```
EEEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEE               MMMMMMM   MMMMMMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMMMMMM   MMMMMMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMMMMMM   MMMMMMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM MMM MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM MMM MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM MMM MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEEEEEEEEEEE      MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEEEEEEEEEEE      MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEEEEEEEEEEE      MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEE               MMM     MMM  UUU         UUU  LLL              AAA         AAA         TTT
EEEEEEEEEEEEEEEE  MMM     MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA         TTT
EEEEEEEEEEEEEEEE  MMM     MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA         TTT
EEEEEEEEEEEEEEEE  MMM     MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA         TTT
```

```
BBBBBBBB      OOOOOO      OOOOOO    EEEEEEEEEE  MM      MM  UU      UU  LL            AAAAAA    TTTTTTTTTT
BBBBBBBB      OOOOOO      OOOOOO    EEEEEEEEEE  MM      MM  UU      UU  LL            AAAAAA    TTTTTTTTTT
BB      BB  OO      OO  OO      OO  EE          MMMM  MMMM  UU      UU  LL          AA      AA      TT
BB      BB  OO      OO  OO      OO  EE          MMMM  MMMM  UU      UU  LL          AA      AA      TT
BB      BB  OO      OO  OO      OO  EE          MM  MM  MM  UU      UU  LL          AA      AA      TT
BB      BB  OO      OO  OO      OO  EE          MM  MM  MM  UU      UU  LL          AA      AA      TT
BBBBBBBB    OO      OO  OO      OO  EEEEEEEE    MM      MM  UU      UU  LL          AA      AA      TT
BBBBBBBB    OO      OO  OO      OO  EEEEEEEE    MM      MM  UU      UU  LL          AA      AA      TT
BB      BB  OO      OO  OO      OO  EE          MM      MM  UU      UU  LL          AAAAAAAAAA      TT
BB      BB  OO      OO  OO      OO  EE          MM      MM  UU      UU  LL          AAAAAAAAAA      TT
BB      BB  OO      OO  OO      OO  EE          MM      MM  UU      UU  LL          AA      AA      TT
BB      BB  OO      OO  OO      OO  EE          MM      MM  UU      UU  LL          AA      AA      TT
BBBBBBBB      OOOOOO      OOOOOO    EEEEEEEEEE  MM      MM  UUUUUUUUUU  LLLLLLLLLL  AA      AA      TT
BBBBBBBB      OOOOOO      OOOOOO    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      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

....
....
....
....

Table of contents

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

```
0000     1          .NOSHOW CONDITIONALS
0000     5          .TITLE  BOO$EMULATE - Subset VAX-11 Instruction Emulator for VMB
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 ;       This 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 ;--
```

```
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   176
0000   177          .EXTERNAL       VAX$_OPCDEC, -
0000   178                          VAX$_OPCDEC_FPD
0000   179
```

```
                        ; PSECT Declarations:

                        .DEFAULT        DISPLACEMENT , WORD

                        .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, QUAD

                        .NOSHOW         CONDITIONALS
```

```
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.
0000    244 ;
```

```
                     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   291 ;       OPCODE_BASE = OP$_CMPC3         ; Smallest (in signed sense) opcode
                     0000   292
                     0000   293         SIGN_EXTEND     OP$_CMPC3 , OPCODE_BASE
                     0000   299
            00000013 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
 12   29   6E   8F  0000   306         CASEB   OPCODE(SP),#OPCODE_BASE,#<OPCODE_MAX-OPCODE_BASE>
                     0004   307
```

```
                     0004    308          INIT_CASE_TABLE CASE_TABLE_SIZE,CASE_TABLE_BASE,10$
                     002A    309
                     002A    310 ; If we drop through the case dispatcher, then the fault was not caused
                     002A    311 ; by executing one of the instructions supported by this emulator. Such
                     002A    312 ; exceptions will simply be passed through to VMS. (In the bootstrap emulator,
                     002A    313 ; there is no operating system to reflect the exception. We simply HALT.)
                     002A    314
00000000'8F   DD     002A    315 10$:     PUSHL   #VAX$_OPCDEC                ; Store signal name
           0D   DD   0030    316          PUSHL   #13                         ; Total of 13 longwords in signal array
                     0032    317
                00   0032    319          HALT                                ; Nothing else we can do
```

