  ;+
                        0315  1461  ; Input Parameters:
                        0315  1462  ;
                        0315  1463  ;        OPCODE(SP)
                        0315  1464  ;        OLD_PC(SP)
                        0315  1465  ;        OPERAND_1(SP) - srclen.rw
                        0315  1466  ;        OPERAND_2(SP) - srcaddr.ab
                        0315  1467  ;        OPERAND_3(SP) - dstlen.rw
                        0315  1468  ;        OPERAND_4(SP) - dstaddr.ab
                        0315  1469  ;        OPERAND_5(SP)
                        0315  1470  ;        OPERAND_6(SP)
                        0315  1471  ;        OPERAND_7(SP)
                        0315  1472  ;        OPERAND_8(SP)
                        0315  1473  ;        NEW_PC(SP)
                        0315  1474  ;        EXCEPTION_PSL(SP)
                        0315  1475  ;
                        0315  1476  ; Output Parameters:
                        0315  1477  ;
                        0315  1478  ;        R0<15:0> - srclen.rw
                        0315  1479  ;        R1       - srcaddr.ab
                        0315  1480  ;        R2<15:0> - dstlen.rw
                        0315  1481  ;        R3       - dstaddr.ab
                        0315  1482  ;
                        0315  1483  ; Implicit Output:
                        0315  1484  ;
                        0315  1485  ;        R0<31:16> - 0
                        0315  1486  ;        R2<31:16> - 0
                        0315  1487  ;-
                        0315  1488
                        0315  1489  CVTPS:
                        0315  1490
50    08 AE  3C         0315  1491              MOVZWL   OPERAND_1(SP),R0      ; R0<15:0> <- srclen.rw
51    0C AE  D0         0319  1492              MOVL     OPERAND_2(SP),R1      ; R1       <- srcaddr.ab
52    10 AE  3C         031D  1493              MOVZWL   OPERAND_3(SP),R2      ; R2<15:0> <- dstlen.rw
53    14 AE  D0         0321  1494              MOVL     OPERAND_4(SP),R3      ; R3       <- dstaddr.ab
                        0325  1495
                        0325  1496  ; Now that the operands have been loaded, the only exception parameter
                        0325  1497  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0325  1498  ; there is no reason why the state of the stack needs to be altered and we
                        0325  1499  ; save two instructions if we leave the stack alone.
                        0325  1500
    0425'CF  9F         0325  1501              PUSHAB   VAX$EXIT_EMULATOR     ; Store the return PC
      FCD4'  31         0329  1502              BRW      VAX$CVTPS             ; Do the actual work
```

B 14

VAX$EMULATE   — VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 32
V04-000   CVTPT - Exception handler for CVTPT inst 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (25)

```
                          032C  1506            .SUBTITLE        CVTPT - Exception handler for CVTPT instruction
                          032C  1507 ;+
                          032C  1508 ; Input Parameters:
                          032C  1509 ;
                          032C  1510 ;         OPCODE(SP)
                          032C  1511 ;         OLD_PC(SP)
                          032C  1512 ;         OPERAND_1(SP) - srclen.rw
                          032C  1513 ;         OPERAND_2(SP) - srcaddr.ab
                          032C  1514 ;         OPERAND_3(SP) - tbladdr.ab
                          032C  1515 ;         OPERAND_4(SP) - dstlen.rw
                          032C  1516 ;         OPERAND_5(SP) - dstaddr.ab
                          032C  1517 ;         OPERAND_6(SP)
                          032C  1518 ;         OPERAND_7(SP)
                          032C  1519 ;         OPERAND_8(SP)
                          032C  1520 ;         NEW_PC(SP)
                          032C  1521 ;         EXCEPTION_PSL(SP)
                          032C  1522 ;
                          032C  1523 ; Output Parameters:
                          032C  1524 ;
                          032C  1525 ;         R0<15:0>  - srclen.rw
                          032C  1526 ;         R0<31:16> - dstlen.rw
                          032C  1527 ;         R1        - srcaddr.ab
                          032C  1528 ;         R2        - tbladdr.ab
                          032C  1529 ;         R3        - dstaddr.ab
                          032C  1530 ;-
                          032C  1531
                          032C  1532 CVTPT:
                          032C  1533
       50   14 AE   10  9C 032C  1534            ROTL    #16,OPERAND_4(SP),R0    ; R0<31:16> <- dstlen.rw
       50      08 AE  B0 0331  1535            MOVW    OPERAND_1(SP),R0        ; R0<15:0>  <- srclen.rw
       51      0C AE  D0 0335  1536            MOVL    OPERAND_2(SP),R1        ; R1        <- srcaddr.ab
       52      10 AE  D0 0339  1537            MOVL    OPERAND_3(SP),R2        ; R2        <- tbladdr.ab
       53      18 AE  D0 033D  1538            MOVL    OPERAND_5(SP),R3        ; R3        <- dstaddr.ab
                          0341  1539
                          0341  1540 ; Now that the operands have been loaded, the only exception parameter
                          0341  1541 ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          0341  1542 ; there is no reason why the state of the stack needs to be altered and we
                          0341  1543 ; save two instructions if we leave the stack alone.
                          0341  1544
          0425'CF     9F 0341  1545            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
            FCB8'     31 0345  1546            BRW     VAX$CVTPT               ; Do the actual work
```

C 14

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 33   VAX'
V04-000   CVTSP - Exception handler for CVTSP inst   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (26)   Tab

