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

```
_$2

Sym
---
CMP
DEC
DEC
DEC
DEC

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

```
VV      VV    AAAAAA    XX      XX  HH      HH    AAAAAA    NN      NN  DDDDDDDD    LL            RRRRRRRR
VV      VV    AAAAAA    XX      XX  HH      HH    AAAAAA    NN      NN  DDDDDDDD    LL            RRRRRRRR
VV      VV  AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DD      DD  LL            RR      RR
VV      VV  AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DD      DD  LL            RR      RR
VV      VV  AA      AA    XX  XX    HH      HH  AA      AA  NNNN    NN  DD      DD  LL            RR      RR
VV      VV  AA      AA    XX  XX    HH      HH  AA      AA  NNNN    NN  DD      DD  LL            RR      RR
VV      VV  AA      AA      XX      HHHHHHHHHH  AA      AA  NN  NN  NN  DD      DD  LL            RRRRRRRR
VV      VV  AA      AA      XX      HHHHHHHHHH  AA      AA  NN  NN  NN  DD      DD  LL            RRRRRRRR
VV      VV  AAAAAAAAAA    XX  XX    HH      HH  AAAAAAAAAA  NN    NNNN  DD      DD  LL            RR  RR
VV      VV  AAAAAAAAAA    XX  XX    HH      HH  AAAAAAAAAA  NN    NNNN  DD      DD  LL            RR  RR
  VV  VV    AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DD      DD  LL            RR    RR        ....
  VV  VV    AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DD      DD  LL            RR    RR        ....
    VV      AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DDDDDDDD    LLLLLLLLLL  RR      RR      ....
    VV      AA      AA  XX      XX  HH      HH  AA      AA  NN      NN  DDDDDDDD    LLLLLLLLLL  RR      RR      ....

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
```

```
0000          .TITLE  VAX$HANDLER - Condition Handlers for VAX-11 Instruction Emulator
0000          .IDENT  /V04-000/
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility:
0000    ;
0000    ;       VAX-11 Instruction Emulator
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module contains all interfaces between the VAX-11 instruction
0000    ;       emulator and the VAX/VMS condition handling facility. That is, this is
0000    ;       the only module in the entire VAX-11 instruction emulator package that
0000    ;       has any knowledge of VMS-specific aspects of condition handling. All
0000    ;       exception knowledge contained in other modules is defined by the
0000    ;       VAX-11 Architecture (microVAX subset). If this emulator is to be used
0000    ;       in an environment other than VMS, this is the only module that needs
0000    ;       to be changed.
0000    ;
0000    ;       Note that control flows through this module in two directions. On the
0000    ;       one hand, certain exceptions such as access violation can occur while
0000    ;       the emulator is executing. These exceptions are reported by VMS to a
0000    ;       special access violation routine in this module. These routines and
0000    ;       instruction-specific exception handling routines manipulate this
0000    ;       exception so that it appears to have occurred at the site of the
0000    ;       reserved instruction, rather than within the emulator.
0000    ;
0000    ;       Other exceptions such as reserved operand abort for illegal decimal
0000    ;       string length or decimal overflow trap are detected by emulator
0000    ;       routines. These must be reported by the emulator to VMS, again in such
0000    ;       a way that it appears to the running code that the exception occurred
0000    ;       at the site of the reserved instruction.
```

```
0000     58 ;
0000     59 ; Environment:
0000     60 ;
0000     61 ;       These routines run at any access mode, at any IPL, and are AST
0000     62 ;       reentrant.
0000     63 ;
0000     64 ; Author:
0000     65 ;
0000     66 ;       Kathleen D. Morse
0000     67 ;
0000     68 ; Creation Date
0000     69 ;
0000     70 ;       17 August 1982
0000     71 ;
0000     72 ; Modified by:
0000     73 ;
0000     74 ;       V01-005 LJK0036         Lawrence J. Kenah       17-Jul-1984
0000     75 ;               Fix INSV bug in REFLECT_FAULT that was causing PSL bits
0000     76 ;               to be incorrectly cleared.
0000     77 ;
0000     78 ;       V01-004 LJK0022         Lawrence J. Kenah       9-Feb-1984
0000     79 ;               Final cleanup pass. Eliminate stack switch logic. VMS now
0000     80 ;               transfers control here in the mode of the exception and not
0000     81 ;               in kernel mode. Add logic to distinguish a stack that
0000     82 ;               contains a signal array from a stack that also contains a
0000     83 ;               mechanism array and a CHF argument list.
0000     84 ;
0000     85 ;       V01-003 LJK0019         Lawrence J. Kenah       25-Jan-1984
0000     86 ;               Revise the way that exceptions are handled. Eliminate code
0000     87 ;               that is not being used.
0000     88 ;
0000     89 ;       V01-002 LJK0002         Lawrence J. Kenah       15-Mar-1983
0000     90 ;               Handle software detected exceptions from decimal and EDITPC
0000     91 ;               emulation. Modify initial and final dispatching to use registers
0000     92 ;               instead of stack space for parameter passing. Include stack
0000     93 ;               format generated by microVAX exceptions.
0000     94 ;
0000     95 ;       V01-001 Original        Kathleen D. Morse       17-Aug-1982
0000     96 ;               Intercept access violations and machine checks and conditionally
0000     97 ;               dispatch them to emulator for modification. Reflect exceptions
0000     98 ;               from emulator to regular VMS exception dispatcher.
0000     99 ;--
```

```
                0000          .SUBTITLE       Declarations
                0000
                0000    ; Include files:
                0000
                0000          $CHFDEF                         ; Offsets into signal array
                0000          $OPDEF                          ; Symbolic names for opcodes
                0000          $PRDEF                          ; Get definitions of processor registers
                0000          $PSLDEF                         ; Define bit fields in PSL
                0000          $SRMDEF                         ; Arithmetic trap codes
                0000          $SSDEF                          ; Status codes in VMS
                0000
                0000          .NOCROSS                        ; No cross reference for these
                0000          .ENABLE         SUPPRESSION     ; No symbol table entries either
                0000
                0000          STACK_DEF                       ; Stack storage of exception parameters
                0000          PACK_DEF                        ; Stack usage for reflecting exceptions
                0000
                0000          .DISABLE        SUPPRESSION     ; Turn on symbol table again
                0000          .CROSS                          ; Cross reference is OK now
                0000
                0000    ; Macro definitions
                0000
                0000          .MACRO  DELTA_PC_TABLE_ENTRY    OPCODE
                0000          SIGN_EXTEND     OP$_'OPCODE , ...OPCODE
                0000          .IIF    LESS_THAN <...OPCODE - OPCODE_BASE>,-
                0000                  .ERROR                  ; Opcode not supported by emulator
                0000          .IIF    GREATER <...OPCODE - OPCODE_MAX>,-
                0000                  .ERROR                  ; Opcode not supported by emulator
                0000                  .NOCROSS
                0000                  .ENABLE         SUPPRESSION
                0000          OPCODE'_DEF                     ; Register usage for OPCODE instruction
                0000                  .DISABLE        SUPPRESSION
                0000                  .CROSS
                0000          . = DELTA_PC_TABLE_BASE + <...OPCODE-OPCODE_BASE>
                0000          .BYTE   OPCODE'_B_DELTA_PC
                0000          .ENDM   DELTA_PC_TABLE_ENTRY
                0000
                0000          .MACRO  CMP_L_TO_A      SRC,DSTADR,DST,TYPE=B
                0000          .IF     BLANK   DST
                0000                  PUSHA'TYPE      DSTADR
                0000                  CMPL    SRC,(SP)+
                0000          .IF_FALSE
                0000                  MOVA'TYPE       DSTADR,DST
                0000                  CMPL    SRC,DST
                0000          .ENDC
                0000          .ENDM
                0000
                0000    ; Symbol definitions
                0000
      00009F16  0000          JSB_ABSOLUTE = <^X9F@8> ! OP$_JSB
      00000006  0000          JSB_SIZE = 6                    ; Size of JSB @#VAX$xxxxxx instruction
                0000
                0000    ; External declarations
                0000
                0000          .DISABLE        GLOBAL
                0000
                0000          .EXTERNAL       VAX$EXIT_EMULATOR       ; Return PC to dispatcher
```

