E 14

```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
```

```
**FILE**ID**PATMAC



PPPPPPPP      AAAAAA    TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
PPPPPPPP      AAAAAA    TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
PP      PP  AA      AA      TT      MMMM  MMMM  AA      AA  CC
PP      PP  AA      AA      TT      MMMM  MMMM  AA      AA  CC
PP      PP  AA      AA      TT      MM  MM  MM  AA      AA  CC
PP      PP  AA      AA      TT      MM  MM  MM  AA      AA  CC
PPPPPPPP    AA      AA      TT      MM      MM  AA      AA  CC
PPPPPPPP    AA      AA      TT      MM      MM  AA      AA  CC
PP          AAAAAAAAAA      TT      MM      MM  AAAAAAAAAA  CC
PP          AAAAAAAAAA      TT      MM      MM  AAAAAAAAAA  CC
PP          AA      AA      TT      MM      MM  AA      AA  CC              ....
PP          AA      AA      TT      MM      MM  AA      AA  CC              ....
PP          AA      AA      TT      MM      MM  AA      AA    CCCCCCCC      ....
PP          AA      AA      TT      MM      MM  AA      AA    CCCCCCCC      ....


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
```

PA
VO

```
0 %TITLE 'Instruction decoder'
0 MODULE PATMAC (
0               %IF %VARIANT EQL 1
0               %THEN
0                       ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
0               %FI
0               IDENT = 'V04-000') =
1 BEGIN
1
1 !++
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1 ! FACILITY:     PATCH
1
1 !
1 ! FUNCTIONAL DESCRIPTION:       VAX INSTRUCTION DECODER.
1 !
1 ! Version:      V0218
1 !
1 ! Author:       KEVIN PAMMETT, 2-MAR-77: Version 00
1 !
1 ! Modified by:
1 !
1 !       V0218   CNH0013         Chris Hume              27-Aug-1979     13:30
1 !               Added double byte OPcode support.  Changed use of PAT$CONV_R_50
1 !               to the RTL routine R50ASC.
1 !
1 !       V0217   CNH0008         Chris Hume              28-Jun-1979     14:00
1 !               Fix CASE replacement bug and disallow relocation of these
1 !               instructions.  (PATMAI.B32 V0222, PATACT.B32 V0124,
1 !               PATEXA.B32 V0120, PATMSG.MDL V0202)
1 !
1 !       V0216   KDM0011         KATHLEEN D. MORSE       27-NOV-1978     10:25
1 !               Special case BR_LG in the OP_BR_TYPE field.
1 !
```

PA
V0

```
1 ! Revision history:
1 !
1 !  NO   DATE            PROGRAMMER              PURPOSE
1 !  --   ----            ----------              -------
1 !
1 !  00   20-OCT-77       K.D. MORSE              ADAPT VERSION 15 FOR PATCH
1 !  01   31-OCT-77       K.D. MORSE              ADAPT FOR MAPPED ADDRESSES.
1 !  02   12-DEC-77       K.D. MORSE              CHANGE BRANCH DISPLACEMENT
1 !                                               FROM ".+X" TO "BR  Y"
1 !  03   28-DEC-77       K.D. MORSE              REPLACE PAT$OUT_VALUE CALLS. (18)
1 !  04   5-JAN-78        K.D. MORSE              ADD CODE TO SPECIAL-CASE CASE
1 !                                               INSTRUCTIONS TO PRINT OUT THE
1 !                                               'DATA' FOLLOWING THE CASE
1 !                                               INSTRUCTION STREAM.  ALSO TO
1 !                                               ENABLE COMPUTING THE NEXT
1 !                                               INSTRUCTION ADDRESS. (16)
1 !                                               ADD OUT_BRNCH_OPRND. (16)
1 !                                               CHANGE PAT$INS_DECODE ARGS TO
1 !                                               ENABLE MULTI-LINE OUTPUT. (16)
1 !                                               NO CHANGES FOR 17.