```
                        0348  1550              .SUBTITLE        CVTSP - Exception handler for CVTSP instruction
                        0348  1551  ;+
                        0348  1552  ; Input Parameters:
                        0348  1553  ;
                        0348  1554  ;        OPCODE(SP)
                        0348  1555  ;        OLD_PC(SP)
                        0348  1556  ;        OPERAND_1(SP) - srclen.rw
                        0348  1557  ;        OPERAND_2(SP) - srcaddr.ab
                        0348  1558  ;        OPERAND_3(SP) - dstlen.rw
                        0348  1559  ;        OPERAND_4(SP) - dstaddr.ab
                        0348  1560  ;        OPERAND_5(SP)
                        0348  1561  ;        OPERAND_6(SP)
                        0348  1562  ;        OPERAND_7(SP)
                        0348  1563  ;        OPERAND_8(SP)
                        0348  1564  ;        NEW_PC(SP)
                        0348  1565  ;        EXCEPTION_PSL(SP)
                        0348  1566  ;
                        0348  1567  ; Output Parameters:
                        0348  1568  ;
                        0348  1569  ;        R0<15:0> - srclen.rw
                        0348  1570  ;        R1       - srcaddr.ab
                        0348  1571  ;        R2<15:0> - dstlen.rw
                        0348  1572  ;        R3       - dstaddr.ab
                        0348  1573  ;
                        0348  1574  ; Implicit Output:
                        0348  1575  ;
                        0348  1576  ;        R0<31:16> - 0
                        0348  1577  ;        R2<31:16> - 0
                        0348  1578  ;-
                        0348  1579
                        0348  1580  CVTSP:
                        0348  1581
50    08 AE  3C         0348  1582              MOVZWL   OPERAND_1(SP),R0       ; R0<15:0> <- srclen.rw
51    0C AE  D0         034C  1583              MOVL     OPERAND_2(SP),R1       ; R1       <- srcaddr.ab
52    10 AE  3C         0350  1584              MOVZWL   OPERAND_3(SP),R2       ; R2<15:0> <- dstlen.rw
53    14 AE  D0         0354  1585              MOVL     OPERAND_4(SP),R3       ; R3       <- dstaddr.ab
                        0358  1586
                        0358  1587  ; Now that the operands have been loaded, the only exception parameter
                        0358  1588  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0358  1589  ; there is no reason why the state of the stack needs to be altered and we
                        0358  1590  ; save two instructions if we leave the stack alone.
                        0358  1591
      0425'CF  9F       0358  1592              PUSHAB   VAX$EXIT_EMULATOR      ; Store the return PC
        FCA1'  31       035C  1593              BRW      VAX$CVTSP              ; Do the actual work
```

D 14

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 34   VAX
V04-000   CVTTP - Exception handler for CVTTP inst   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (27)   V04

```
                          035F    1597              .SUBTITLE        CVTTP - Exception handler for CVTTP instruction
                          035F    1598    ;+
                          035F    1599    ; Input Parameters:
                          035F    1600    ;
                          035F    1601    ;         OPCODE(SP)
                          035F    1602    ;         OLD_PC(SP)
                          035F    1603    ;         OPERAND_1(SP) - srclen.rw
                          035F    1604    ;         OPERAND_2(SP) - srcaddr.ab
                          035F    1605    ;         OPERAND_3(SP) - tbladdr.ab
                          035F    1606    ;         OPERAND_4(SP) - dstlen.rw
                          035F    1607    ;         OPERAND_5(SP) - dstaddr.ab
                          035F    1608    ;         OPERAND_6(SP)
                          035F    1609    ;         OPERAND_7(SP)
                          035F    1610    ;         OPERAND_8(SP)
                          035F    1611    ;         NEW_PC(SP)
                          035F    1612    ;         EXCEPTION_PSL(SP)
                          035F    1613    ;
                          035F    1614    ; Output Parameters:
                          035F    1615    ;
                          035F    1616    ;         R0<15:0>  - srclen.rw
                          035F    1617    ;         R0<31:16> - dstlen.rw
                          035F    1618    ;         R1        - srcaddr.ab
                          035F    1619    ;         R2        - tbladdr.ab
                          035F    1620    ;         R3        - dstaddr.ab
                          035F    1621    ;-
                          035F    1622
                          035F    1623    CVTTP:
                          035F    1624
50     14 AE    10    9C  035F    1625              ROTL     #16,OPERAND_4(SP),R0   ; R0<31:16> <- dstlen.rw
       50    08 AE    B0  0364    1626              MOVW     OPERAND_1(SP),R0       ; R0<15:0>  <- srclen.rw
       51    0C AE    D0  0368    1627              MOVL     OPERAND_2(SP),R1       ; R1        <- srcaddr.ab
       52    10 AE    D0  036C    1628              MOVL     OPERAND_3(SP),R2       ; R2        <- tbladdr.ab
       53    18 AE    D0  0370    1629              MOVL     OPERAND_5(SP),R3       ; R3        <- dstaddr.ab
                          0374    1630
                          0374    1631    ; Now that the operands have been loaded, the only exception parameter
                          0374    1632    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          0374    1633    ; there is no reason why the state of the stack needs to be altered and we
                          0374    1634    ; save two instructions if we leave the stack alone.
                          0374    1635
        0425'CF    9F    0374    1636              PUSHAB   VAX$EXIT_EMULATOR      ; Store the return PC
        FC85'      31    0378    1637              BRW      VAX$CVTTP              ; Do the actual work
```

E 14

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 35
V04-000   DIVP - Exception handler for DIVP instru   5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (28)

VAX
V04