```
    0000
    0000          .EXTERNAL       EXE$REFLECT     ; Entry point in VMS
    0000
    0000  ;;; THE FOLLOWING DECLARATIONS ARE NOT NEEDED UNLESS WE DEFINE ENTRY POINTS
    0000  ;;; INTO THE EMULATOR THROUGH THE SYSTEM SERVICE VECTOR PAGES.
    0000  ;;;
    0000  ;;;     .EXTERNAL       SYS$VAX_BEGIN,- ; Lower and upper bounds of JSB
    0000  ;;;                     SYS$VAX_END     ;  entry points in vector page
    0000
    0000  ; Default Addressing Mode
    0000
    0000          .DEFAULT        DISPLACEMENT , WORD
    0000
    0000  ; PSECT Declarations:
    0000
    0000  ;       This label defines the beginning of the emulator image
    0000
00000000          .PSECT _VAX$$$BEGIN PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, PAGE
    0000
    0000  VAX$BEGIN:
    0000
    0000  ;       This label locates the end of the emulator image
    0000
00000000          .PSECT _VAX$__END   PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, BYTE
    0000
    0000  VAX$END:
```

```
                             0000          .SUBTITLE       VAX$AL_DELTA_PC_TABLE
                             0000   ;+
                             0000   ; Functional Description:
                             0000   ;
                             0000   ;       The following table contains a byte entry for each emulated
                             0000   ;       instruction. That byte describes where in the saved register set the
                             0000   ;       delta-PC quantity is stored in the event that the instruction is
                             0000   ;       interrupted and restarted.
                             0000   ;-
                             0000
                         00000000          .PSECT _VAX$DATA    PIC, USR, CON, REL, LCL, SHR, NOEXE, RD, NOWRT, LONG
                             0000
                             0000   ; Because the assembler does not understand sign extension of byte and
                             0000   ; word quantities, we must accomplish this sign extension with macros. The
                             0000   ; assignment statements that appear as comments illustrate the sense of the
                             0000   ; macro invocations that immediately follow.
                             0000
                             0000   ;       OPCODE_BASE = OP$_ASHP          ; Smallest (in signed sense) opcode
                             0000
                             0000           SIGN_EXTEND     OP$_ASHP , OPCODE_BASE
                             0000
                             0000   ;       OPCODE_MAX = OP$_SKPC           ; Largest opcode in this emulator
                             0000
                             0000           SIGN_EXTEND     OP$_SKPC , OPCODE_MAX
                             0000
                 00000044    0000   DELTA_PC_TABLE_SIZE = -
                             0000           <OPCODE_MAX - OPCODE_BASE> + 1  ; Define table size
                             0000
                             0000   DELTA_PC_TABLE_BASE:                    ; Locate start of table for macro
                             0000
                             0000   ; Initialize entire table to empty
                             0000
00'00'00'00'00'00'00'00'00'00'00'00' 0000          .BYTE   0[DELTA_PC_TABLE_SIZE]  ; Lots of bytes that contain zero
00'00'00'00'00'00'00'00'00'00'00'00' 000C
00'00'00'00'00'00'00'00'00'00'00'00' 0018
00'00'00'00'00'00'00'00'00'00'00'00' 0024
00'00'00'00'00'00'00'00'00'00'00'00' 0030
             00'00'00'00'00'00'00'00' 003C
                             0044
                             0044   ; Define table entries for string instructions
                             0044
                             0044           DELTA_PC_TABLE_ENTRY    MOVTC
                             0037           DELTA_PC_TABLE_ENTRY    MOVTUC
                             0038           DELTA_PC_TABLE_ENTRY    CMPC3
                             0032           DELTA_PC_TABLE_ENTRY    CMPC5
                             0036           DELTA_PC_TABLE_ENTRY    LOCC
                             0043           DELTA_PC_TABLE_ENTRY    SKPC
                             0044           DELTA_PC_TABLE_ENTRY    SCANC
                             0033           DELTA_PC_TABLE_ENTRY    SPANC
                             0034           DELTA_PC_TABLE_ENTRY    MATCHC
                             0042           DELTA_PC_TABLE_ENTRY    CRC
                             0014
                             0014   ; Define table entries for decimal instructions
                             0014
                             0014           DELTA_PC_TABLE_ENTRY    ADDP4
                             0029           DELTA_PC_TABLE_ENTRY    ADDP6
                             002A           DELTA_PC_TABLE_ENTRY    ASHP
```

```
                  0001            DELTA_PC_TABLE_ENTRY    CMPP3
                  003E            DELTA_PC_TABLE_ENTRY    CMPP4
                  0040            DELTA_PC_TABLE_ENTRY    CVTLP
                  0002            DELTA_PC_TABLE_ENTRY    CVTPL
                  003F            DELTA_PC_TABLE_ENTRY    CVTPS
                  0011            DELTA_PC_TABLE_ENTRY    CVTPT
                  002D            DELTA_PC_TABLE_ENTRY    CVTSP
                  0012            DELTA_PC_TABLE_ENTRY    CVTTP
                  002F            DELTA_PC_TABLE_ENTRY    DIVP
                  0030            DELTA_PC_TABLE_ENTRY    MOVP
                  003D            DELTA_PC_TABLE_ENTRY    MULP
                  002E            DELTA_PC_TABLE_ENTRY    SUBP4
                  002B            DELTA_PC_TABLE_ENTRY    SUBP6
                  002C
                  002C     ; Don't forget good old EDITPC
                  002C
                  002C            DELTA_PC_TABLE_ENTRY    EDITPC
                  0041
                  0041     ; Locate the table through entry 0
                  0041
        00000008  0041     VAX$AL_DELTA_PC TABLE == DELTA_PC_TABLE_BASE + <0 - OPCODE_BASE>
                  0041
                  0041     ; Finally, set the location counter to the end of the table
                  0041
        00000044  0041     . = DELTA_PC_TABLE_BASE + DELTA_PC_TABLE_SIZE
                  0044
                  0044     ;      The code that does useful work is put into this section.
                  0044
              00000000            .PSECT _VAX$CODE    PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, QUAD
```

```
0000          .SUBTITLE       VAX$ACVIOLAT - Modify Access Violation
0000     ;+
0000     ; Functional Description:
0000     ;
0000     ;       This routine receives control from the exception dispatcher in VMS
0000     ;       after the mechanism and signal arrays have been moved from the kernel
0000     ;       stack to the stack of the mode in which the access violation occurred.
0000     ;       This routine determines whether the exception occurred inside the
0000     ;       emulator and, if so, passes control to an instruction-specific routine
0000     ;       that performs further consistenct checks.
0000     ;
0000     ;       The purpose of all this work is to allow exceptions that occur inside
0000     ;       the emulator to be modified before being passed on to the user. There
0000     ;       are two important pieces of this modification.
0000     ;
0000     ;               The exception PC is changed from a PC within the emulator to
0000     ;               the PC of the instruction that caused the emulator to be
0000     ;               invoked in the first place.
0000     ;
0000     ;               Any stack usage by the emulator is dissolved. This allows
0000     ;               the exception stack to be placed directly on top of the
0000     ;               user's stack usage, removing any overt indication that an
0000     ;               emulator is being used.
0000     ;
0000     ;       Although the only exception that can be modified in this fashion is an
0000     ;       access violation, the routine is written in terms of a general signal
0000     ;       array to allow expansion at a future time.
0000     ;
0000     ; Input Parameters:
0000     ;
0000     ;       If this exception is one that the emulator is capable of modifying,
0000     ;       then R10 must contain the address of an instruction-specific routine
0000     ;       to store context information in the general registers before passing
0000     ;       control back to this module.
0000     ;
0000     ;       Although VMS places the mechanism array directly underneath the
0000     ;       argument list and, with one intervening longword, places the signal
0000     ;       array directly underneath that, these lists display general signal and
0000     ;       mechanism arrays.
0000     ;
0000     ;               00(SP) - Return PC in VMS exception dispatcher
0000     ;               04(SP) - Argument count (always 2)
0000     ;               08(SP) - Address of signal array
0000     ;               12(SP) - Address of mechanism array
0000     ;
0000     ;       The mechanism array looks like this
0000     ;
0000     ;               00(mech) - Argument count (always 4)
0000     ;               04(mech) - Value of FP when exception occurred
0000     ;               08(mech) - Depth value (initially -3)
0000     ;               12(mech) - Value of R0 when exception occurred
0000     ;               16(mech) - Value of R1 when exception occurred
0000     ;
0000     ;       In the general caseof an exception with M optional parameters, the
0000     ;       signal array looks like this
0000     ;
0000     ;               00(signal) - Argument count (M+3)
```