```
                        .SUBTITLE       VAX$EMULATE_FPD - Alternate Entry Path into Emulator
                ;+
                ; Functional Description:
                ;
                ;       This routine is entered through the ^XCC(SCB) exception vector when an
                ;       instruction that is not a part of the microVAX architecture executes
                ;       and the FPD bit is set in the PSL. The software state that was
                ;       preserved by each instruction must be restored and instruction
                ;       execution resumed. Access mode and IPL are preserved across the
                ;       exception occurrence.
                ;
                ;       Before the various VAX$xxxxxx (or VAX$xxxxxx_RESTART) routines regain
                ;       control, this dispatcher must retrieve the delta PC from wherever
                ;       it was stored and place the stack in the same state that it is in
                ;       when the normal (FPD bit not set) instruction dispatcher passes
                ;       control to the various VAX$xxxxxx routines. The pictures below explain
                ;       this.
                ;
                ; Input Parameters:
                ;
                ;       00(SP) - PC of reserved instruction
                ;       04(SP) - PSL at time of exception
                ;
                ; Output Parameters:
                ;
                ;       The following picture shows the state of the stack after the dispatcher
                ;       has executed its preliminary code but before control is passed back to
                ;       instruction-specific  execution. Note that this routine makes the
                ;       stack look like it does when a reserved instruction executes and FPD
                ;       is not yet set. This is done to make the exception exit code independent
                ;       of whether a different exception exception occurred while the emulator
                ;       was running.
                ;
                ;       00(SP) - Return PC (Address of EXIT routine in this module)
                ;       04(SP) - Unused placeholder (OPCODE)
                ;       08(SP) - PC of reserved instruction (old PC)
                ;       12(SP) - Unused placeholder (OPERAND_1)
                ;       16(SP) - Unused placeholder (OPERAND_2)
                ;       20(SP) - Unused placeholder (OPERAND_3)
                ;       24(SP) - Unused placeholder (OPERAND_4)
                ;       28(SP) - Unused placeholder (OPERAND_5)
                ;       32(SP) - Unused placeholder (OPERAND_6)
                ;       36(SP) - Unused placeholder (OPERAND_7)
                ;       40(SP) - Unused placeholder (OPERAND_8)
                ;       44(SP) - PC of instruction following reserved instruction (new PC)
                ;       48(SP) - PSL at time of exception
                ;
                ;       Before this routine dispatches to opcode-specific code, it calculates
                ;       the PC of the next instruction based on the PC of the reserved
                ;       instruction and the delta-PC quantity that was stored as part of the
                ;       instruction's intermediate state. Note that the delta PC quantity
                ;
                ;               delta PC = new PC - old PC
                ;
                ;       is stored in the upper bytes of one of the general registers, usually
                ;       bits <31:24> of R0 or R2. The registers R0 through R3 are stored on
                ;       the stack (in the space used for the first four operands when the
```

```
                     0033   381 ;       reserved instruction is first encountered) so that the same offsets
                     0033   382 ;       that were used to store the delta-PC can be used to retrieve it.
                     0033   383 ;-
                     0033   384
                     0033   385          .ALIGN  LONG                    ; Alignment for exception vector
                     0034   386
                     0034   387 VAX$EMULATE_FPD::
                     0034   388
                     0034   428
00000000'8F    DD    0034   429          PUSHL   #VAX$_OPCDEC_FPD        ; This is the signal name
          04   DD    003A   430          PUSHL   #4                      ; Signal array has four longwords
                     003C   431
               00    003C   433          HALT                            ; Nothing else we can do
```

```
                        .SUBTITLE       Dispatch Tables
                ;+
                ; Functional Description:
                ;
                ;       The case tables for the two CASEB instructions are built with the
                ;       macros that are invoked here. Macros are used to guarantee that both
                ;       tables contain correct entries for a selected opcode at the same
                ;       offset.
                ;
                ; Assumptions:
                ;
                ;       The CASE_TABLE_ENTRY macro assumes that the names of the respective
                ;       case tables are CASE_TABLE_BASE and FPD_CASE_TABLE_BASE.
                ;
                ; Notes:
                ;
                ;       In the following lists, those FPD routines that do not have _FPD in
                ;       their names use the same JSB entry point for initial entry and after
                ;       restarting the instruction. In most of these cases, the register state
                ;       is the same for both starting and restarting. For the remaining cases,
                ;       there is not enough difference between the two cases to justify an
                ;       additional entry point. (See VAX$MOVTC for an example of this latter
                ;       situation.)
                ;
                ;       The FPD routines that include _RESTART in their names have to do a
                ;       certain amount of work to restore the intermediate state from the
                ;       canonical registers before they can resume instruction execution.
                ;-