```
                              037B   1641           .SUBTITLE        DIVP - Exception handler for DIVP instruction
                              037B   1642   ;+
                              037B   1643   ; Input Parameters:
                              037B   1644   ;
                              037B   1645   ;         OPCODE(SP)
                              037B   1646   ;         OLD_PC(SP)
                              037B   1647   ;         OPERAND_1(SP) - divrlen.rw
                              037B   1648   ;         OPERAND_2(SP) - divraddr.ab
                              037B   1649   ;         OPERAND_3(SP) - divdlen.rw
                              037B   1650   ;         OPERAND_4(SP) - divdaddr.ab
                              037B   1651   ;         OPERAND_5(SP) - quolen.rw
                              037B   1652   ;         OPERAND_6(SP) - quoaddr.ab
                              037B   1653   ;         OPERAND_7(SP)
                              037B   1654   ;         OPERAND_8(SP)
                              037B   1655   ;         NEW_PC(SP)
                              037B   1656   ;         EXCEPTION_PSL(SP)
                              037B   1657   ;
                              037B   1658   ; Output Parameters:
                              037B   1659   ;
                              037B   1660   ;         R0<15:0> - divrlen.rw
                              037B   1661   ;         R1       - divraddr.ab
                              037B   1662   ;         R2<15:0> - divdlen.rw
                              037B   1663   ;         R3       - divdaddr.ab
                              037B   1664   ;         R4<15:0> - quolen.rw
                              037B   1665   ;         R5       - quoaddr.ab
                              037B   1666   ;
                              037B   1667   ; Implicit Output:
                              037B   1668   ;
                              037B   1669   ;         R0<31:16> - 0
                              037B   1670   ;         R2<31:16> - 0
                              037B   1671   ;         R4<31:16> - 0
                              037B   1672   ;-
                              037B   1673
                              037B   1674   DIVP:
                              037B   1675
50    08 AE   3C              037B   1676           MOVZWL   OPERAND_1(SP),R0          ; R0<15:0> <- divrlen.rw
51    0C AE   D0              037F   1677           MOVL     OPERAND_2(SP),R1          ; R1       <- divraddr.ab
52    10 AE   3C              0383   1678           MOVZWL   OPERAND_3(SP),R2          ; R2<15:0> <- divdlen.rw
53    14 AE   D0              0387   1679           MOVL     OPERAND_4(SP),R3          ; R3       <- divdaddr.ab
54    18 AE   3C              038B   1680           MOVZWL   OPERAND_5(SP),R4          ; R4<15:0> <- quolen.rw
55    1C AE   D0              038F   1681           MOVL     OPERAND_6(SP),R5          ; R5       <- quoaddr.ab
                              0393   1682
                              0393   1683   ; Now that the operands have been loaded, the only exception parameter
                              0393   1684   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              0393   1685   ; there is no reason why the state of the stack needs to be altered and we
                              0393   1686   ; save two instructions if we leave the stack alone.
                              0393   1687
      0425'CF   9F           0393   1688           PUSHAB   VAX$EXIT_EMULATOR         ; Store the return PC
         FC66'  31           0397   1689           BRW      VAX$DIVP                  ; Do the actual work
```

F 14

VAXEMULATE   – VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 36
V04-000   MOVP – Exception handler for MOVP instru 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (29)

VAX
V04