```
0000    ;               04(signal) - Exception name
0000    ;               08(signal) - First optional parameter if M GTRU 0
0000    ;                 .
0000    ;                 .
0000    ;               4*M +  4(signal) - Last parameter if M GTRU 0
0000    ;               4*M +  8(signal) - PC of faulting instruction
0000    ;               4*M + 12(signal) - PSL at time of exception
0000    ;
0000    ;       For the case of an access violation, which has two optional
0000    ;       parameters, the signal array takes this form.
0000    ;
0000    ;               00(signal) - Argument count (always 5)
0000    ;               04(signal) - SS$_ACCVIO
0000    ;               08(signal) - Access violation reason mask
0000    ;               12(signal) - Inaccessible virtual address
0000    ;               16(signal) - PC of faulting instruction
0000    ;               20(signal) - PSL at time of exception
0000    ;
0000    ;       The longword immediately following the exception PSL was on top of the
0000    ;       stack when the exception occurred.
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       If the exception passes the small number of tests performed here,
0000    ;       control is passed to the routine whose address is stored in R10
0000    ;       with the following output parameters.
0000    ;
0000    ;               R0 - Address of top of stack (value of SP) when exception occurred
0000    ;               R1 - Exception PC
0000    ;
0000    ;       R0 to R3 are saved on the stack to allow this routine and the
0000    ;       instruction-specific routines to do useful work without juggling
0000    ;       registers.
0000    ;
0000    ;               00(SP) - Value of R0 when exception occurred
0000    ;               04(SP) - Value of R1 when exception occurred
0000    ;               08(SP) - Value of R2 when exception occurred
0000    ;               12(SP) - Value of R3 when exception occurred
0000    ;               16(SP) - Return PC in VMS exception dispatcher
0000    ;               20(SP) - Argument count (always 2)
0000    ;               24(SP) - Address of signal array
0000    ;               28(SP) - Address of mechanism array
0000    ;
0000    ;       If the tests performed here determine that the exception does not
0000    ;       fit the pattern for a candidate to be modified, control is passed
0000    ;       back to VMS with an RSB instruction.
0000    ;
0000    ; Notes:
0000    ;
0000    ;       Some of the code here assumes that it knows what a VMS exception stack
0000    ;       looks like. If it finds the stack in a different shape, it does not
0000    ;       continue. The reason is that, if VMS changes these insignificant pieces
0000    ;       of the stack, it may also change things that this routine relies on that
0000    ;       are more difficult to detect.
0000    ;-
0000
0000    VAX$MODIFY_EXCEPTION::
```

```
         7E    52   7D  0000            MOVQ    R2,-(SP)                ; Start with two scratch registers
   53    0C    AE   DE  0003            MOVAL   12(SP),R3               ; R3 locates argument list
         63    02   D1  0007            CMPL    #2,(R3)                 ; Is argument count 2?
               41   12  000A            BNEQ    20$                     ; Quit if not
   52    04    A3   D0  000C            MOVL    CHF$L_SIGARGLST(R3),R2  ; R2 locates the signal array
   53    08    A3   D0  0010            MOVL    CHF$L_MCHARGLST(R3),R3  ; R3 locates the mechanism array
         63    04   D1  0014            CMPL    #4,CHF$L_MCH_ARGS(R3)   ; Is this argument count 4?
               34   12  0017            BNEQ    20$                     ; Quit if not
   7E    0C    A3   7D  0019            MOVQ    CHF$L_MCH_SAVR0(R3),-(SP)       ; Save original R0 and R1
         50    62   D0  001D            MOVL    CHF$L_SIG_ARGS(R2),R0   ; R0 contains signal argument count
   51    FC A240    D0  0020            MOVL    <8-12>(R2)[R0],R1       ; Get exception PC
                        0025
                        0025  ; Check that exception PC and R10 are both within the bounds of the emulator.
                        0025
   53    0000'CF    9E  0025            MOVAB   VAX$BEGIN,R3            ; Put base address in convenient place
         53    51   D1  002A            CMPL    R1,R3                   ; Is exception PC within this limit?
               1B   1F  002D            BLSSU   10$                     ; Quit if PC is at smaller address
         53    5A   D1  002F            CMPL    R10,R3                  ; Is R10 within this limit?
               16   1F  0032            BLSSU   10$                     ; Quit if R10 is at smaller address
                        0034
                        0034  ; Do the same checks with the ending address
                        0034
   53    0000'CF    9E  0034            MOVAB   VAX$END,R3              ; Put end address in convenient place
         53    51   D1  0039            CMPL    R1,R3                   ; Is exception PC within this limit?
               0C   1E  003C            BGEQU   10$                     ; Quit if PC is too large
         53    5A   D1  003E            CMPL    R10,R3                  ; Is R10 within this limit?
               07   1E  0041            BGEQU   10$                     ; Quit if R10 is too large
                        0043
                        0043  ; Load R0 with the value of SP when the exception occurred
                        0043
   50    04 A240    DE  0043            MOVAL   <16-12>(R2)[R0],R0      ; Get top of stack at time of exception
               6A   17  0048            JMP     (R10)                   ; Call instruction-specific routine
                        004A
         50    8E   7D  004A  10$:      MOVQ    (SP)+,R0                ; Restore R0 and R1
         52    8E   7D  004D  20$:      MOVQ    (SP)+,R2                ; ... and R2 and R3
                    05  0050            RSB
```

```
0051           .SUBTITLE       Reserved Operand or Addressing Mode Exception
0051    ;+
0051    ; Functional Description:
0051    ;
0051    ;       This routine receives control from the emulator routines for the
0051    ;       decimal instructions or EDITPC when those routines have detected a
0051    ;       condition that requires signalling of an exception. The exception can
0051    ;       either be a reserved addressing mode fault from CVTPL or a reserved
0051    ;       operand exception, either a fault or an abort, signalled from a
0051    ;       variety of places. This  routine simply makes the stack look like the
0051    ;       stack on entry to VAX$REFLECT_FAULT and passes control to that routine
0051    ;       to perform the instruction backup in common code.
0051    ;
0051    ; Input Parameters:
0051    ;
0051    ;       00(SP) - Offset in packed register array to delta PC byte
0051    ;       04(SP) - Return PC from VAX$xxxxxx routine
0051    ;
0051    ; Output Parameters:
0051    ;
0051    ;       R0 - Locates return PC in middle of stack
0051    ;       R1 - Contains delta PC originally stored on top of stack
0051    ;
0051    ;       00(SP) - Saved R0
0051    ;       04(SP) - Saved R1
0051    ;       08(SP) - Saved R2
0051    ;       12(SP) - Saved R3
0051    ;       16(SP) - Size of signal array (always 3)
0051    ;       20(SP) - Exception name (SS$_ROPRAND or SS$_RADRMOD)
0051    ;       24(SP) - Place holder for PC of exception
0051    ;       28(SP) - PSL of exception
0051    ;       32(SP) - Offset to delta PC byte (no longer needed)
0051    ; R0 -> 36(SP) - Return PC from VAX$xxxxxx routine
0051    ;
0051    ; Implicit Output:
0051    ;
0051    ;       This routine exits by dropping into the VAX$REFLECT_FAULT routine
0051    ;       that decides the particular form the instruction backup will take.
0051    ;
0051    ; Notes:
0051    ;
0051    ;       There are three ways that a reserved operand exception can occur.
0051    ;
0051    ;       1. Digit count of packed decimal string GTRU 31
0051    ;
0051    ;               This is an abort where the PC points to the offending
0051    ;               decimal or EDITPC instruction.
0051    ;
0051    ;       2. Illegal numeric or sign digit detected by CVTSP or CVTTP
0051    ;
0051    ;               An illegal numeric digit was detected by one of these
0051    ;               instructions or an illegal sign character was detected by
0051    ;               CVTSP. This exception is also not restartable.
0051    ;
0051    ;       3. Illegal EDITPC pattern operator
0051    ;
0051    ;               The EDITPC decoder detected an illegal pattern operator. This
```

```
                         0051   ;               exception stores the intermediate state in registers and may
                         0051   ;               be restarted.
                         0051   ;
                         0051   ;       A reserved addressing mode exception can only occur when the PC is
                         0051   ;       used as the destination operand in the CVTPL instruction.
                         0051   ;-
                         0051
                         0051           .ENABLE         LOCAL_BLOCK
                         0051
                         0051   VAX$RADRMOD::
              7E    DC   0051           MOVPSL  -(SP)                   ; Store exception PSL
              7E    D4   0053           CLRL    -(SP)                   ; Save space for the exception PC
     0000044C 8F    DD   0055           PUSHL   #SS$_RADRMOD            ; Store exception name
              0A    11   005B           BRB     10$                     ; Join common code
                         005D
                         005D   VAX$ROPRAND::
              7E    DC   005D           MOVPSL  -(SP)                   ; Store exception PSL
              7E    D4   005F           CLRL    -(SP)                   ; Save space for the exception PC
     00000454 8F    DD   0061           PUSHL   #SS$_ROPRAND            ; Store exception name
                         0067
              03    DD   0067   10$:    PUSHL   #3                      ; Store signal array size
              0F    BB   0069           PUSHR   #^M<R0,R1,R2,R3>        ; Store the usual registers
  51     20 AE      D0   006B           MOVL    32(SP),R1               ; R1 gets delta PC offset
  50     24 AE      DE   006F           MOVAL   36(SP),R0               ; R0 locates the return PC
                         0073                                           ; Drop through to VAX$REFLECT_FAULT
                         0073
                         0073           .DISABLE        LOCAL_BLOCK
```