                        .SAVE                                   ; Remember current location counter

                ; First generate table entries for the string instructions

                        CASE_TABLE_ENTRY        OPCODE=MOVTC,-
                                                ROUTINE=MOVTC,-
                                                FPD_ROUTINE=VAX$MOVTC

                        CASE_TABLE_ENTRY        OPCODE=MOVTUC,-
                                                ROUTINE=MOVTUC,-
                                                FPD_ROUTINE=VAX$MOVTUC

                        CASE_TABLE_ENTRY        OPCODE=CMPC3,-
                                                ROUTINE=CMPC3,-
                                                FPD_ROUTINE=VAX$CMPC3,-
                                                BOOT_FLAG=BOOT

                        CASE_TABLE_ENTRY        OPCODE=CMPC5,-
                                                ROUTINE=CMPC5,-
                                                FPD_ROUTINE=VAX$CMPC5,-
                                                BOOT_FLAG=BOOT

                        CASE_TABLE_ENTRY        OPCODE=LOCC,-
                                                ROUTINE=LOCC,-
                                                FPD_ROUTINE=VAX$LOCC,-
                                                BOOT_FLAG=BOOT

                        CASE_TABLE_ENTRY        OPCODE=SKPC,-
```

```
0028   495                                  ROUTINE=SKPC,-
0028   496                                  FPD_ROUTINE=VAX$SKPC
0028   497
0028   498          CASE_TABLE_ENTRY        OPCODE=SCANC,-
0028   499                                  ROUTINE=SCANC,-
0028   500                                  FPD_ROUTINE=VAX$SCANC
0028   501
0028   502          CASE_TABLE_ENTRY        OPCODE=SPANC,-
0028   503                                  ROUTINE=SPANC,-
0028   504                                  FPD_ROUTINE=VAX$SPANC
0028   505
0028   506          CASE_TABLE_ENTRY        OPCODE=MATCHC,-
0028   507                                  ROUTINE=MATCHC,-
0028   508                                  FPD_ROUTINE=VAX$MATCHC
0028   509
0028   510          CASE_TABLE_ENTRY        OPCODE=CRC,-
0028   511                                  ROUTINE=CRC,-
0028   512                                  FPD_ROUTINE=VAX$CRC
0028   513
0028   514 ; Now generate table entries for the decimal instructions
0028   515
0028   516          CASE_TABLE_ENTRY        OPCODE=ADDP4,-
0028   517                                  ROUTINE=ADDP4,-
0028   518                                  FPD_ROUTINE=VAX$ADDP4
0028   519
0028   520          CASE_TABLE_ENTRY        OPCODE=ADDP6,-
0028   521                                  ROUTINE=ADDP6,-
0028   522                                  FPD_ROUTINE=VAX$ADDP6
0028   523
0028   524          CASE_TABLE_ENTRY        OPCODE=ASHP,-
0028   525                                  ROUTINE=ASHP,-
0028   526                                  FPD_ROUTINE=VAX$ASHP
0028   527
0028   528          CASE_TABLE_ENTRY        OPCODE=CMPP3,-
0028   529                                  ROUTINE=CMPP3,-
0028   530                                  FPD_ROUTINE=VAX$CMPP3
0028   531
0028   532          CASE_TABLE_ENTRY        OPCODE=CMPP4,-
0028   533                                  ROUTINE=CMPP4,-
0028   534                                  FPD_ROUTINE=VAX$CMPP4
0028   535
0028   536          CASE_TABLE_ENTRY        OPCODE=CVTLP,-
0028   537                                  ROUTINE=CVTLP,-
0028   538                                  FPD_ROUTINE=VAX$CVTLP_RESTART
0028   539
0028   540          CASE_TABLE_ENTRY        OPCODE=CVTPL,-
0028   541                                  ROUTINE=CVTPL,-
0028   542                                  FPD_ROUTINE=VAX$CVTPL_RESTART
0028   543
0028   544          CASE_TABLE_ENTRY        OPCODE=CVTPS,-
0028   545                                  ROUTINE=CVTPS,-
0028   546                                  FPD_ROUTINE=VAX$CVTPS
0028   547
0028   548          CASE_TABLE_ENTRY        OPCODE=CVTPT,-
0028   549                                  ROUTINE=CVTPT,-
0028   550                                  FPD_ROUTINE=VAX$CVTPT_RESTART
0028   551
```

