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

```
_$2

Sym
---
CMP
DEC
DEC
DEC
DEC

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

```
VV       VV    AAAAAA    XX       XX  MM       MM    AAAAAA      CCCCCCCC  RRRRRRRR      000000      SSSSSSSS
VV       VV    AAAAAA    XX       XX  MM       MM    AAAAAA      CCCCCCCC  RRRRRRRR      000000      SSSSSSSS
VV       VV   AA    AA   XX       XX  MMMM   MMMM   AA    AA    CC        RR      RR   OO    OO    SS
VV       VV   AA    AA   XX       XX  MMMM   MMMM   AA    AA    CC        RR      RR   OO    OO    SS
VV       VV   AA    AA     XX   XX    MM  MM  MM   AA    AA    CC        RR      RR   OO    OO    SS
VV       VV   AA    AA     XX   XX    MM  MM  MM   AA    AA    CC        RR      RR   OO    OO    SS
VV       VV   AA    AA       XX      MM      MM   AA    AA    CC        RRRRRRRR     OO    OO     SSSSSS
VV       VV   AA    AA       XX      MM      MM   AA    AA    CC        RRRRRRRR     OO    OO     SSSSSS
VV       VV   AAAAAAAAAA     XX   XX MM      MM   AAAAAAAAAA  CC        RR   RR      OO    OO          SS
VV       VV   AAAAAAAAAA     XX   XX MM      MM   AAAAAAAAAA  CC        RR   RR      OO    OO          SS
  VV   VV     AA    AA   XX       XX  MM      MM   AA    AA    CC        RR     RR    OO    OO          SS      ....
  VV   VV     AA    AA   XX       XX  MM      MM   AA    AA    CC        RR     RR    OO    OO          SS      ....
    VV       AA    AA   XX       XX  MM      MM   AA    AA      CCCCCCCC  RR      RR     000000    SSSSSSSS      ....
    VV       AA    AA   XX       XX  MM      MM   AA    AA      CCCCCCCC  RR      RR     000000    SSSSSSSS      ....

MM       MM    AAAAAA    RRRRRRRR
MM       MM    AAAAAA    RRRRRRRR
MMMM   MMMM   AA    AA   RR      RR
MMMM   MMMM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AAAAAAAAAA RR  RR
MM      MM   AAAAAAAAAA RR  RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR      RR
MM      MM   AA    AA   RR      RR
```

```
        .NLIST
;       Version 'V04-000'
```



```
;++
; Facility:
;
;       VAX-11 Instruction Emulator (Macros for Condition Handling)
;
; Abstract:
;
;       This file contains a set of macros that are used in each source
;       module of the emulator as an aid in condition handling. The
;       MARK_POINT macro is used to indicate all instructions that can
;       cause exceptions (such as access violations) that must be modified
;       in some way before they are passed on to the caller. The other macros
;       exist to support the MARK_POINT macro.
;
; Author:
;
;       Lawrence J. Kenah
;
; Creation Date:
;
;       16 December 1982
;
; Modification History:
;
;       V01-005 LJK0026         Lawrence J. Kenah       19-Mar-1984
;               Final cleanup pass. Add MODULE_END label generation to
;               END_MARK_POINT macro.
;
```

```
;       V01-004 LJK0020         Lawrence J. Kenah       26-Oct-1983
;               Add alternate restart capability to MARK_POINT macro. Add
;               RESTART_POINT macro.
;
;       V01-003 LJK0008         LAWRENCE J. KENAH       19-Oct-1983
;               Move ROPRAND_CHECK macro here from source code to support
;               decimal instruction emulation routines in several modules.
;
;               Steal CASE macro from VMS so that emulator does not need
;               VMS-specific libraries in its assembly phase.
;
;       V01-002 Sign extension  Lawrence J. Kenah       16-Mar-1983
;               Add macro that performs sign-extended assignment.
;
;       V01-001 Original        Lawrence J. Kenah       16-Dec-1982
;               Add macros that are used in locating instructions that can
;               incur exceptions that are capable of being backed up.
;--
```

```
;+
;       BEGIN_MARK_POINT - Set up names for MARK_POINT macro
;
;       This macro is invoked before the first call to MARK_POINT to define
;       certain symbol that are used by the MARK_POINT macro. The four symbol
;       names defined by this macro and later referenced by the code are
;
;           MODULE_BASE         Address from which other offsets are computed
;           PC_TABLE_BASE       Base address of exception PC table
;           HANDLER_TABLE_BASE  Base address of handler table
;           TABLE_SIZE          Number of entries in each table
;
;       Two other symbols needed by VAX$EDITPC can be defined if the optional
;       flag parameter is set to RESTART. These symbols are called
;
;           RESTART_PC_TABLE_BASE       Base address of restart PC table
;           RESTART_TABLE_SIZE          Number of entries in restart table
;-