```
0073            .SUBTITLE       VAX$REFLECT_FAULT - Reflect Fault to User
0073    ;+
0073    ; Functional Description:
0073    ;
0073    ;       This routine reflects a fault (such as an access violation that
0073    ;       occurred inside the emulator) back to the user. The signal array, in
0073    ;       particular, the exception PC, is modified to point to the reserved
0073    ;       instruction or the JSB instruction into the emulator.
0073    ;
0073    ; Input Parameters:
0073    ;
0073    ;       R0 - Address on stack of return PC
0073    ;       R1<7:0> - Byte offset from top of stack into saved register array
0073    ;                 (R0..R3) where delta-PC will be stored if original path into
0073    ;                 emulator was through a reserved instruction exception
0073    ;       R1<8>   - distinguisnes restartable exceptions (faults) from
0073    ;                 exceptions that cannot be restarted (aborts)
0073    ;       R1<9>   - distinguishes software generated exceptions from exceptions
0073    ;                 detected by hardware, detoured through the emulator, and
0073    ;                 modified by instruction-specific routines.
0073    ;
0073    ;       Note that the condition codes in the exception PSL are significant for
0073    ;       faults, primarily to make the EDITPC illegal pattern operator
0073    ;       exception conform to the architecture.
0073    ;
0073    ;       00(SP) - Saved R0
0073    ;       04(SP) - Saved R1
0073    ;       08(SP) - Saved R2
0073    ;       12(SP) - Saved R3
0073    ;
0073    ;       16(SP) - Number of additional longwords in signal array (called N)
0073    ;       20(SP) - Exception name
0073    ;
0073    ;       IF N GTRU 3 THEN
0073    ;         24(SP)                - First exception-specific parameter
0073    ;            .
0073    ;            .
0073    ;         <4*<N-2> + 16>(SP) - Last exception specific parameter
0073    ;
0073    ;       <4*<N-1> + 16>(SP)      - PC of exception
0073    ;       <4*N + 16>(SP)          - PSL of exception
0073    ;
0073    ;       <4*N + 16 + 4>(SP)      - Instruction specific storage (no longer needed)
0073    ;            .
0073    ;            .
0073    ;       -04(R0)                 - Last longword of instruction specific storage
0073    ;
0073    ;       (R0) - Return PC from VAX$xxxxxx routine in emulator
0073    ;
0073    ; There are three possibilities for the return PC. The action of this
0073    ; routine depends on this return PC value.
0073    ;
0073    ; Case 1.       (R0) - VAX$EXIT_EMULATOR
0073    ;
0073    ;       This is the usual case where the emulator was entered as a result of
0073    ;       an emulated instruction exception. The signal array from the second
0073    ;       exception is put on top of the original exception array, the rest of
```

```
                   0073  ;       the stack is evaporated, and the exception is reflected to the user.
                   0073  ;
                   0073  ;       The FPD bit in the exception PSL is set for those exceptions that are
                   0073  ;       restartable.
                   0073  ;
                   0073  ; Case 2.       (R0) - Address of instruction following
                   0073  ;
                   0073  ;                      JSB     @#VAX$xxxxxx
                   0073  ;
                   0073  ;       In this case, the signal array that is passed back to the user is
                   0073  ;       reflected has the address of the JSB instruction as the exception PC.
                   0073  ;       The FPD bit is ALWAYS clear. Note that it is much more difficult (if
                   0073  ;       not impossible) to back up an arbitrary instruction that transfers
                   0073  ;       control to the em  ator.
                   0073  ;
                   0073  ; Case 3.       (R0) - Anything else
                   0073  ;
                   0073  ;       In this case, the emulator was entered in some other way. Because
                   0073  ;       instruction state has already been modified, it is no longer possible to
                   0073  ;       simply reflect the secondary exception (as we did with unrecognized
                   0073  ;       exceptions in routine VAX$MODIFY_EXCEPTION). We add an additional longword
                   0073  ;       to the signal array (VAX$_ABORT) and report this slightly modified
                   0073  ;       exception to the original caller without modifying the return PC.
                   0073  ;-
                   0073
                   0073  VAX$REFLECT_FAULT::
    52   80   D0   0073          MOVL    (R0)+,R2                ; Get return PC from stack
                   0076          CMP_L_TO_A      R2,VAX$EXIT_EMULATOR,R3
         53   12   007E          BNEQU   NO_SIGNAL_ARRAY         ; Branch if no secondary signal array
                   0080
                   0080  ; If we drop through the branch, we are examining Case 1. We can use the signal
                   0080  ; array that already exists to hold the modified exception parameters.
```

```
                              0080     ; Case 1.        (R0) - VAX$EXIT_EMULATOR
                              0080     ;
                              0080     ;       More Input Parameters:
                              0080     ;
                              0080     ;               04(R0) - Opcode of reserved instruction
                              0080     ;               08(R0) - PC of reserved instruction (old PC)
                              0080     ;               12(R0) - First operand specifier (no longer needed)
                              0080     ;                  .
                              0080     ;                  .
                              0080     ;               40(R0) - Eight operand specifier (place holder)
                              0080     ;               44(R0) - PC of instruction following reserved instruction
                              0080     ;                        (new PC)
                              0080     ;               48(R0) - PSL at time of exception
                              0080     ;
                              0080     ;       Output Parameters for Case 1:
                              0080     ;
                              0080     ;               R0 through R3 restored from top of stack
                              0080     ;
                              0080     ;               00(SP) - Size of signal array (called N)
                              0080     ;               04(SP) - Exception name
                              0080     ;
                              0080     ;               IF N GTRU 3 THEN
                              0080     ;                 08(SP)        - First exception-specific parameter
                              0080     ;                    .
                              0080     ;                    .
                              0080     ;                 <4*<N-2>>(SP) - Last exception specific parameter
                              0080     ;
                              0080     ;               <4*<N-1>>(SP)   - Old PC (PC of reserved instruction)
                              0080     ;               <4*N>(SP)       - PSL of second exception (FPD set)
                              0080     ;
                              0080
                              0080             .ENABLE         LOCAL_BLOCK
                              0080
                              0080     ; We need to capture the FPD information stored in R1 before that register
                              0080     ; is modified (or used as an index register).
                              0080
        05 51    08     E5    0080             BBCC    #PACK_V_FPD,R1,10$      ; Branch if FPD bit remains clear
                 1B     E2    0084             BBSS    #PSL$V_FPD,-
          00 2C  A0           0086                     EXCEPTION_PSL(R0),10$   ; Set FPD bit in exception PSL
   52    28 A0   04 A0  C3    0089     10$:    SUBL3   OLD_PC(R0),NEW_PC(R0),R2        ; Calculate delta PC
           53    51     9A    008F             MOVZBL  R1,R3                   ; Isolate delta-PC offset in R3
         6E43    52     90    0092             MOVB    R2,(SP)[R3]             ; Store delta PC in one of R0..R3
        0B 51    09     E0    0096             BBS     #PACK_V_ACCVIO,R1,20$   ; Branch if more than signal array
                              009A
                              009A     ; In this case, the signal array is located immediately underneath the
                              009A     ; saved register array.
                              009A
        53    10 AE     9A    009A             MOVZBL  <PACK_L_SIGNAL_ARRAY+CHF$L_SIG_ARGS>(SP),R3
                              009E                                             ; Get signal array size
      52    10 AE43     DE    009E             MOVAL   PACK_L_SIGNAL_ARRAY(SP)[R3],R2
                              00A3                                             ; R2 points to exception PSL
                 0B     11    00A3             BRB     30$                     ; Rejoin common code
                              00A5
                              00A5     ; In this case, there is other information on the stack between the saved
                              00A5     ; register array and the signal array, namely a return address in VMS, an
                              00A5     ; argument list that would have been passed to condition hanlders had the
                              00A5     ; exception not been detoured through this code, and a mechanism array.
```