1 !  05   24-JAN-78       K.D. MORSE              NO CHANGES FOR 19-20.
1 !  06   31-JAN-78       K.D. MORSE              CHANGED MODULE SO THAT IT WILL
1 !                                               UNDERSTAND INSTRUCTIONS NOT AT
1 !                                               THE PC BUT IN A BUFFER, I.E.,
1 !                                               STREAM_PTR IS NOT NECESSARILY
1 !                                               EQUAL TO INS_PC.  INS_PC IS NEW
1 !                                               PARAMETER.
1 !  07   17-MAR-78       K.D. MORSE              MOVE CODE TO OUTPUT A LEADING
1 !                                               0 FOR HEX NUMBERS TO PAT$SV\
1 !                                               PAT$OUT_NUM_VAL.  THIS SHOULD
1 !                                               FIX THE DISPLAY OF NEGATIVE
1 !                                               DISPLACEMENTS WHEN SYMBOLIC
1 !                                               OUTPUT IS ENABLED.
1 !  08   24-MAR-78       K.D. MORSE              NO CHANGES FOR VERS 21.
1 !  09   07-APR-78       K.D. MORSE              THE @# ADDRESSING MODE NOW TRIES
1 !                                               TO SYMBOLIZE THE OPERAND. (22)
1 !  10   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
1 !  11   28-APR-78       K.D. MORSE              ADD ASSEMBLER DIRECTIVE OUTPUT
1 !                                               BY CHECKING ASD TABLE, CHK_ASD_TBL.
1 !  12   18-MAY-78       K.D. MORSE              NO CHANGES FOR VERS 23.
1 !  13   13-JUN-78       K.D. MORSE              ADD FAO COUNTS TO SIGNALS.
1 !  14   21-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 23-24.
1 !                                               NOW PUT OUT ONLY EFFECTIVE
1 !                                               OPERAND FOR BRANCH AND CASE
1 !                                               OPERANDS.  DELETE ROUTINE
1 !                                               OUT_BRNCH_OPRND. (25)
1 !                                               FIX CASE BUG--SET CASE_FLAG
1 !                                               ONLY ON 3RD OPERAND. (26)
1 !  15   28-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 27.
1 !
1 !--
```

```
1 %SBTTL 'Module declarations'
1 !
1 ! TABLE OF CONTENTS
1 !
1 FORWARD ROUTINE
1         PAT$INS_DECODE,                          ! Decode an instruction
1         INS_OPERAND,                             ! Print out an operand reference
1         DISPLACEMENT,                            ! Extract displacements from instructions
1         BRANCH_TYPE,                             ! Decide and handle branch type addressing
1         INS_CONTEXT,                             ! Decide what context this instruction is
1         PUT_REG : NOVALUE,                       ! Print a register reference
1         CHK_ASD_TBL;                             ! Searches ASD table for specific PC
1
1 LIBRARY 'SYS$LIBRARY:STARLET.L32';
1 REQUIRE 'SRC$:PATPCT.REQ';
1 REQUIRE 'SRC$:VAXOPS.REQ';                       ! Literals and macros related to opcodes
1 REQUIRE 'SRC$:SYSLIT.REQ';                       ! Literals needed to call system services
1 REQUIRE 'SRC$:VXSMAC.REQ';                       ! Widely-used standard literals
1 REQUIRE 'SRC$:PATGEN.REQ';
1 REQUIRE 'SRC$:VAXERR.REQ';                       ! Error codes
1 REQUIRE 'SRC$:PREFIX.REQ';                       ! Structure macros
1 REQUIRE 'SRC$:PATPRE.REQ';                       ! ASD structure definition
1
1 EXTERNAL ROUTINE
1         PAT$GET_VALUE : NOVALUE,                 ! Gets value from image byte stream
1         PAT$MAP_ADDR : NOVALUE,                  ! Maps an image address
1         R50ASC,                                  ! Convert from RAD50
1         PAT$FAO_PUT : NOVALUE,                   ! Formatted I/O to terminal
1         PAT$OUT_NUM_VAL : NOVALUE,               ! Output values as numbers
1         PAT$OUT_SYM_VAL : NOVALUE,               ! Output values as numerics or symbols
1         PAT$OUT_PUT : NOVALUE;                   ! Flush the output buffer
1
1 EXTERNAL
1         PAT$CP_OUT_STR : REF VECTOR[,BYTE],      ! Points to current output buffer
1         PAT$GB_OPINFO : OPCODE_TBL,
1         PAT$GB_MOD_PTR : REF VECTOR[,BYTE],      ! Mode data structure pointer
1         PAT$GL_BUF_SIZ,                          ! Holds character count of output buffer
1         PAT$GL_LAST_VAL;                         ! Branch instructions overwrite this so that
1                                                  ! the user can "EX \" to trace thru a branch
```

```
;  150          1193  1 !