```
                        039A  1693              .SUBTITLE       MOVP - Exception handler for MOVP instruction
                        039A  1694    :+
                        039A  1695    ; Input Parameters:
                        039A  1696    ;
                        039A  1697    ;       OPCODE(SP)
                        039A  1698    ;       OLD_PC(SP)
                        039A  1699    ;       OPERAND_1(SP) - len.rw
                        039A  1700    ;       OPERAND_2(SP) - srcaddr.ab
                        039A  1701    ;       OPERAND_3(SP) - dstaddr.ab
                        039A  1702    ;       OPERAND_4(SP)
                        039A  1703    ;       OPERAND_5(SP)
                        039A  1704    ;       OPERAND_6(SP)
                        039A  1705    ;       OPERAND_7(SP)
                        039A  1706    ;       OPERAND_8(SP)
                        039A  1707    ;       NEW_PC(SP)
                        039A  1708    ;       EXCEPTION_PSL(SP)
                        039A  1709    ;
                        039A  1710    ; Output Parameters:
                        039A  1711    ;
                        039A  1712    ;       R0<15:0> - len.rw
                        039A  1713    ;       R1       - srcaddr.ab
                        039A  1714    ;       R3       - dstaddr.ab
                        039A  1715    ;
                        039A  1716    ; Implicit Output:
                        039A  1717    ;
                        039A  1718    ;       R0<31:16> - 0
                        039A  1719    ;       R2        - UNPREDICTABLE
                        039A  1720    :-
                        039A  1721
                        039A  1722    MOVP:
                        039A  1723
50    08 AE   3C        039A  1724              MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- len.rw
51    0C AE   D0        039E  1725              MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
53    10 AE   D0        03A2  1726              MOVL    OPERAND_3(SP),R3        ; R3       <- dstaddr.ab
                        03A6  1727
                        03A6  1728    ; Now that the operands have been loaded, the only exception parameter
                        03A6  1729    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        03A6  1730    ; there is no reason why the state of the stack needs to be altered and we
                        03A6  1731    ; save two instructions if we leave the stack alone.
                        03A6  1732
                        03A6  1733    ; The MOVP instruction is the only instruction in this entire set that
                        03A6  1734    ; preserves the setting of the C-bit. The C-bit setting in the saved PSL
                        03A6  1735    ; is propogated into the current PSL because the current PSL forms the
                        03A6  1736    ; initial setting for the final settings of the condition codes.
                        03A6  1737
            01  B9      03A6  1738              BICPSW  #PSL$M_C                ; Assume C bit is clear
                        03A8  1739
                        03A8  1740              ASSUME PSL$V_C EQ 0            ; Make sure that BLBC is OK
                        03A8  1741
02 2C AE    E9          03A8  1742              BLBC    EXCEPTION_PSL(SP),10$   ; Skip next if saved C-bit is clear
            01  B8      03AC  1743              BISPSW  #PSL$M_C                ; Otherwise, set the C-bit
                        03AE  1744
                        03AE  1745    ; Note that it is crucial that no instructions that alter the C-bit can
                        03AE  1746    ; execute until the PSL is saved in VAX$MOVP. PUSHAB preserves the C-bit.
                        03AE  1747
0425'CF     9F          03AE  1748    10$:      PUSHAB  V.X$EXIT_EMULATOR       ; Store the return PC
   FC4B'    31          03B2  1749              BRW     VAX$MOVP                ; Do the actual work
```

VAXSEMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 37
V04-000   MULP - Exception handler for MULP instru 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (30)

G 14

```
                              03B5  1753                  .SUBTITLE          MULP - Exception handler for MULP instruction
                              03B5  1754  ;+
                              03B5  1755  ; Input Parameters:
                              03B5  1756  ;
                              03B5  1757  ;       OPCODE(SP)
                              03B5  1758  ;       OLD_PC(SP)
                              03B5  1759  ;       OPERAND_1(SP) - mulrlen.rw
                              03B5  1760  ;       OPERAND_2(SP) - mulraddr.ab
                              03B5  1761  ;       OPERAND_3(SP) - muldlen.rw
                              03B5  1762  ;       OPERAND_4(SP) - muldaddr.ab
                              03B5  1763  ;       OPERAND_5(SP) - prodlen.rw
                              03B5  1764  ;       OPERAND_6(SP) - prodaddr.ab
                              03B5  1765  ;       OPERAND_7(SP)
                              03B5  1766  ;       OPERAND_8(SP)
                              03B5  1767  ;       NEW_PC(SP)
                              03B5  1768  ;       EXCEPTION_PSL(SP)
                              03B5  1769  ;
                              03B5  1770  ; Output Parameters:
                              03B5  1771  ;
                              03B5  1772  ;       R0<15:0> - mulrlen.rw
                              03B5  1773  ;       R1       - mulraddr.ab
                              03B5  1774  ;       R2<15:0> - muldlen.rw
                              03B5  1775  ;       R3       - muldaddr.ab
                              03B5  1776  ;       R4<15:0> - prodlen.rw
                              03B5  1777  ;       R5       - prodaddr.ab
                              03B5  1778  ;
                              03B5  1779  ; Implicit Output:
                              03B5  1780  ;
                              03B5  1781  ;       R0<31:16> - 0
                              03B5  1782  ;       R2<31:16> - 0
                              03B5  1783  ;       R4<31:16> - 0
                              03B5  1784  ;-
                              03B5  1785
                              03B5  1786  MULP:
                              03B5  1787
      50   08 AE   3C         03B5  1788          MOVZWL  OPERAND_1(SP),R0          ; R0<15:0> <- mulrlen.rw
      51   0C AE   D0         03B9  1789          MOVL    OPERAND_2(SP),R1          ; R1       <- mulraddr.ab
      52   10 AE   3C         03BD  1790          MOVZWL  OPERAND_3(SP),R2          ; R2<15:0> <- muldlen.rw
      53   14 AE   D0         03C1  1791          MOVL    OPERAND_4(SP),R3          ; R3       <- muldaddr.ab
      54   18 AE   3C         03C5  1792          MOVZWL  OPERAND_5(SP),R4          ; R4<15:0> <- prodlen.rw
      55   1C AE   D0         03C9  1793          MOVL    OPERAND_6(SP),R5          ; R5       <- prodaddr.ab
                              03CD  1794
                              03CD  1795  ; Now that the operands have been loaded, the only exception parameter
                              03CD  1796  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              03CD  1797  ; there is no reason why the state of the stack needs to be altered and we
                              03CD  1798  ; save two instructions if we leave the stack alone.
                              03CD  1799
         0425'CF   9F         03CD  1800          PUSHAB  VAX$EXIT_EMULATOR         ; Store the return PC
            FC2C'  31         03D1  1801          BRW     VAX$MULP                  ; Do the actual work
```

H 14