```
                      00A5   ; All of this extra data is discarded and the exception as the modified
                      00A5   ; stack is passed back to the VMS exception dispatcher.
                      00A5
      52   18 AE  D0  00A5   20$:    MOVL    PACK_L_SIGNAL_ARRAY_POINTER(SP),R2
                      00A9                                           ; R2 locates signal array
         53   62  D0  00A9           MOVL    CHF$L_SIG_ARGS(R2),R3   ; Get signal array size
      52    6243  DE  00AC           MOVAL   CHF$L_SIG_ARGS(R2)[R3],R2
                      00B0                                           ; R2 points to exception PSL
      51   2C AC  DE  00B0   30$:    MOVAL   EXCEPTION_PSL(R0),R1    ; R1 points to original PSL
                      00B4
                      00B4   ;+
                      00B4   ; R0 - Address of VAX$_OPCDEC exception array
                      00B4   ; R1 - Address of PSL of original VAX$_OPCDEC exception
                      00B4   ; R2 - Address of PSL in signal array of exception being backed up
                      00B4   ; R3 - Number of longwords in signal array
                      00B4   ;-
                      00B4
   FC A2   04 A0  D0  00B4           MOVL    OLD_PC(R0),-4(R2)       ; This extra MOVL actually saves code!
                      00B9
                      00B9   ; There are two more operations that need to be performed on the PSL that
                      00B9   ; will appear in the signal array. These operations are only significant for
                      00B9   ; faults. The affected PSL fields are defined to be UNPREDICTABLE in the case
                      00B9   ; of aborts. Rather than complicate the code with unnecessary branches,
                      00B9   ; however, the condition codes will be propogated and the TP bit will be
                      00B9   ; cleared in all cases.
                      00B9
61   04   00   62  F0  00B9          INSV    (R2),#0,#4,(R1)         ; Copy condition codes to PSL
      00 61   1E  E4  00BE           BBSC    #PSL$V_TP,(R1),40$      ; Clear the TP bit
                      00C2
                      00C2   ; We now move a modified signal array down the stack, from the back (PSL) end
                      00C2   ; to the front (argument count) end. The PSL has already been moved before
                      00C2   ; the loop executes to allow the FPD bit to get set and to make the loop
                      00C2   ; count work correctly.
                      00C2
         71   72  D0  00C2   40$:    MOVL    -(R2),-(R1)             ; Move next longword
           FA 53  F5  00C5           SOBGTR  R3,40$                  ; Check for any more
                      00C8
                      00C8   ; At the end of this loop, R1 points to what we want to report as the top of the
                      00C8   ; stack to the VMS exception dispatcher. We store R1 on the stack following the
                      00C8   ; saved R0 through R3 and set the new stack pointer as part of the POPR
                      00C8   ; instruction.
                      00C8
      10 AE   51  D0  00C8           MOVL    R1,PACK_L_SAVED_SP(SP)  ; Load new SP underneath R0..R3 array
         400F 8F  BA  00CC           POPR    #^M<R0,R1,R2,R3,SP>     ; Restore registers and set SP
            0079  31  00D0           BRW     VAX$REFLECT_TO_VMS      ; Use common exit path to VMS
```

```
                        00D3  ; Case 2.       (R0) - Address of instruction following
                        00D3  ;
                        00D3  ;                         JSB     @#VAX$xxxxxx
                        00D3  ;
                        00D3  ;       More Input Parameters:
                        00D3  ;
                        00D3  ;               There is nothing else of interest on the stack in this case.
                        00D3  ;
                        00D3  ;       Output Parameters for Case 2:
                        00D3  ;
                        00D3  ;               R0 through R3 restored from top of stack
                        00D3  ;
                        00D3  ;               00(SP) - Size of signal array (called N)
                        00D3  ;               04(SP) - Exception name
                        00D3  ;
                        00D3  ;               IF N GTRU 3 THEN
                        00D3  ;                 08(SP)        - First exception-specific parameter
                        00D3  ;                   .
                        00D3  ;                   .
                        00D3  ;                 <4*<N-2>>(SP) - Last exception specific parameter
                        00D3  ;
                        00D3  ;               <4*<N-1>>(SP)   - Address of JSB instruction
                        00D3  ;               <4*N>(SP)       - PSL of second exception (FPD clear!)
                        00D3  ;
                        00D3
                        00D3  ;+
                        00D3  ; This is either Case 2 or Case 3, depending on the instruction located by
                        00D3  ; the return PC. If this is a return PC from a JSB @# into the emulator,
                        00D3  ; then the instruction that we wish to examine lies six bytes before the
                        00D3  ; location pointed to by R2.
                        00D3  ;-
                        00D3
                        00D3  NO_SIGNAL_ARRAY:
9F16 8F   FA A2   B1    00D3          CMPW    -6(R2),#JSB_ABSOLUTE    ; Is the opcode JSB @# ?
          16      12    00D9          BNEQ    UNKNOWN                 ; Branch if not
                        00DB
                        00DB  ;;; IF WE EVER INSTALL ENTRY POINTS INTO THE EMULATOR THROUGH THE SYSTEM
                        00DB  ;;; SERVICE VECTOR PAGE, THE FOLLOWING CHECKS CAN BE TURNED ON. THE INTENT
                        00DB  ;;; IS THAT, LIKE SYSTEM SERVICE CALLS, THE EMULATOR REFERENCES WOULD
                        00DB  ;;; GENERATE JSB @# VAX$xxxxxx INSTRUCTIONS TO JMP INSTRUCTIONS IN THE
                        00DB  ;;; VECTOR PAGES. IT IS ONLY THIS SET OF JSB INSTRUCTIONS THAT WOULD BE
                        00DB  ;;; BACKED UP ACCORDING TO METHOD 2. ALL OTHER PATHS INTO THE EMULATOR
                        00DB  ;;; WILL GENERATE EXCEPTIONS WITH AN EXCEPTION PC INSIDE THE EMULATOR ITSELF.
                        00DB
                        00DB  ;;;     MOVL    -4(R2),R2               ; Get destination of JSB @#
                        00DB  ;;;     CMP_L_TO_A      R2,SYS$VAX_BEGIN,R3     ; Make lower bounds check
                        00DB  ;;;     BLSSU   UNKNOWN                 ; Branch if too small
                        00DB  ;;;     CMP_L_TO_A      R2,SYS$VAX_END,R3       ; Make upper bounds check
                        00DB  ;;;     BGEQU   UNKNOWN                 ; Branch if too large
                        00DB
                        00DB  ; This is Case 2 as described above. It differs from Case 1 in two ways. There
                        00DB  ; is no signal array on the stack underneath the stack that is overwritten.
                        00DB  ; The FPD bit in the saved PSL is not set.
                        00DB
     53   10 AE   9A    00DB          MOVZBL  <PACK_L_SIGNAL_ARRAY+CHF$L_SIG_ARGS>(SP),R3
                        00DF                                          ; Get signal array size
          53      D6    00DF          INCL    R3                      ; Make loop count correct
```

```
            52   1D AE43  DE  00E1   753        MOVAL   PACK_L_SIGNAL_ARRAY(SP)[R3],R2  ; R2 points beyond exception PSL
   F8 A2    FC A0    06  C3  00E6   754        SUBL3   #JSB_SIZE,-4(R0),-8(R2) ; PC of JSB becomes exception PC
               51    50  D0  00EC   755        MOVL    R0,R1                   ; Start writing signal array at return PC
                     D1  11  00EF   756        BRB     40$                     ; Join common exit at top of loop
```

```
                        00F1  ; Case 3.       (R0) - Anything else
                        00F1  ;
                        00F1  ;                 This is a case where the emulator was entered in a nonstandard
                        00F1  ;                 way. This code has no way of creating an exception PC that
                        00F1  ;                 will cause the emulator to be reentered. Instead, the
                        00F1  ;                 exception is redefined to be VAX$_ABORT with the rest of the
                        00F1  ;                 original signal array comprising the exception parameters. The
                        00F1  ;                 PC is not modified so that knowledgable code in the form of a
                        00F1  ;                 condition handler could conceivably restart such an exception
                        00F1  ;                 if it knew how to reenter the emulator.
                        00F1  ;
                        00F1  ;         More Input Parameters:
                        00F1  ;
                        00F1  ;                 There is nothing else of interest on the stack in this case.
                        00F1  ;
                        00F1  ;         Output Parameters for Case 3:
                        00F1  ;
                        00F1  ;                 R0 through R3 restored from top of stack
                        00F1  ;
                        00F1  ;                 00(SP) - Size of new signal array (N + 1)
                        00F1  ;                 04(SP) - New exception name (VAX$_ABORT)
                        00F1  ;                 08(SP) - Original exception name
                        00F1  ;
                        00F1  ;                 IF N GTRU 3 THEN
                        00F1  ;                   12(SP)            - First exception-specific parameter
                        00F1  ;                      .
                        00F1  ;                      .
                        00F1  ;                   <4*<N-2>+4>(SP) - Last exception specific parameter
                        00F1  ;
                        00F1  ;                 <4*<N-1>+4>(SP)   - "Anything else" (original return PC)
                        00F1  ;                 <4*N+4>(SP)       - PSL of second exception (FPD clear!)
                        00F1  ;-
                        00F1
                        00F1  ;+
                        00F1  ; This is Case 3. The emulator was entered in some unorthodox fashion. We leave
                        00F1  ; the exception PC alone but change the facility field in the exception name
                        00F1  ; located in the signal array to VAX$_.
                        00F1  ;-
                        00F1
                        00F1  UNKNOWN:
      53   10 AE    9A  00F1          MOVZBL  <PACK_L_SIGNAL_ARRAY+CHF$L_SIG_ARGS>(SP),R3
                        00F5                                                  ; Get signal array size
              53    D6  00F5          INCL    R3                              ; Make loop count correct
   52   10 AE43    DE  00F7          MOVAL   PACK_L_SIGNAL_ARRAY(SP)[R3],R2  ; R2 points beyond exception PSL
 F8 A2    FC A0    D0  00FC          MOVL    -4(R0),-8(R2)                   ; Unmodified return PC is exception PC
      51    50    D0  0101          MOVL    R0,R1                           ; Start writing signal array at return PC
                        0104
              BC    11  0104          BRB     40$                             ; Join common exit at top of loop
                        0106
                        0106          .DISABLE        LOCAL_BLOCK
```