;  151          1194  1 ! Literals used only in this module
;  152          1195  1 !
;  153          1196  1
;  154          1197  1 LITERAL
;  155          1198  1         ROUND_BRACKETS  = 0,                                    ! These are all flag parameters to
;  156          1199  1         SQUARE_BRACKETS = 2,                                    !  the routine 'PUT_REG'.
;  157          1200  1         NO_BRACKETS     = 1;
;  158          1201  1
;  159          1202  1 MACRO
;  160        M 1203  1         PUTC(C) =                                               ! Put 1 char into the output buffer
;  161        M 1204  1                 BEGIN
;  162        M 1205  1                 (.PAT$CP_OUT_STR)<0,8> = C;
;  163        M 1206  1                 PAT$CP_OUT_STR = .PAT$CP_OUT_STR + 1;
;  164        M 1207  1                 PAT$GL_BUF_SIZ = .PAT$GL_BUF_SIZ + 1;
;  165          1208  1                 END %;
;  166          1209  1
;  167          1210  1 !++
;  168          1211  1 ! OWN STORAGE
;  169          1212  1 !--
;  170          1213  1 OWN
;  171          1214  1         CASE_FLAG,                                              ! Flag to special-case CASE instructions
;  172          1215  1         MAP_FLAG;                                               ! Flag whether or not to map the stream addr
```

```
%SBTTL 'PAT$INS_DECODE - Instructions ==> ASCII'
GLOBAL ROUTINE PAT$INS_DECODE( STREAM_PNTR, OUTPUT_BUFFER, INS_PC, ASM_DIR_TBL, CASE_TBL) =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine is the entry point for this module.
!
!       This routine examines a byte stream that it is passed a pointer to, and
!       tries to output what instructions this corresponds to symbolically.
!
! CALLING SEQUENCE:
!
!       PAT$INS_DECODE ();
!
! INPUTS:
!
!       STREAM_PNTR     - A byte pointer to the supposed instruction
!                         stream (unmapped address or buffer address).
!       OUTPUT_BUFFER   - This is a pointer to the beginning of the
!                         current output buffer.
!       INS_PC          - PC for which instruction is encoded
!       ASM_DIR_TBL     - Address of assembler directive table descriptor
!       CASE_TBL        - TRUE => Print CASE dispatch tables
!
! IMPLICIT INPUTS:
!
!       PAT$GB_OPINFO   - Data vector that contains the instruction
!                         mneonics and related information.
!       PAT$CP_OUT_STR  - Points into current output buffer.
!       PAT$GL_BUF_SIZ  - Holds character count in output buffer.
!
! OUTPUTS:
!
!       none.
!
! IMPLICIT OUTPUTS:
!
!       none.
!
! ROUTINE VALUE:
!
!       This routine returns a pointer to the beginning of the next instruction.
!       In case there is a need to differentiate some other reasons for
!       returning, the returned values are actually macros:
!
!       DETECTED:                       RETURNED:
!
!       -UNKNOWN INSTRUCTION            INS_UNKNOWN
!       -RESERVED INSTRUCTION           INS_RESERVED
!       -CAN'T READ INSTRUCTION         INS_UNREADABLE
!