```
                        03D4   1805              .SUBTITLE         SUBP4 – Exception handler for SUBP4 instruction
                        03D4   1806   ;+
                        03D4   1807   ; Input Parameters:
                        03D4   1808   ;
                        03D4   1809   ;         OPCODE(SP)
                        03D4   1810   ;         OLD_PC(SP)
                        03D4   1811   ;         OPERAND_1(SP) – sublen.rw
                        03D4   1812   ;         OPERAND_2(SP) – subaddr.ab
                        03D4   1813   ;         OPERAND_3(SP) – diflen.rw
                        03D4   1814   ;         OPERAND_4(SP) – difaddr.ab
                        03D4   1815   ;         OPERAND_5(SP)
                        03D4   1816   ;         OPERAND_6(SP)
                        03D4   1817   ;         OPERAND_7(SP)
                        03D4   1818   ;         OPERAND_8(SP)
                        03D4   1819   ;         NEW_PC(SP)
                        03D4   1820   ;         EXCEPTION_PSL(SP)
                        03D4   1821   ;
                        03D4   1822   ; Output Parameters:
                        03D4   1823   ;
                        03D4   1824   ;         R0<15:0> – sublen.rw
                        03D4   1825   ;         R1        – subaddr.ab
                        03D4   1826   ;         R2<15:0> – diflen.rw
                        03D4   1827   ;         R3        – difaddr.ab
                        03D4   1828   ;
                        03D4   1829   ; Implicit Output:
                        03D4   1830   ;
                        03D4   1831   ;         R0<31:16> – 0
                        03D4   1832   ;         R2<31:16> – 0
                        03D4   1833   ;-
                        03D4   1834
                        03D4   1835   SUBP4:
                        03D4   1836
  50    08 AE   3C      03D4   1837              MOVZWL    OPERAND_1(SP),R0      ; R0<15:0> <– sublen.rw
  51    0C AE   D0      03D8   1838              MOVL      OPERAND_2(SP),R1      ; R1        <– subaddr.ab
  52    10 AE   3C      03DC   1839              MOVZWL    OPERAND_3(SP),R2      ; R2<15:0> <– diflen.rw
  53    14 AE   D0      03E0   1840              MOVL      OPERAND_4(SP),R3      ; R3        <– difaddr.ab
                        03E4   1841
                        03E4   1842   ; Now that the operands have been loaded, the only exception parameter
                        03E4   1843   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        03E4   1844   ; there is no reason why the state of the stack needs to be altered and we
                        03E4   1845   ; save two instructions if we leave the stack alone.
                        03E4   1846
  0425'CF   9F          03E4   1847              PUSHAB    VAX$EXIT_EMULATOR     ; Store the return PC
     FC15'  31          03E8   1848              BRW       VAX$SUBP4             ; Do the actual work
```

I 14

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10 VAX/VMS Macro V04-00   Page 39   VAX
V04-000   SUBP6 - Exception handler for SUBP6 inst   5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (32)   V04

```
                        03EB    1852            .SUBTITLE       SUBP6 - Exception handler for SUBP6 instruction
                        03EB    1853    ;+
                        03EB    1854    ; Input Parameters:
                        03EB    1855    ;
                        03EB    1856    ;       OPCODE(SP)
                        03EB    1857    ;       OLD_PC(SP)
                        03EB    1858    ;       OPERAND_1(SP) - sublen.rw
                        03EB    1859    ;       OPERAND_2(SP) - subaddr.ab
                        03EB    1860    ;       OPERAND_3(SP) - minlen.rw
                        03EB    1861    ;       OPERAND_4(SP) - minaddr.ab
                        03EB    1862    ;       OPERAND_5(SP) - diflen.rw
                        03EB    1863    ;       OPERAND_6(SP) - difaddr.ab
                        03EB    1864    ;       OPERAND_7(SP)
                        03EB    1865    ;       OPERAND_8(SP)
                        03EB    1866    ;       NEW_PC(SP)
                        03EB    1867    ;       EXCEPTION_PSL(SP)
                        03EB    1868    ;
                        03EB    1869    ; Output Parameters:
                        03EB    1870    ;
                        03EB    1871    ;       R0<15:0> - sublen.rw
                        03EB    1872    ;       R1       - subaddr.ab
                        03EB    1873    ;       R2<15:0> - minlen.rw
                        03EB    1874    ;       R3       - minaddr.ab
                        03EB    1875    ;       R4<15:0> - diflen.rw
                        03EB    1876    ;       R5       - difaddr.ab
                        03EB    1877    ;
                        03EB    1878    ; Implicit Output:
                        03EB    1879    ;
                        03EB    1880    ;       R0<31:16> - 0
                        03EB    1881    ;       R2<31.16> - 0
                        03EB    1882    ;       R4<31:16> - 0
                        03EB    1883    ;-
                        03EB    1884
                        03EB    1885    SUBP6:
                        03EB    1886
50      08 AE   3C      03EB    1887            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- sublen.rw
51      0C AE   D0      03EF    1888            MOVL    OPERAND_2(SP),R1        ; R1       <- subaddr.ab
52      10 AE   3C      03F3    1889            MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- minlen.rw
53      14 AE   D0      03F7    1890            MOVL    OPERAND_4(SP),R3        ; R3       <- minaddr.ab
54      18 AE   3C      03FB    1891            MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- diflen.rw
55      1C AE   D0      03FF    1892            MOVL    OPERAND_6(SP),R5        ; R5       <- difaddr.ab
                        0403    1893
                        0403    1894    ; Now that the operands have been loaded, the only exception parameter
                        0403    1895    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0403    1896    ; there is no reason why the state of the stack needs to be altered and we
                        0403    1897    ; save two instructions if we leave the stack alone.
                        0403    1898
    0425'CF     9F      0403    1899            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
       FBF6'    31      0407    1900            BRW     VAX$SUBP6               ; Do the actual work
```

VAX$EMULATE
V04-000

J 14

- VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 40
EDITPC - Exception handler for EDITPC in 5-SEP-1984 00:45:28 [EMULAT.SRC]VAXEMULAT.MAR;1   (33)

VAX
V04