```
0106          .SUBTITLE       VAX$REFLECT_TRAP - Reflect Arithmetic Traps
0106  ;+
0106  ; Functional Description:
0106  ;
0106  ;       This routine receives control from the various instruction specific
0106  ;       emulator routines in order to reflect arithmetic traps back to the
0106  ;       caller. There are three arithmetic traps that can occur.
0106  ;
0106  ;       Decimal overflow
0106  ;
0106  ;               This can occur in most of the decimal instructions and EDITPC
0106  ;               when there is not enough room in the destination string to
0106  ;               store all of the nonzero digits in the source.
0106  ;
0106  ;       Integer overflow
0106  ;
0106  ;               CVTPL can incur this exception when the input decimal string
0106  ;               converts into a longword that cannot fit into 31 bits.
0106  ;
0106  ;       Divide by zero
0106  ;
0106  ;               DIVP generates this exception when the divisor is zero.
0106  ;
0106  ; Input Parameters:
0106  ;
0106  ;       00(SP) - Arithmetic Trap Code (from $SRMDEF)
0106  ;       04(SP) - PSL on exit from VAX$xxxxxx routine
0106  ;       08(SP) - Return PC from VAX$xxxxxx routine
0106  ;
0106  ;       The return PC will determine whether an exception frame (signal array)
0106  ;       already exists or must be built. Briefly, if 08(SP) is equal to the
0106  ;       address called VAX$EXIT_EMULATOR, then the emulator was entered as
0106  ;       a result of execution of a reserved instruction and a signal array
0106  ;       already exists. If 08(SP) is anything else, then it is treated as the
0106  ;       address of an instruction following a JSB into the emulator.
0106  ;
0106  ; Implicit Input:
0106  ;
0106  ;       If 08(SP) is VAX$EXIT_EMULATOR, then the rest of the stack that is
0106  ;       relevant looks like this.
0106  ;
0106  ;       12(SP) - Opcode of reserved instruction
0106  ;       16(SP) - PC of reserved instruction
0106  ;       20(SP)
0106  ;          .
0106  ;          .
0106  ;       48(SP)
0106  ;       52(SP) - PC of next instruction
0106  ;       56(SP) - PSL of original exception
0106  ;
0106  ; Output Parameters:
0106  ;
0106  ;       00(SP) - Count of longwords in signal array (always equal to 3)
0106  ;       04(SP) - Signal name (modified form of trap code)
0106  ;       08(SP) - PC of next instruction
0106  ;       12(SP) - PSL on exit from VAX$xxxxxx routine
0106  ;
```

```
                          0106  ; Implicit Output:
                          0106  ;
                          0106  ;       This routine passes control to EXE$REFLECT, which will eventually
                          0106  ;       reflect the arithmetic trap back to the user.
                          0106  ;-
                          0106
                          0106  ; Insure that architectural status codes for arithmetic traps are consistent
                          0106  ; with the VMS status codes that they are mapped onto.
                          0106
                          0106          ASSUME SS$_INTOVF EQ <SS$_ARTRES + <8 * SRM$K_INT_OVF_T>>
                          0106          ASSUME SS$_FLTDIV EQ <SS$_ARTRES + <8 * SRM$K_FLT_DIV_T>>
                          0106          ASSUME SS$_DECOVF EQ <SS$_ARTRES + <8 * SRM$K_DEC_OVF_T>>
                          0106
                          0106  VAX$REFLECT_TRAP::
           7E    50   7D  0106          MOVQ    R0,-(SP)                ; Get some scratch registers
 51    08 AE    08   C5   0109          MULL3   #8,8(SP),R1             ; Turn trap code into status
 51   00000474 8F    C0   010E          ADDL2   #SS$_ARTRES,R1          ; ... by adding in base status code
    50    0000'CF    9E   0115          MOVAB   VAX$EXIT_EMULATOR,R0    ; This allows address comparisons
       10 AE    50   D1   011A          CMPL    R0,16(SP)               ; Compare with return PC
                18   13   011E          BEQL    10$                     ; Some signal array already exists
                          0120
                          0120  ;+
                          0120  ; This code path is taken if the emulator was entered in any way other than
                          0120  ; by executing one of the reserved instructions. We assume that the longword
                          0120  ; on the stack represents a return PC, the address of the next instruction
                          0120  ; that will execute.
                          0120  ;
                          0120  ;       R0 - Scratch
                          0120  ;       R1 - Modified Trap Code
                          0120  ;
                          0120  ;       00(SP) - Saved R0
                          0120  ;       04(SP) - Saved R1
                          0120  ;       08(SP) - Space for trap code
                          0120  ;       12(SP) - PSL on exit from VAX$xxxxxx routine
                          0120  ;       16(SP) - Return PC
                          0120  ;-
                          0120
      08 AE    51    D0   0120          MOVL    R1,8(SP)                ; Store exception code
      50    0C AE    D0   0124          MOVL    12(SP),R0               ; Save PSL in R0 to start switch
   0C AE    10 AE    D0   0128          MOVL    16(SP),12(SP)           ; Put PC into proper place
      10 AE    50    D0   012D          MOVL    R0,16(SP)               ; Store new exception PSL
         50    8E    7D   0131          MOVQ    (SP)+,R0                ; Restore saved R0 and R1
               03    DD   0134          PUSHL   #3                      ; Store size of signal array
               14    11   0136          BRB     VAX$REFLECT_TO_V[illegible]     ; Join common exit to VMS
                          0138
                          0138  ;+
                          0138  ; This code path is taken if the emulator was entered as a result of executing
                          0138  ; one of the reserved instructions. There is lots of extra space on the stack
                          0138  ; to fool around with so the juggling act exhibited by the previous block of
                          0138  ; code is not necessary here.
                          0138  ;
                          0138  ;       R0 - Scratch
                          0138  ;       R1 - Modified Trap Code
                          0138  ;
                          0138  ;       00(SP) - Saved R0
                          0138  ;       04(SP) - Saved R1
                          0138  ;       08(SP) - Initial trap code (no longer needed)
```