```
                0028    552          CASE_TABLE_ENTRY        OPCODE=CVTSP,-
                0028    553                                  ROUTINE=CVTSP,-
                0028    554                                  FPD_ROUTINE=VAX$CVTSP
                0028    555
                0028    556          CASE_TABLE_ENTRY        OPCODE=CVTTP,-
                0028    557                                  ROUTINE=CVTTP,-
                0028    558                                  FPD_ROUTINE=VAX$CVTTP_RESTART
                0028    559
                0028    560          CASE_TABLE_ENTRY        OPCODE=DIVP,-
                0028    561                                  ROUTINE=DIVP,-
                0028    562                                  FPD_ROUTINE=VAX$DIVP
                0028    563
                0028    564          CASE_TABLE_ENTRY        OPCODE=MOVP,-
                0028    565                                  ROUTINE=MOVP,-
                0028    566                                  FPD_ROUTINE=VAX$MOVP
                0028    567
                0028    568          CASE_TABLE_ENTRY        OPCODE=MULP,-
                0028    569                                  ROUTINE=MULP,-
                0028    570                                  FPD_ROUTINE=VAX$MULP
                0028    571
                0028    572          CASE_TABLE_ENTRY        OPCODE=SUBP4,-
                0028    573                                  ROUTINE=SUBP4,-
                0028    574                                  FPD_ROUTINE=VAX$SUBP4
                0028    575
                0028    576          CASE_TABLE_ENTRY        OPCODE=SUBP6,-
                0028    577                                  ROUTINE=SUBP6,-
                0028    578                                  FPD_ROUTINE=VAX$SUBP6
                0028    579
                0028    580 ; EDITPC always seems to find itself in last place
                0028    581
                0028    582          CASE_TABLE_ENTRY        OPCODE=EDITPC,-
                0028    583                                  ROUTINE=EDITPC,-
                0028    584                                  FPD_ROUTINE=VAX$EDITPC_RESTART
                0028    585
           0000003D     586          .RESTORE                        ; Reset current location counter
```

```
003D    588          .SUBTITLE       Description of instruction-specific routines
003D    589
003D    590 ;++
003D    591 ; The instruction-specific routines do similar things. Rather than clutter up
003D    592 ; each routine with the same comments, we will describe the steps that each
003D    593 ; routine takes in this section.
003D    594 ;
003D    595 ; The input parameters to each routine are identical.
003D    596 ;
003D    597 ;                       Contents of exception stack
003D    598 ;                       ---------------------------
003D    599 ;
003D    600 ;           OPCODE(SP)     - Opcode of reserved instruction
003D    601 ;           OLD_PC(SP)     - PC of reserved instruction
003D    602 ;           OPERAND_1(SP) - First operand specifier
003D    603 ;           OPERAND_2(SP) - Second operand specifier
003D    604 ;           OPERAND_3(SP) - Third operand specifier
003D    605 ;           OPERAND_4(SP) - Fourth operand specifier
003D    606 ;           OPERAND_5(SP) - Fifth operand specifier
003D    607 ;           OPERAND_6(SP) - Sixth operand specifier
003D    608 ;           OPERAND_7(SP) - Seventh operand specifier (currently unused)
003D    609 ;           OPERAND_8(SP) - Eight operand specifier (currently unused)
003D    610 ;           NEW_PC(SP)     - PC of instruction following reserved instruction
003D    611 ;           EXCEPTION_PSL(SP) - PSL at time of exception
003D    612 ;
003D    613 ;           The routine headers for the instruction-specific routines in this
003D    614 ;           module will list the input and output parameters in symbolic form
003D    615 ;           only. The VAX$xxxxxx routines in other modules in the emulator contain
003D    616 ;           the exact meanings of the various operands (parameters) to the
003D    617 ;           routines.
003D    618 ;
003D    619 ; Outline of execution:
003D    620 ;
003D    621 ;           The operands are loaded into registers as required by the instruction
003D    622 ;           specific routines. Routine headers for each routine contain detailed
003D    623 ;           descriptions.
003D    624 ;
003D    625 ;           A routine of the form VAX$xxxxxx (where xxxxxx is the instruction
003D    626 ;           name) is called to perform the actual work indicated by each
003D    627 ;           instruction.
003D    628 ;
003D    629 ;           Common exit code executes to allow the condition codes returned by the
003D    630 ;           VAX$xxxxxx routines to be passed back to the code that generated the
003D    631 ;           original exception.
003D    632 ;
003D    633 ; Notes:
003D    634 ;
003D    635 ;           The following routines are constructed to be reasonably fast. In
003D    636 ;           particular, each instruction has its own separate routine, even though
003D    637 ;           several instructions differ only in the instruction-specific routine
003D    638 ;           to which final control is passed. Rather than share this common code
003D    639 ;           at the expense of another dispatch on opcode, we shoose to duplicate
003D    640 ;           the common code.
003D    641 ;--
003D    642
```