```
                          040A    1904              .SUBTITLE         EDITPC - Exception handler for EDITPC instruction
                          040A    1905    ;+
                          040A    1906    ; Input Parameters:
                          040A    1907    ;
                          040A    1908    ;         OPCODE(SP)
                          040A    1909    ;         OLD_PC(SP)
                          040A    1910    ;         OPERAND_1(SP) - srclen.rw
                          040A    1911    ;         OPERAND_2(SP) - srcaddr.ab
                          040A    1912    ;         OPERAND_3(SP) - pattern.ab
                          040A    1913    ;         OPERAND_4(SP) - dstaddr.ab
                          040A    1914    ;         OPERAND_5(SP)
                          040A    1915    ;         OPERAND_6(SP)
                          040A    1916    ;         OPERAND_7(SP)
                          040A    1917    ;         OPERAND_8(SP)
                          040A    1918    ;         NEW_PC(SP)
                          040A    1919    ;         EXCEPTION_PSL(SP)
                          040A    1920    ;
                          040A    1921    ; Output Parameters:
                          040A    1922    ;
                          040A    1923    ;         R0<15:0> - srclen.rw
                          040A    1924    ;         R1         - srcaddr.ab
                          040A    1925    ;         R3         - pattern.ab
                          040A    1926    ;         R5         - dstaddr.ab
                          040A    1927    ;
                          040A    1928    ; Implicit Output:
                          040A    1929    ;
                          040A    1930    ;         R0<31:16> - 0
                          040A    1931    ;         R2         - explicitly set to zero
                          040A    1932    ;         R4         - explicitly set to zero
                          040A    1933    ;-
                          040A    1934
                          040A    1935    EDITPC:
                          040A    1936
50    08 AE    3C         040A    1937              MOVZWL   OPERAND_1(SP),R0       ; R0<15:0> <- srclen.rw
51    0C AE    D0         040E    1938              MOVL     OPERAND_2(SP),R1       ; R1         <- srcaddr.ab
         52    D4         0412    1939              CLRL     R2                     ; R2         <- 0
53    10 AE    D0         0414    1940              MOVL     OPERAND_3(SP),R3       ; R3         <- pattern.ab
         54    D4         0418    1941              CLRL     R4                     ; R4         <- 0
55    14 AE    D0         041A    1942              MOVL     OPERAND_4(SP),R5       ; R5         <- dstaddr.ab
                          041E    1943
                          041E    1944    ; Now that the operands have been loaded, the only exception parameter
                          041E    1945    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          041E    1946    ; there is no reason why the state of the stack needs to be altered and we
                          041E    1947    ; save two instructions if we leave the stack alone.
                          041E    1948
       0425'CF    9F      041E    1949              PUSHAB   VAX$EXIT_EMULATOR      ; Store the return PC
        FBDB'     31      0422    1950              BRW      VAX$EDITPC             ; Do the actual work
```

VAXSEMULATE
V04-000

K 14

- VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 41
Common Exit Path for VAX$xxxxxx Routines   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (34)

VAX
V04