! SIDE EFFECTS:
!
!       The current output buffer pointer is incremented, the character
!       representation of the instruction having been stuffed into the buffer.
```

```
1 !        The count of the output buffer is also incremented.
1 !--
1
2 BEGIN
2
2 MACRO                                                             ! Local macros -- see 'routine value' above
2         INS_UNREADABLE  = 0 %,
2         INS_UNKNOWN     = 0 %,
2         INS_RESERVED    = 0 %;
2
2 MAP
2         INS_PC : REF VECTOR[,LONG],                               ! Effect a REF LONGWORD, so can update to ne
2         STREAM_PNTR : REF VECTOR[,BYTE];
2
2 LOCAL
2         ASD_TBL_PTR: REF BLOCK[,BYTE],                            ! Points to the ASD entry matching PC
2         STREAM_PTR: REF BLOCK[,BYTE],                             ! Points to the unmapped instr stream
2         MAP_STREAM_PTR : REF VECTOR[,BYTE],                       ! Points to the mapped instr stream
2         ISE_ADDR,                                                 ! Address of ISE
2         OPCODE,                                                   ! INstruction opcode
2         OPRNDS;                                                   ! Number of operands for instruction
2
2 !++
2 ! Determine if the instruction stream is at the PC it was encoded for or if
2 ! it is in a buffer.  Then set a MAP_FLAG indicating whether or not to map
2 ! STREAM_PTR in order to access the byte stream.
2 !--
3 IF (.INS_PC[0] EQLA .STREAM_PNTR)
2 THEN
2         MAP_FLAG = TRUE
2 ELSE
2         MAP_FLAG = FALSE;
2
2 !++
2 ! Use an OWN copy of the formal, STREAM_PNTR, because the compiler does not
2 ! do this automatically, and because this module writes into this variable.
2 !--
2 STREAM_PTR = .STREAM_PNTR;
2 IF .MAP_FLAG                                                      ! Is instruction at PC?
2 THEN
2         PAT$MAP_ADDR(.STREAM_PTR, MAP_STREAM_PTR, ISE_ADDR)       ! Yes, get mapped address
2 ELSE
2         MAP_STREAM_PTR = .STREAM_PNTR;                            ! No, use buffer address
2
2 !++
2 ! Set up to special-case CASE instructions.
2 !--
2 CASE_FLAG = 0;
2
2 !++
2 ! Check if the PC to be output is known to contain an assembler directive.
2 ! If so, then CHK_ASD_TBL finds the appropriate "OPCODE" to offset into the
2 ! OPINFO table and also the pointer into the ASD table.
2 !--
3 IF ((OPCODE = CHK_ASD_TBL(.INS_PC[0], ASD_TBL_PTR, .ASM_DIR_TBL)) EQL FALSE)
2 THEN
3         BEGIN
```

```
3         !++
3         ! The instruction is not an assembler directive.  Therefore, pick up
3         ! the opcode and check it for validity.  Then increment the instruction
3         ! pointers past the opcode.
3         ! NOTE:  A MAPPED ADDRESS MAY BE DOTTED ONLY IF IT IS DOTTED TO ACQUIRE
3         ! ONE AND ONLY ONE BYTE.
3         !--
3
3         OPCODE = .MAP_STREAM_PTR[0];
3
3         IF .OPCODE EQL %X'FD'
3         THEN
4                 BEGIN                                           ! Check to see if 2 byte OPcode.
4                 OPCODE = .MAP_STREAM_PTR[1]^8 + .OPCODE;        ! It is.  Get the next byte of OPcode.
4                 STREAM_PTR = .STREAM_PTR + 1;
4                 INS_PC[0] = .INS_PC[0] + 1;
3                 END;
3         !++
3         ! Make sure that this is a recognized opcode, i.e., the number of expected
3         ! operands is known.