```
                                0138   923 ;        12(SP) - PSL on exit from VAX$xxxxxx routine
                                0138   924 ;        16(SP) - VAX$EXIT_EMULATOR (no longer needed)
                                0138   925 ;        20(SP)
                                0138   926 ;          .
                                0138   927 ;          .
                                0138   928 ;        56(SP)
                                0138   929 ;        60(SP) - PC of next instruction (preserved by this code)
                                0138   930 ;        64(SP) - PSL of original exception (with modified condition codes)
                                0138   931 ;
                                0138   932 ; The condition codes that were generated by the VAX$xxxxxx routine are
                                0138   933 ; stored in the exception PSL.
                                0138   934 ;-
                                0138   935
   40 AE   04   00   0C AE   F0 0138   936 10$:     INSV    12(SP),#0,#4,<EXCEPTION_PSL+20>(SP)
                                013F   937                                            ; Store new condition codes
             38 AE   51      D0 013F   938          MOVL    R1,<OPERAND_8+20>(SP)      ; ... and modified trap code
                  50   8E    7D 0143   939          MOVQ    (SP)+,R0                   ; Restore saved registers
             5E   30 AE      9E 0146   940          MOVAB   <OPERAND_8+12>(SP),SP      ; Eliminate unneeded stack space
                       03    DD 014A   941          PUSHL   #3                         ; Store signal array size and ...
                                014C   942                                            ;  drop through to VAX$REFLECT_TO_VMS
```

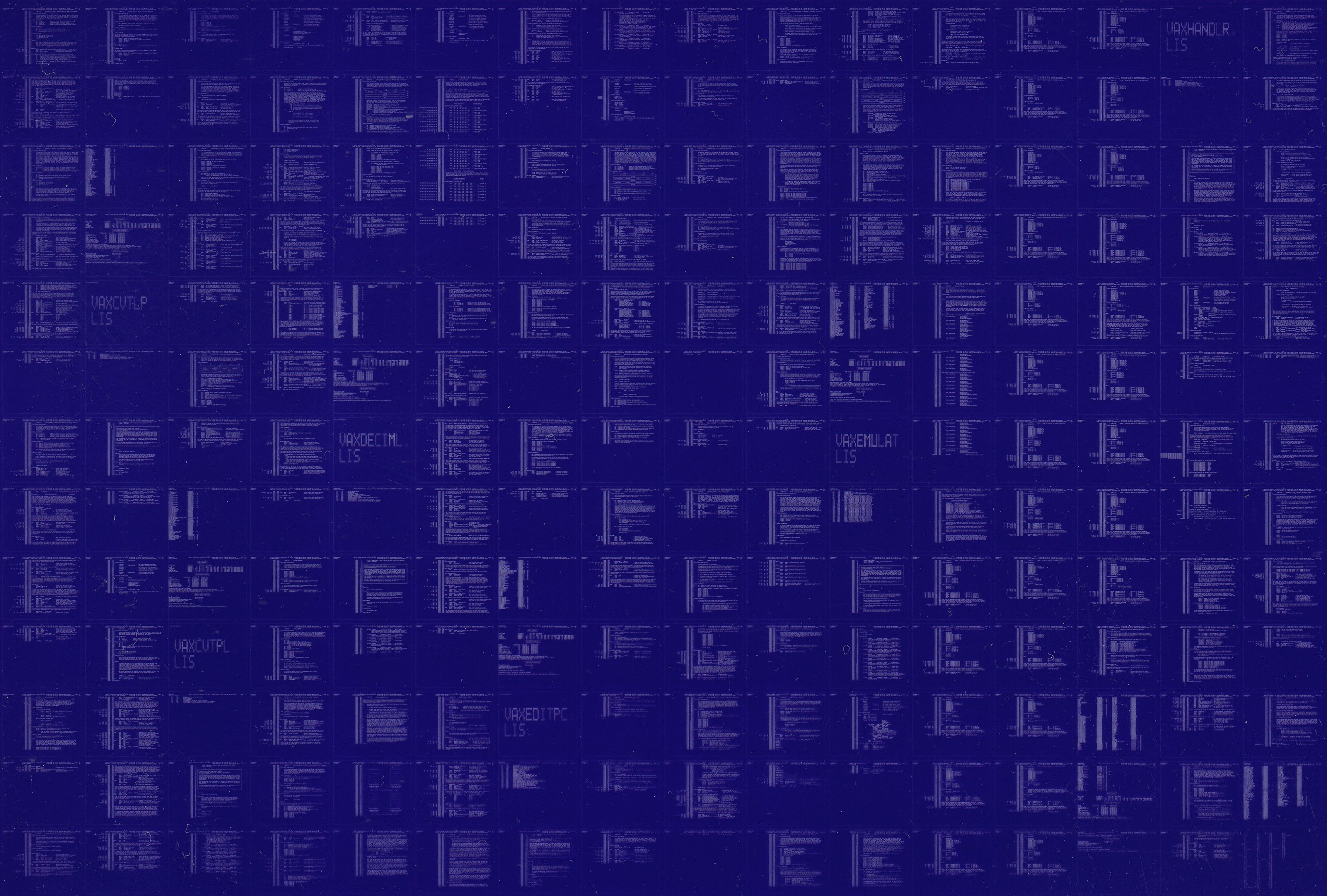
```
                        014C         .SUBTITLE      VAX$REFLECT_TO_VMS - Let VMS Reflect the Exception
                        014C   ;+
                        014C   ; Functional Description:
                        014C   ;
                        014C   ;       This routine is the common exit path to the VMS exception dispatcher.
                        014C   ;       It executes in three different sets of circumstances.
                        014C   ;
                        014C   ;    1. Software detected exceptions
                        014C   ;
                        014C   ;       There are several exceptions that are detected by software.
                        014C   ;       Exception-specific and context-specific routines build a signal array
                        014C   ;       om the stack. This code then passes that signal array to VMS.
                        014C   ;
                        014C   ;    2. Modified hardware exceptions
                        014C   ;
                        014C   ;       Certain forms of access violation are modified to appear as if they
                        014C   ;       occurred at the site of a reserved instruction rather than within the
                        014C   ;       emulator. Any extraneous stack storage is removed. The PC within the
                        014C   ;       access violation signal array is modified. This code then passes the
                        014C   ;       modified signal array to VMS.
                        014C   ;
                        014C   ;    3. All unmodified exceptions
                        014C   ;
                        014C   ;       Certain exceptions cause the emulator to receive control, even though
                        014C   ;       the exception in question will be passed intact to VMS. These include
                        014C   ;       exceptions caused by a random control transfer into the emulator and
                        014C   ;       access violations such as stack overflow that would not have occurred
                        014C   ;       in the first place if the reserved instructions were being executed by
                        014C   ;       the base machine rather than by a software emulator.
                        014C   ;
                        014C   ; Input Parameters:
                        014C   ;
                        014C   ;       The signal array is on the stack of the access mode in which the
                        014C   ;       exception occurred.
                        014C   ;
                        014C   ;               00(SP) - Number of signal array elements (called N)
                        014C   ;               04(SP) - Signal name (integer value)
                        014C   ;               08(SP) - First exception-specific parameter (if any)
                        014C   ;               12(SP) - Second exception-specific parameter (if any)
                        014C   ;                  .
                        014C   ;                  .
                        014C   ;               <8+4*N>(SP)  - Exception PC that will be reported
                        014C   ;               <12+4*N>(SP) - Exception PSL that will be reported
                        014C   ;
                        014C   ; Implicit Output:
                        014C   ;
                        014C   ;       Control is passed to the VMS exception dispatcher at label EXE$REFLECT.
                        014C   ;-
                        014C
                        014C  VAX$REFLECT_TO_VMS::
00000000'GF      17     014C         JMP     G^EXE$REFLECT          ; Let VMS handle this exception
                        0152
                        0152         .END
```