        .MACRO  BEGIN_MARK_POINT        FLAG

        .IF     NOT_DEFINED     BOOT_SWITCH

MODULE_BASE = .                 ; Define base address for module

TABLE_SIZE = 0                  ; Start with an empty table

        .SAVE_PSECT     LOCAL_BLOCK
        .PSECT  PC_TABLE        CON,NOEXE,LCL,PIC,SHR,RD,NOWRT
PC_TABLE_BASE:
        .PSECT  HANDLER_TABLE   CON,NOEXE,LCL,PIC,SHR,RD,NOWRT
HANDLER_TABLE_BASE:
        .RESTORE_PSECT

        .IF     IDENTICAL       <FLAG>,RESTART
        RESTART_TABLE_SIZE = 0
        .SAVE_PSECT     LOCAL_BLOCK
        .PSECT  RESTART_PC_TABLE        CON,NOEXE,LCL,PIC,SHR,RD,NOWRT
RESTART_PC_TABLE_BASE:
        .RESTORE_PSECT

        .ENDC

        .ENDC

        .ENDM   BEGIN_MARK_POINT
```

```
;+
;       MARK_POINT - Indicate Potential Exception Site
;
;       This macro is invoked before writing an instruction that can cause
;       an exception that must be backed up before being passed on to the
;       user. Its single argument is the address of the handler code that
;       will properly back up this exception.
;-

        .MACRO  MARK_POINT      HANDLER , FLAG

        .IF     NOT_DEFINED     BOOT_SWITCH

        ...PC... = . - MODULE_BASE      ; Remember relative PC

        .IIF    NOT_DEFINED     TABLE_SIZE      ,-
                .ERROR  ; The BEGIN_MARK_POINT macro must be called first
        TABLE_SIZE = TABLE_SIZE + 1     ; Count another table entry

        .SAVE_PSECT     LOCAL_BLOCK
        .PSECT  PC_TABLE
        .WORD   ...PC...
        .PSECT  HANDLER_TABLE
        .WORD   HANDLER - MODULE_BASE   ; Store relative handler offset
        .RESTORE_PSECT

        .IF     IDENTICAL       <FLAG>,RESTART
        RESTART_POINT           HANDLER
        .ENDC

        .ENDC

        .ENDM   MARK_POINT
```

```
;+
;       RESTART_POINT - Indicate Alternate Instruction Restart Site
;
;       This macro is invoked as part of the MARK_POINT macro or by the
;       EO_READ macro used by VAX$EDITPC to indicate an alternate site to
;       restart the instruction. If the optional argument is present, a symbol
;       of the form code'_RESTART is defined to be equal to the current value
;       of the restart PC table index.
;-

        .MACRO  RESTART_POINT   CODE

        .IF     NOT_DEFINED     BOOT_SWITCH

        ...RESTART_PC... = . - MODULE_BASE      ; Remember relative PC

        .IIF    NOT_DEFINED     RESTART_TABLE_SIZE      ,-
                .ERROR  ; The BEGIN_MARK_POINT macro must be called first
        RESTART_TABLE_SIZE = RESTART_TABLE_SIZE + 1
                                                ; Count another table entry

        .IIF    NOT_BLANK       <CODE>          ,-
                CODE'_RESTART = RESTART_TABLE_SIZE

        .SAVE_PSECT     LOCAL_BLOCK
        .PSECT  RESTART_PC_TABLE
        .WORD   ...RESTART_PC...
        .RESTORE_PSECT

        .ENDC

        .ENDM   RESTART_POINT
```

```
;+
;       END_MARK_POINT - Perform mark point consistency checks
;
;       This macro is invoked at the end of the modules that use the mark
;       point tables for exception modification. Its only purpose is to insure
;       that all PC offsets can fit into 16 bits. (Note that it also tests the
;       maximum size of the restart table used by EDITPC to insure that the
;       table index can fit into the STATE field.)
;-

        .MACRO  END_MARK_POINT  STATE_VECTOR_SIZE

        .IF     NOT_DEFINED     BOOT_SWITCH

        MODULE_END = .                  ; Define label for end of module

        .IIF    GREATER         <<. - MODULE_BASE> - 65535> ,-
                .ERROR  ; Module is too large for PC offsets stored in a word