```
                        0425  1953              .SUBTITLE        Common Exit Path for VAX$xxxxxx Routines
                        0425  1954  ;+
                        0425  1955  ; Functional Description:
                        0425  1956  ;
                        0425  1957  ;       This is the common exit path for all instruction-specific routines.
                        0425  1958  ;       The condition codes returned by the VAX$xxxxxx routine are stored in
                        0425  1959  ;       the exception PSL and control is passed back to the instruction stream
                        0425  1960  ;       that executed the reserved instruction.
                        0425  1961  ;
                        0425  1962  ; Input Parameters:
                        0425  1963  ;
                        0425  1964  ;  ,      PSL contains condition code settings from VAX$xxxxxx routine.
                        0425  1965  ;
                        0425  1966  ;       OPCODE(SP)     - Opcode of reserved instruction
                        0425  1967  ;       OLD_PC(SP)     - PC of reserved instruction
                        0425  1968  ;       OPERAND_1(SP) - First operand specifier
                        0425  1969  ;       OPERAND_2(SP) - Second operand specifier
                        0425  1970  ;       OPERAND_3(SP) - Third operand specifier
                        0425  1971  ;       OPERAND_4(SP) - Fourth operand specifier
                        0425  1972  ;       OPERAND_5(SP) - Fifth operand specifier
                        0425  1973  ;       OPERAND_6(SP) - Sixth operand specifier
                        0425  1974  ;       OPERAND_7(SP) - Seventh operand specifier (currently unused)
                        0425  1975  ;       OPERAND_8(SP) - Eight operand specifier (currently unused)
                        0425  1976  ;       NEW_PC(SP)     - PC of instruction following reserved instruction
                        0425  1977  ;       EXCEPTION_PSL(SP) - PSL at time of exception
                        0425  1978  ;
                        0425  1979  ; Implicit Input:
                        0425  1980  ;
                        0425  1981  ;       General registers contain architecturally specified values according
                        0425  1982  ;       to specific instruction that was emulated.
                        0425  1983  ;
                        0425  1984  ; Implicit Output:
                        0425  1985  ;
                        0425  1986  ;       Control is passed to the location designated by "new PC" with the
                        0425  1987  ;       condition codes as determined by VAX$xxxxxx. The EXIT routine also
                        0425  1988  ;       preserves general registers.
                        0425  1989  ;-
                        0425  1990
                        0425  1991  VAX$EXIT_EMULATOR::
              7E    DC  0425  1992              MOVPSL    -(SP)                        ; Save the new PSL on the stack
                        0427  1993
                        0427  1994  ; Note that the next instruction makes no assumptions about the condition
                        0427  1995  ; codes in the saved PSL.
                        0427  1996
04    00  8E  F0  0427  1997              INSV      (SP)+,#0,#4,-
          2C  AE      042B  1998                        EXCEPTION_PSL(SP)      ; Replace saved condition codes
      5E  28  C0      042D  1999              ADDL      #NEW_PC,SP              ; Adjust stack pointer (discard old PC)
              02      0430  2000              REI                              ; Return
                        0431  2001
                        0431  2002              .END
```

Symbol	Value			Symbol	Value		
...OFFSET	= 00000040			OP$_ADDP4	= 00000020		
...OPCODE	= 00000038			OP$_ADDP6	= 00000021		
ADDP4	0000026A	R	02	OP$_ASHP	= 000000F8		
ADDP6	00000281	R	02	OP$_CMPC3	= 00000029		
ASHP	000002A0	R	02	OP$_CMPC5	= 0000002D		
CASE_TABLE_BASE	00000006	R	02	OP$_CMPP3	= 00000035		
CASE_TABLE_SIZE	= 00000044			OP$_CMPP4	= 00000037		
CMPC3	000001B7	R	02	OP$_CRC	= 0000000B		
CMPC5	000001CA	R	02	OP$_CVTLP	= 000000F9		
CMPP3	000002C1	R	02	OP$_CVTPL	= 00000036		
CMPP4	000002D4	R	02	OP$_CVTPS	= 00000008		
CRC	00000253	R	02	OP$_CVTPT	= 00000024		
CVTLP	000002EB	R	02	OP$_CVTSP	= 00000009		
CVTPL	00000300	R	02	OP$_CVTTP	= 00000026		
CVTPS	00000315	R	02	OP$_DIVP	= 00000027		
CVTPT	0000032C	R	02	OP$_EDITPC	= 00000038		
CVTSP	00000348	R	02	OP$_LOCC	= 0000003A		
CVTTP	0000035F	R	02	OP$_MATCHC	= 00000039		
DIVP	0000037B	R	02	OP$_MOVP	= 00000034		
EDITPC	0000040A	R	02	OP$_MOVTC	= 0000002E		
EXCEPTION_PSL	= 0000002C			OP$_MOVTUC	= 0000002F		
FPD_CASE_TABLE_BASE	000000D6	R	02	OP$_MULP	= 00000025		
INCLUDE_ADDP4	= 00000000			OP$_SCANC	= 0000002A		
INCLUDE_ADDP6	= 00000000			OP$_SKPC	= 0000003B		
INCLUDE_ASHP	= 00000000			OP$_SPANC	= 0000002B		
INCLUDE_CMPC3	= 00000000			OP$_SUBP4	= 00000022		
INCLUDE_CMPC5	= 00000000			OP$_SUBP6	= 00000023		
INCLUDE_CMPP3	= 00000000			OPCODE	= 00000000		
INCLUDE_CMPP4	= 00000000			OPCODE_BASE	= FFFFFFF8		
INCLUDE_CRC	= 00000000			OPCODE_MAX	= 0000003B		
INCLUDE_CVTLP	= 00000000			OPERAND_1	= 00000008		
INCLUDE_CVTPL	= 00000000			OPERAND_2	= 0000000C		
INCLUDE_CVTPS	= 00000000			OPERAND_3	= 00000010		
INCLUDE_CVTPT	= 00000000			OPERAND_4	= 00000014		
INCLUDE_CVTSP	= 00000000			OPERAND_5	= 00000018		
INCLUDE_CVTTP	= 00000000			OPERAND_6	= 0000001C		
INCLUDE_DIVP	= 00000000			OPERAND_8	= 00000024		
INCLUDE_EDITPC	= 00000000			PSL$M_C	= 00000001		
INCLUDE_LOCC	= 00000000			PSL$V_C	= 00000000		
INCLUDE_MATCHC	= 00000000			PSL$V_FPD	= 0000001B		
INCLUDE_MOVP	= 00000000			SCANC	000001E6	R	02
INCLUDE_MOVTC	= 00000000			SKPC	00000228	R	02
INCLUDE_MOVTUC	= 00000000			SPANC	000001FD	R	02
INCLUDE_MULP	= 00000000			SUBP4	000003D4	R	02
INCLUDE_SCANC	= 00000000			SUBP6	000003EB	R	02
INCLUDE_SKPC	= 00000000			VAX$ADDP4	********	X	02
INCLUDE_SPANC	= 00000000			VAX$ADDP6	********	X	02
INCLUDE_SUBP4	= 00000000			VAX$AL_DELTA_PC_TABLE	********	X	00
INCLUDE_SUBP6	= 00000000			VAX$ASHP	********	X	02
LOCC	00000214	R	02	VAX$CMPC3	********	X	02
MATCHC	0000023C	R	02	VAX$CMPC5	********	X	02
MOVP	0000039A	R	02	VAX$CMPP3	********	X	02
MOVTC	00000179	R	02	VAX$CMPP4	********	X	02
MOVTUC	00000198	R	02	VAX$CRC	********	X	02
MULP	000003B5	R	02	VAX$CVTLP	********	X	02
NEW_PC	= 00000028			VAX$CVTLP_RESTART	********	X	02
OLD_PC	= 00000004			VAX$CVTPL	********	X	02

M 14

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 43
Symbol table   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (34)

VAX
V04