```
...OPCODE                      = 00000038                 OP$_MOVTC                       = 0000002E
ADDP4_B_DELTA_PC               = 00000003                 OP$_MOVTUC                      = 0000002F
ADDP6_B_DELTA_PC               = 00000003                 OP$_MULP                        = 00000025
ASHP_B_DELTA_PC                = 00000003                 OP$_SCANC                       = 0000002A
CHF$L_MCHARGLST                = 00000008                 OP$_SKPC                        = 0000003B
CHF$L_MCH_ARGS                 = 00000000                 OP$_SPANC                       = 0000002B
CHF$L_MCH_SAVR0                = 0000000C                 OP$_SUBP4                       = 00000022
CHF$L_SIGARGLST                = 00000004                 OP$_SUBP6                       = 00000023
CHF$L_SIG_ARGS                 = 00000000                 OPCODE_BASE                     = FFFFFFF8
CMPC3_B_DELTA_PC               = 00000003                 OPCODE_MAX                      = 0000003B
CMPC5_B_DELTA_PC               = 00000003                 OPERAND_8                       = 00000024
CMPP3_B_DELTA_PC               = 00000003                 PACK_L_SAVED_SP                 = 00000010
CMPP4_B_DELTA_PC               = 00000003                 PACK_L_SIGNAL_ARRAY             = 00000010
CRC_B_DELTA_PC                 = 0000000B                 PACK_L_SIGNAL_ARRAY_POINTER     = 00000018
CVTLP_B_DELTA_PC               = 0000000B                 PACK_V_ACCVIO                   = 00000009
CVTPL_B_DELTA_PC               = 00000003                 PACK_V_FPD                      = 00000008
CVTPS_B_DELTA_PC               = 00000003                 PSL$V_FPD                       = 0000001B
CVTPT_B_DELTA_PC               = 00000003                 PSL$V_TP                        = 0000001E
CVTSP_B_DELTA_PC               = 00000003                 SCANC_B_DELTA_PC                = 00000003
CVTTP_B_DELTA_PC               = 00000003                 SKPC_B_DELTA_PC                 = 00000003
DELTA_PC_TABLE_BASE              00000000 R     04        SPANC_B_DELTA_PC                = 00000003
DELTA_PC_TABLE_SIZE            = 00000044                 SRM$K_DEC_OVF_T                 = 00000006
DIVP_B_DELTA_PC                = 00000003                 SRM$K_FLT_DIV_T                 = 00000004
EDITPC_B_DELTA_PC              = 0000000A                 SRM$K_INT_OVF_T                 = 00000001
EXCEPTION_PSL                  = 0000002C                 SS$_ARTRES                      = 00000474
EXE$REFLECT                      ********   X   00        SS$_DECOVF                      = 000004A4
JSB_ABSOLUTE                   = 00009F16                 SS$_FLTDIV                      = 00000494
JSB_SIZE                       = 00000006                 SS$_INTOVF                      = 0000047C
LOCC_B_DELTA_PC                = 00000003                 SS$_RADRMOD                     = 0000044C
MATCHC_B_DELTA_PC              = 00000003                 SS$_ROPRAND                     = 00000454
MOVP_B_DELTA_PC                = 00000003                 SUBP4_B_DELTA_PC                = 00000003
MOVTC_B_DELTA_PC               = 0000000B                 SUBP6_B_DELTA_PC                = 00000003
MOVTUC_B_DELTA_PC              = 0000000B                 UNKNOWN                           000000F1 R     05
MULP_B_DELTA_PC                = 00000003                 VAX$AL_DELTA_PC_TABLE           = 00000008 RG    04
NEW_PC                         = 00000028                 VAX$BEGIN                         00000000 R     02
NO_SIGNAL_ARRAY                  000000D3 R     05        VAX$END                           00000000 R     03
OLD_PC                         = 00000004                 VAX$EXIT_EMULATOR                 ********   X   00
OP$_ADDP4                      = 00000020                 VAX$MODIFY_EXCEPTION              00000000 RG    05
OP$_ADDP6                      = 00000021                 VAX$RADRMOD                       00000051 RG    05
OP$_ASHP                       = 000000F8                 VAX$REFLECT_FAULT                 00000073 RG    05
OP$_CMPC3                      = 00000029                 VAX$REFLECT_TO_VMS                0000014C RG    05
OP$_CMPC5                      = 0000002D                 VAX$REFLECT_TRAP                  00000106 RG    05
OP$_CMPP3                      = 00000035                 VAX$ROPRAND                       0000005D RG    05
OP$_CMPP4                      = 00000037
OP$_CRC                        = 0000000B
OP$_CVTLP                      = 000000F9
OP$_CVTPL                      = 00000036
OP$_CVTPS                      = 00000008
OP$_CVTPT                      = 00000024
OP$_CVTSP                      = 00000009
OP$_CVTTP                      = 00000026
OP$_DIVP                       = 00000027
OP$_EDITPC                     = 00000038
OP$_JSB                        = 00000016
OP$_LOCC                       = 0000003A
OP$_MATCHC                     = 00000039
OP$_MOVP                       = 00000034
```

```
                                          +----------------+
                                          ! 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$$$BEGIN               00000000  (     0.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC PAGE
_VAX$  END                 00000000  (     0.)  03 (  3.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC BYTE
_VAX$DATA                  00000044  (    68.)  04 (  4.)    PIC   USR   CON   REL   LCL   SHR NOEXE   RD   NOWRT NOVEC LONG
_VAX$CODE                  00000152  (   338.)  05 (  5.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC QUAD

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    17    00:00:00.07    00:00:00.46
Command processing                76    00:00:00.52    00:00:05.45
Pass 1                           326    00:00:12.74    00:00:35.67
Symbol table sort                  0    00:00:01.86    00:00:03.81
Pass 2                           178    00:00:03.09    00:00:08.09
Symbol table output               12    00:00:00.10    00:00:00.16
Psect synopsis output              2    00:00:00.03    00:00:00.03
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             611    00:00:18.41    00:00:53.69

The working set limit was 1350 pages.
70076 bytes (137 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1287 non-local and 8 local symbols.
996 source lines were read in Pass 1, producing 19 object records in Pass 2.
47 pages of virtual memory were used to define 45 macros.

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

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1               30
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     10
TOTALS (all libraries)                                40

1432 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

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

0144 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



VAXHANDLR
LIS

VAXCVTLP
LIS

VAXDECIML
LIS

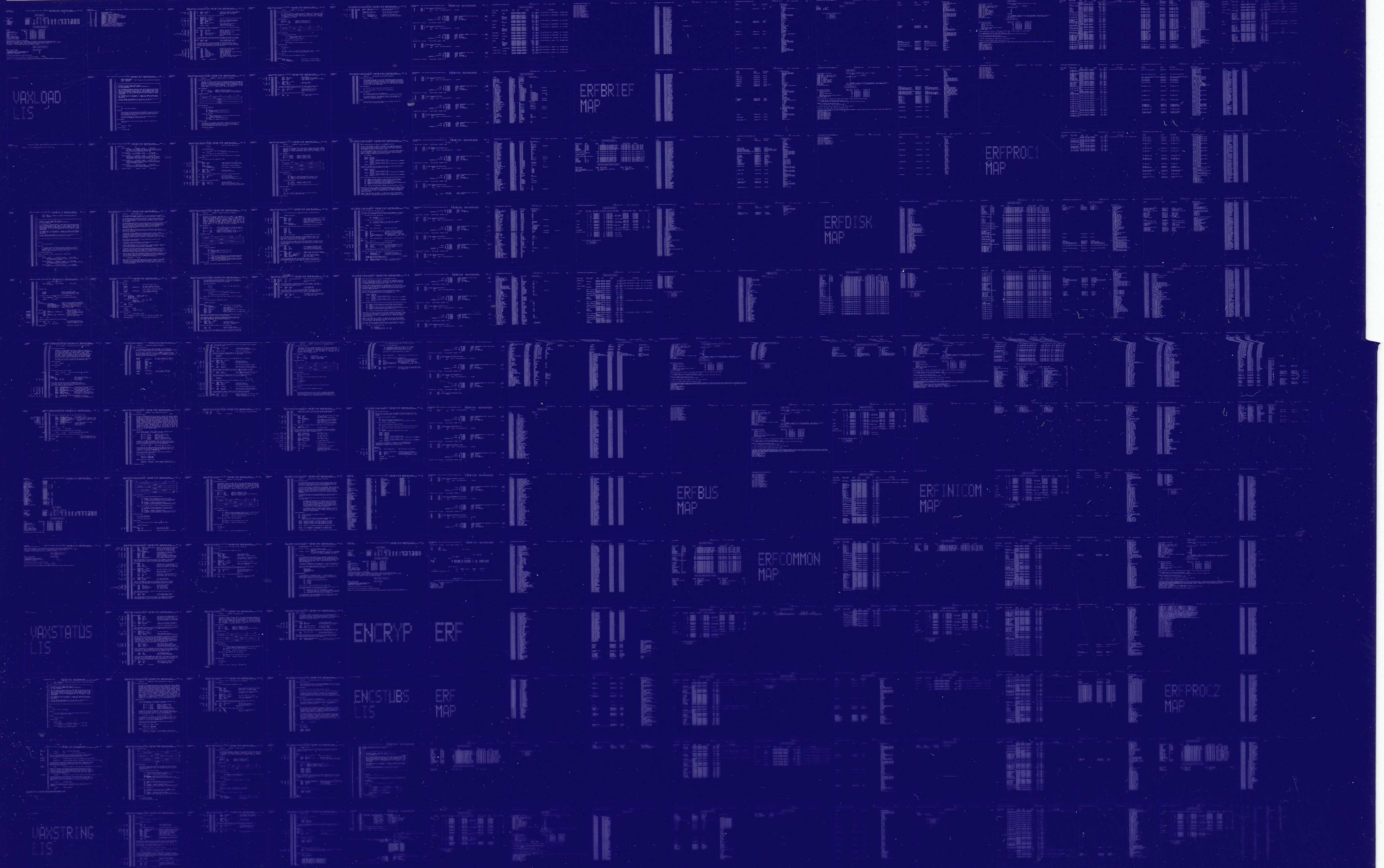
VAXEMULAT
LIS

VAXCVTPL
LIS

VAXEDITPC
LIS

0145 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


VAXLOAD
LIS

ERFBRIEF
MAP

ERFPROC1
MAP

ERFDISK
MAP

ERFBUS
MAP

ERFINICOM
MAP

ERFCOMMON
MAP

VAXSTATUS
LIS

ENCRYP

ERF

ENCSTUBS
LIS

ERF
MAP

ERFPROC2
MAP

VAXSTRING
LIS