```
                     003D   749          .SUBTITLE       CMPC3 - Exception handler for CMPC3 instruction
                     003D   750 ;+
                     003D   751 ; Input Parameters:
                     003D   752 ;
                     003D   753 ;       OPCODE(SP)
                     003D   754 ;       OLD_PC(SP)
                     003D   755 ;       OPERAND_1(SP) - len.rw
                     003D   756 ;       OPERAND_2(SP) - src1addr.ab
                     003D   757 ;       OPERAND_3(SP) - src2addr.ab
                     003D   758 ;       OPERAND_4(SP)
                     003D   759 ;       OPERAND_5(SP)
                     003D   760 ;       OPERAND_6(SP)
                     003D   761 ;       OPERAND_7(SP)
                     003D   762 ;       OPERAND_8(SP)
                     003D   763 ;       NEW_PC(SP)
                     003D   764 ;       EXCEPTION_PSL(SP)
                     003D   765 ;
                     003D   766 ; Output Parameters:
                     003D   767 ;
                     003D   768 ;       R0<15:0> - len.rw
                     003D   769 ;       R1       - src1addr.ab
                     003D   770 ;       R3       - src2addr.ab
                     003D   771 ;
                     003D   772 ; Implicit Output:
                     003D   773 ;
                     003D   774 ;       R0<31:16> - 0
                     003D   775 ;       R2        - UNPREDICTABLE
                     003D   776 ;-
                     003D   777
                     003D   778 CMPC3:
                     003D   779
50   08 AE    3C     003D   780          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
51   0C AE    D0     0041   781          MOVL    OPERAND_2(SP),R1        ; R1       <- src1addr.ab
53   10 AE    D0     0045   782          MOVL    OPERAND_3(SP),R3        ; R3       <- src2addr.ab
                     0049   783
                     0049   784 ; Now that the operands have been loaded, the only exception parameter
                     0049   785 ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                     0049   786 ; there is no reason why the state of the stack needs to be altered and we
                     0049   787 ; save two instructions if we leave the stack alone.
                     0049   788
   0080'CF    9F     0049   789          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
      FFB0'   31     004D   790          BRW     VAX$CMPC3               ; Do the actual work
```

```
                                0050   794          .SUBTITLE       CMPC5 - Exception handler for CMPC5 instruction
                                0050   795 ;+
                                0050   796 ; Input Parameters:
                                0050   797 ;
                                0050   798 ;       OPCODE(SP)
                                0050   799 ;       OLD_PC(SP)
                                0050   800 ;       OPERAND_1(SP) - src1len.rw
                                0050   801 ;       OPERAND_2(SP) - src1addr.ab
                                0050   802 ;       OPERAND_3(SP) - fill.rb
                                0050   803 ;       OPERAND_4(SP) - src2len.rw
                                0050   804 ;       OPERAND_5(SP) - src2addr.ab
                                0050   805 ;       OPERAND_6(SP)
                                0050   806 ;       OPERAND_7(SP)
                                0050   807 ;       OPERAND_8(SP)
                                0050   808 ;       NEW_PC(SP)
                                0050   809 ;       EXCEPTION_PSL(SP)
                                0050   810 ;
                                0050   811 ; Output Parameters:
                                0050   812 ;
                                0050   813 ;       R0<15:0>    - srclen.rw
                                0050   814 ;       R0<23:16>   - fill.rb
                                0050   815 ;       R1          - srcaddr.ab
                                0050   816 ;       R2<15:0>    - src2len.rw
                                0050   817 ;       R3          - src2addr.ab
                                0050   818 ;
                                0050   819 ; Implicit Output:
                                0050   820 ;
                                0050   821 ;       R0<31:24> - UNPREDICTABLE
                                0050   822 ;       R2<31:16> - 0
                                0050   823 ;-
                                0050   824
                                0050   825 CMPC5:
                                0050   826
50   10 AE    10   9C           0050   827          ROTL    #16,OPERAND_3(SP),R0    ; R0<23:16> <- fill.rb
     50   08 AE   B0           0055   828          MOVW    OPERAND_1(SP),R0        ; R0<15:0>  <- src1len.rw
     51   0C AE   D0           0059   829          MOVL    OPERAND_2(SP),R1        ; R1        <- src1addr.ab
     52   14 AE   3C           005D   830          MOVZWL  OPERAND_4(SP),R2        ; R2<15:0>  <- src2len.rw
     53   18 AE   D0           0061   831          MOVL    OPERAND_5(SP),R3        ; R3        <sca- src2addr.ab
                                0065   832
                                0065   833 ; Now that the operands have been loaded, the only exception parameter
                                0065   834 ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                0065   835 ; there is no reason why the state of the stack needs to be altered and we
                                0065   836 ; save two instructions if we leave the stack alone.
                                0065   837
        0080'CF    9F           0065   838          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
           FF94'   31           0069   839          BRW     VAX$CMPC5               ; Do the actual work
```