```
VAX$CVTPL_RESTART         ********   X   02
VAX$CVTPS                 ********   X   02
VAX$CVTPT                 ********   X   02
VAX$CVTPT_RESTART         ********   X   02
VAX$CVTSP                 ********   X   02
VAX$CVTTP                 ********   X   02
VAX$CVTTP_RESTART         ********   X   02
VAX$DIVP                  ********   X   02
VAX$EDITPC                ********   X   02
VAX$EDITPC_RESTART        ********   X   02
VAX$EMULATE               00000000  RG  02
VAX$EMULATE_FPD           0000009C  RG  02
VAX$EXIT_EMULATOR         00000425  RG  02
VAX$LOCC                  ********   X   02
VAX$MATCHC                ********   X   02
VAX$MOVP                  ********   X   02
VAX$MOVTC                 ********   X   02
VAX$MOVTUC                ********   X   02
VAX$MULP                  ********   X   02
VAX$REFLECT_TO_VMS        ********   X   00
VAX$SCANC                 ********   X   02
VAX$SKPC                  ********   X   02
VAX$SPANC                 ********   X   02
VAX$SUBP4                 ********   X   02
VAX$SUBP6                 ********   X   02
VAX$_OPCDEC               ********   X   00
VAX$_OPCDEC_FPD           ********   X   00
```

```
                      +-----------------+
                      ! 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	00000431 (	1073.)	02 (	2.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	QUAD

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.05	00:00:00.76
Command processing	84	00:00:00.57	00:00:05.27
Pass 1	271	00:00:09.05	00:00:29.52
Symbol table sort	0	00:00:00.71	00:00:03.06
Pass 2	316	00:00:04.15	00:00:12.58
Symbol table output	16	00:00:00.12	00:00:00.39
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	698	00:00:14.67	00:00:51.60

The working set limit was 1500 pages.
66591 bytes (131 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 532 non-local and 4 local symbols.

N 14

VAX$EMULATE   - VAX-11 Instruction Emulator   16-SEP-1984 01:29:10   VAX/VMS Macro V04-00   Page 44   VAX
VAX-11 Macro Run Statistics   5-SEP-1984 00:45:28   [EMULAT.SRC]VAXEMULAT.MAR;1   (34)   V04

2002 source lines were read in Pass 1, producing 21 object records in Pass 2.
16 pages of virtual memory were used to define 14 macros.

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

Macro library name   Macros defined
------------------   --------------
_255DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1   3
_255DUA28:[SYSLIB]STARLET.MLB;2   6
TOTALS (all libraries)   9

524 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:VAXEMULAT/OBJ=OBJ$:VAXEMULAT MSRC$:VAXEMULAT/UPDATE=(ENH$:VAXEMULAT)+LIB$:VAXMACROS/LIB

VAXHANDLR
LIS

VAXCVTLP
LIS

VAXDECIML
LIS

VAXEMULAT
LIS

VAXCVTPL
LIS

VAXEDITPC
LIS