        .IF     DEFINED         RESTART_TABLE_SIZE

        .IIF    GREATER         <RESTART_TABLE_SIZE - STATE_VECTOR_SIZE> ,-
                .ERROR  ; Restart state code too large

        .ENDC

        .ENDC

        .ENDM   END_MARK_POINT
```

```
;+
;       SIGN_EXTEND - Perform sign-extended assignment
;
;       The VAX-11 MACRO assembler does not understand signed byte or word
;       quantities. It treats all quantities as longwords (zero extended if
;       necessary). This macro allows sign-extended byte or word assignments.
;
;       When an assignment of the form
;
;               SYMBOL = EXPRESSION
;
;       is required, treating EXPRESSION as a signed byte, the macro call
;
;               SIGN_EXTEND     EXPRESSION , SYMBOL , [BYTE]
;
;       performs the assignment, padding the upper 24 bytes with zero if the
;       expression is in the range 0 to 127 and padding the upper 24 bite with
;       ones if the expression s in the range -128 to -1 (128 to 255). The third
;       parameter, BYTE, is not necessary.
;
;       If a signed word assignment is needed, the SIGN_EXTEND macro is invoked
;       in the same way, including WORD as the optional third parameter.
;-

        .MACRO  SIGN_EXTEND     NUMBER,RESULT,TYPE=BYTE
        .IF     IDENTICAL       <TYPE>,BYTE
                .IF     EQUAL   <NUMBER & ^X80>
                RESULT = NUMBER
                .IF_FALSE
                RESULT = NUMBER ! ^XFFFFFF00
                .ENDC
        .IF_FALSE
                .IF     IDENTICAL       <TYPE>,WORD
                .IF     EQUAL   <NUMBER & ^X8000>
                RESULT = NUMBER
                .IF_FALSE
                RESULT = NUMBER ! ^XFFFF0000
                .ENDC
                .IF_FALSE
                .ERROR          ; TYPE parameter must be BYTE or WORD
                .ENDC
        .ENDC
        .ENDM   SIGN_EXTEND
```

```
;+
;       ESTABLISH_HANDLER - Load Handler Address into R10
;
;       This macro simply loads a packing routine address into R10 so that
;       intercepted exceptions can be dispatched to the correct instruction
;       specific routine for putting an instruction into a consistent state.
;
;       The only reason that this exists in a macro is to allow the reference
;       to the packing routine to be disabled when creating the subset
;       emulator for the bootstrap.
;-

        .MACRO  ESTABLISH_HANDLER       HANDLER_ADDRESS
        .IF     NOT_DEFINED     BOOT_SWITCH
        MOVAB   HANDLER_ADDRESS,R10
        .ENDC
        .ENDM   ESTABLISH_HANDLER
```

```
;+
;       ROPRAND_CHECK - Insure that digit count is LEQU 31
;
;       The ROPRAND_CHECK macro determines whether the length of a packed
;       decimal string is larger than the allowed length of 31. If an illegal
;       length is detected, then special code is invoked that will reflect a
;       reserved operand abort exception back to the caller. The macro is
;       defined in such a way that it is possible for multiple invocations in
;       a small block of code to use the same BRW instruction.
;-

        .MACRO  ROPRAND_CHECK   LEN,?OK

            CMPW        LEN,#31

            .IF         DEFINED         ...ROPRAND...
                .IF     LESS_EQUAL      < < . - ...ROPRAND... > - <128-2> >
                BGTRU   ...ROPRAND...
                .IF_FALSE               ; IF < . - ...ROPRAND... > GTRU 128
                BLEQU   OK
                ...ROPRAND... = .
                BRW     DECIMAL_ROPRAND
OK:
                .ENDC                   ; Is ...ROPRAND... within 128 bytes
            .IF_FALSE                   ; IF ...ROPRAND... is not defined
            BLEQU       OK
            ...ROPRAND... = .
            BRW         DECIMAL_ROPRAND
OK:
            .ENDC                       ; Is ...ROPRAND... defined

            MOVZWL      LEN,LEN

        .ENDM   ROPRAND_CHECK
```

```
;+
;       CASE - Macro for generating CASE instruction and case table
;
;       CASE    SRC,<DISPATCH LIST>,[TYPE=W/B/L],[LIMIT=#0],[NMODE=S^#]
;-

        .MACRO  CASE,SRC,DISPLIST,TYPE=W,LIMIT=#0,NMODE=S^#,?BASE,?MAX
        CASE'TYPE       SRC,LIMIT,NMODE'<<MAX-BASE>/2>-1
BASE:
        .IRP    EP,<DISPLIST>
        .SIGNED_WORD    EP-BASE
        .ENDR
MAX:
        .ENDM
```

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