```
                        006C   937          .SUBTITLE       LOCC - Exception handler for LOCC instruction
                        006C   938 ;+
                        006C   939 ; Input Parameters:
                        006C   940 ;
                        006C   941 ;       OPCODE(SP)
                        006C   942 ;       OLD_PC(SP)
                        006C   943 ;       OPERAND_1(SP) - char.rb
                        006C   944 ;       OPERAND_2(SP) - len.rw
                        006C   945 ;       OPERAND_3(SP) - addr.ab
                        006C   946 ;       OPERAND_4(SP)
                        006C   947 ;       OPERAND_5(SP)
                        006C   948 ;       OPERAND_6(SP)
                        006C   949 ;       OPERAND_7(SP)
                        006C   950 ;       OPERAND_8(SP)
                        006C   951 ;       NEW_PC(SP)
                        006C   952 ;       EXCEPTION_PSL(SP)
                        006C   953 ;
                        006C   954 ; Output Parameters:
                        006C   955 ;
                        006C   956 ;       R0<15:0>  - len.rw
                        006C   957 ;       R0<23:16> - char.rb
                        006C   958 ;       R1        - addr.ab
                        006C   959 ;
                        006C   960 ; Implicit Output:
                        006C   961 ;
                        006C   962 ;       R0<31:24> - UNPREDICTABLE
                        006C   963 ;-
                        006C   964
                        006C   965 LOCC:
                        006C   966
50   08 AE    10   9C   006C   967          ROTL    #16,OPERAND_1(SP),R0    ; R0<23:16> <- char.ab
     50   0C AE    B0   0071   968          MOVW    OPERAND_2(SP),R0        ; R0<15:0>  <- len.rw
     51   10 AE    D0   0075   969          MOVL    OPERAND_3(SP),R1        ; R1        <- addr.ab
                        0079   970
                        0079   971 ; Now that the operands have been loaded, the only exception parameter
                        0079   972 ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0079   973 ; there is no reason why the state of the stack needs to be altered and we
                        0079   974 ; save two instructions if we leave the stack alone.
                        0079   975
          0080'CF    9F   0079   976          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
             FF80'   31   007D   977          BRW     VAX$LOCC                ; Do the actual work
```

```
                     0080  1953          .SUBTITLE       Common Exit Path for VAX$xxxxxx Routines
                     0080  1954 ;+
                     0080  1955 ; Functional Description:
                     0080  1956 ;
                     0080  1957 ;       This is the common exit path for all instruction-specific routines.
                     0080  1958 ;       The condition codes returned by the VAX$xxxxxx routine are stored in
                     0080  1959 ;       the exception PSL and control is passed back to the instruction stream
                     0080  1960 ;       that executed the reserved instruction.
                     0080  1961 ;
                     0080  1962 ; Input Parameters:
                     0080  1963 ;
                     0080  1964 ;       PSL contains condition code settings from VAX$xxxxxx routine.
                     0080  1965 ;
                     0080  1966 ;       OPCODE(SP)     - Opcode of reserved instruction
                     0080  1967 ;       OLD_PC(SP)     - PC of reserved instruction
                     0080  1968 ;       OPERAND_1(SP) - First operand specifier
                     0080  1969 ;       OPERAND_2(SP) - Second operand specifier
                     0080  1970 ;       OPERAND_3(SP) - Third operand specifier
                     0080  1971 ;       OPERAND_4(SP) - Fourth operand specifier
                     0080  1972 ;       OPERAND_5(SP) - Fifth operand specifier
                     0080  1973 ;       OPERAND_6(SP) - Sixth operand specifier
                     0080  1974 ;       OPERAND_7(SP) - Seventh operand specifier (currently unused)
                     0080  1975 ;       OPERAND_8(SP) - Eight operand specifier (currently unused)
                     0080  1976 ;       NEW_PC(SP)     - PC of instruction following reserved instruction
                     0080  1977 ;       EXCEPTION_PSL(SP) - PSL at time of exception
                     0080  1978 ;
                     0080  1979 ; Implicit Input:
                     0080  1980 ;
                     0080  1981 ;       General registers contain architecturally specified values according
                     0080  1982 ;       to specific instruction that was emulated.
                     0080  1983 ;
                     0080  1984 ; Implicit Output:
                     0080  1985 ;
                     0080  1986 ;       Control is passed to the location designated by "new PC" with the
                     0080  1987 ;       condition codes as determined by VAX$xxxxxx. The EXIT routine also
                     0080  1988 ;       preserves general registers.
                     0080  1989 ;-
                     0080  1990
                     0080  1991 VAX$EXIT_EMULATOR::
                7E   DC   0080  1992          MOVPSL  -(SP)                   ; Save the new PSL on the stack
                     0082  1993
                     0082  1994 ; Note that the next instruction makes no assumptions about the condition
                     0082  1995 ; codes in the saved PSL.
                     0082  1996
          04    00   8E   F0   0082  1997          INSV    (SP)+,#0,#4,-
                2C AE                0086  1998                   EXCEPTION_PSL(SP)  ; Replace saved condition codes
                5E   28   C0   0088  1999          ADDL    #NEW_PC,SP              ; Adjust stack pointer (discard old PC)
                          02   008B  2000          REI                             ; Return
                               008C  2001
                               008C  2002          .END
```

```
...OFFSET                = 0000000F
...OPCODE                = 00000038
BOOT_SWITCH              = 00000001
CASE_TABLE_BASE            00000004 R      02
CASE_TABLE_SIZE          = 00000013
CMPC3                      0000003D R      02
CMPC5                      00000050 R      02
EXCEPTION_PSL            = 0000002C
INCLUDE_CMPC3            = 00000000
INCLUDE_CMPC5            = 00000000
INCLUDE_LOCC             = 00000000
LOCC                       0000006C R      02
NEW_PC                   = 00000028
OP$_ACBD                 = 0000006F
OP$_ACBF                 = 0000004F
OP$_ACBG                 = 00004FFD
OP$_ACBH                 = 00006FFD
OP$_ADDD2                = 00000060
OP$_ADDD3                = 00000061
OP$_ADDF2                = 00000040
OP$_ADDF3                = 00000041
OP$_ADDG2                = 000040FD
OP$_ADDG3                = 000041FD
OP$_ADDH2                = 000060FD
OP$_ADDH3                = 000061FD
OP$_ADDP4                = 00000020
OP$_ADDP6                = 00000021
OP$_ASHP                 = 000000F8
OP$_CLRD                 = 0000007C
OP$_CLRF                 = 000000D4
OP$_CLRG                 = 0000007C
OP$_CLRH                 = 00007CFD
OP$_CMPC3                = 00000029
OP$_CMPC5                = 0000002D
OP$_CMPD                 = 00000071
OP$_CMPF                 = 00000051
OP$_CMPG                 = 000051FD
OP$_CMPH                 = 000071FD
OP$_CMPP3                = 00000035
OP$_CMPP4                = 00000037
OP$_CRC                  = 0000000B
OP$_CVTBD                = 0000006C
OP$_CVTBF                = 0000004C
OP$_CVTBG                = 00004CFD
OP$_CVTBH                = 00006CFD
OP$_CVTDB                = 00000068
OP$_CVTDF                = 00000076
OP$_CVTDH                = 000032FD
OP$_CVTDL                = 0000006A
OP$_CVTDW                = 00000069
OP$_CVTFB                = 00000048
OP$_CVTFD                = 00000056
OP$_CVTFG                = 000099FD
OP$_CVTFH                = 000098FD
OP$_CVTFL                = 0000004A
OP$_CVTFW                = 00000049
OP$_CVTGB                = 000048FD
OP$_CVTGF                = 000033FD
OP$_CVTGH                = 000056FD
OP$_CVTGL                = 00004AFD
OP$_CVTGW                = 000049FD
OP$_CVTHB                = 000068FD
OP$_CVTHD                = 0000F7FD
OP$_CVTHF                = 0000F6FD
OP$_CVTHG                = 000076FD
OP$_CVTHL                = 00006AFD
OP$_CVTHW                = 000069FD
OP$_CVTLD                = 0000006E
OP$_CVTLF                = 0000004E
OP$_CVTLG                = 00004EFD
OP$_CVTLH                = 00006EFD
OP$_CVTLP                = 000000F9
OP$_CVTPL                = 00000036
OP$_CVTPS                = 00000008
OP$_CVTPT                = 00000024
OP$_CVTRDL               = 0000006B
OP$_CVTRFL               = 0000004B
OP$_CVTRGL               = 00004BFD
OP$_CVTRHL               = 00006BFD
OP$_CVTSP                = 00000009
OP$_CVTTP                = 00000026
OP$_CVTWD                = 0000006D
OP$_CVTWF                = 0000004D
OP$_CVTWG                = 00004DFD
OP$_CVTWH                = 00006DFD
OP$_DIVD2                = 00000066
OP$_DIVD3                = 00000067
OP$_DIVF2                = 00000046
OP$_DIVF3                = 00000047
OP$_DIVG2                = 000046FD
OP$_DIVG3                = 000047FD
OP$_DIVH2                = 000066FD
OP$_DIVH3                = 000067FD
OP$_DIVP                 = 00000027
OP$_EDITPC               = 00000038
OP$_EMODD                = 00000074
OP$_EMODF                = 00000054
OP$_EMODG                = 000054FD
OP$_EMODH                = 000074FD
OP$_LOCC                 = 0000003A
OP$_MATCHC               = 00000039
OP$_MNEGD                = 00000072
OP$_MNEGF                = 00000052
OP$_MNEGG                = 000052FD
OP$_MNEGH                = 000072FD
OP$_MOVD                 = 00000070
OP$_MOVF                 = 00000050
OP$_MOVG                 = 000050FD
OP$_MOVH                 = 000070FD
OP$_MOVP                 = 00000034
OP$_MOVTC                = 0000002E
OP$_MOVTUC               = 0000002F
OP$_MULD2                = 00000064
OP$_MULD3                = 00000065
```

```
OP$_MULF2                       =  00000044
OP$_MULF3                       =  00000045
OP$_MULG2                       =  000044FD
OP$_MULG3                       =  000045FD
OP$_MULH2                       =  000064FD
OP$_MULH3                       =  000065FD
OP$_MULP                        =  00000025
OP$_POLYD                       =  00000075
OP$_POLYF                       =  00000055
OP$_POLYG                       =  000055FD
OP$_POLYH                       =  000075FD
OP$_SCANC                       =  0000002A
OP$_SKPC                        =  0000003B
OP$_SPANC                       =  0000002B
OP$_SUBD2                       =  00000062
OP$_SUBD3                       =  00000063
OP$_SUBF2                       =  00000042
OP$_SUBF3                       =  00000043
OP$_SUBG2                       =  000042FD
OP$_SUBG3                       =  000043FD
OP$_SUBH2                       =  000062FD
OP$_SUBH3                       =  000063FD
OP$_SUBP4                       =  00000022
OP$_SUBP6                       =  00000023
OP$_TSTD                        =  00000073
OP$_TSTF                        =  00000053
OP$_TSTG                        =  000053FD
OP$_TSTH                        =  000073FD
OPCODE                          =  00000000
OPCODE_BASE                     =  00000029
OPCODE_MAX                      =  0000003B
OPERAND_1                       =  00000008
OPERAND_2                       =  0000000C
OPERAND_3                       =  00000010
OPERAND_4                       =  00000014
OPERAND_5                       =  00000018
VAX$CMPC3                       ********    X   02
VAX$CMPC5                       ********    X   02
VAX$EMULATE                     00000000 RG     02
VAX$EMULATE_FPD                 00000034 RG     02
VAX$EXIT_EMULATOR               00000080 RG     02
VAX$LOCC                        ********    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                       0000008C  (  140.)  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	00:00:01.44
Command processing	71	00:00:00.79	00:00:04.49
Pass 1	409	00:00:13.70	00:00:39.56
Symbol table sort	0	00:00:00.65	00:00:01.11
Pass 2	131	00:00:05.51	00:00:21.29
Symbol table output	19	00:00:00.15	00:00:00.37
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	641	00:00:20.87	00:01:08.29

The working set limit was 1500 pages.
82473 bytes (162 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 453 non-local and 1 local symbols.
4755 source lines were read in Pass 1, producing 13 object records in Pass 2.
141 pages of virtual memory were used to define 139 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	8

518 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

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

XLATE
LIS

VAXEMUL
MAP

WIOHNDLR
LIS

WORKIO
LIS

BOOSTRING
LIS

EMULAT

FPEMUL
MAP

BOOEMULAT
LIS

WORDWRAP
LIS

VAXMACROS
MAR

WRIEDTMSG
LIS

FPEMULATE
LIS

VAXREGDEF
SDL