3         !--
4         IF( .PAT$GB_OPINFO[ .OPCODE, OP_NUMOPS] EQL NOT_AN_OP )
3         THEN
3                 !++
3                 ! The opcode is reserved, so not enough is known about it to go any further.
3                 !--
3                 RETURN(INS_RESERVED);
3
3         STREAM_PTR = .STREAM_PTR + 1;
3         INS_PC[0] = .INS_PC[0] + 1;
2         END;
2
2 !++
2 ! Output the character sequence which corresponds to the opcode.
2 ! Also put out two spaces since some opcodes take up the full OP_CH_SIZE
2 ! field printed, above.
2 !--
2 R50ASC( %REF(OP_CH_SIZE), PAT$GB_OPINFO[ .OPCODE, OP_NAME], .PAT$CP_OUT_STR );
2 PAT$CP_OUT_STR = .PAT$CP_OUT_STR + OP_CH_SIZE;
2 PAT$GL_BUF_SIZ = .PAT$GL_BUF_SIZ + OP_CH_SIZE;
2
2 PAT$FAO_PUT( UPLIT( %ASCIC '  ' ) );
2
2 !++
2 ! Check if this is a case instruction.
2 !--
3 IF (.OPCODE EQL OP_CASEB) OR (.OPCODE EQL OP_CASEW) OR (.OPCODE EQL OP_CASEL)
2 THEN
2         CASE_FLAG = -1;
2
2 !++
2 ! Loop, encoding how each operand is referenced.
2 !--
3 IF ((OPRNDS = .PAT$GB_OPINFO[.OPCODE, OP_NUMOPS]) EQL ASM_DIR_OP)
2 THEN
2         OPRNDS = .ASD_TBL_PTR[ASD$B_NUM_OPRND];
2 INCR I FROM 1 TO .OPRNDS
```

```
2 DO
3         BEGIN
4         IF( (STREAM_PTR = INS_OPERAND( .STREAM_PTR, .I, .OPCODE, INS_PC[0] )) EQL 0 )
3         THEN
3                 RETURN(INS_UNREADABLE);                         ! Decoding failure - probably due to accessa
4         IF (.I NEQ 0) AND (.I LSS .OPRNDS)
3         THEN
3                 PUTC(',');
2         END;
2
2
2 !++
2 ! CASE instructions are special-cased as they do not follow the syntax of
2 ! other instructions, namely an opcode followed by a fixed number of operands.
2 ! They are followed by N+1 words (offsets), where N is the last operand of the
2 ! instruction.  Therefore this case can only be handled if the operand was
2 ! given as a literal.  If this is TRUE, the offsets are printed.
2 !--
2 IF .CASE_TBL
2 THEN
3         BEGIN
3         LOCAL
3                 CASE_OFFSET : SIGNED WORD;                      ! Buffer to hold offsets
3
4         IF (.CASE_FLAG GTR 0)
3         THEN
3                 !++
3                 ! The flag contains N+1.  There are N+1 offsets to print.
3                 !--
3                 INCR I FROM 1 TO .CASE_FLAG
3                 DO
4                 BEGIN
4                 !++
4                 ! Loop, getting each offset and printing one offset per line.
4                 ! Update the instruction-stream pointer after each offset.
4                 !--
4                 IF .MAP_FLAG                                    ! Is instruction at PC?
4                 THEN
4                         PAT$GET_VALUE (.STREAM_PTR, A_WORD, CASE_OFFSET) ! Yes, map address
4                 ELSE
4                         CASE_OFFSET = .STREAM_PTR[0,0,16,1];    ! No, take offset from buffer
4                 PAT$OUT_PUT(.OUTPUT_BUFFER);
4                 PAT$CP_OUT_STR = .OUTPUT_BUFFER + 1;
4                 PAT$GL_BUF_SIZ = 0;
4                 PAT$FAO_PUT(UPLIT (%ASCIC '!_!_'));
4                 PAT$OUT_SYM_VAL(.INS_PC[0] + .CASE_OFFSET, LONG_LENGTH, NO_OVERRIDE);
4                 STREAM_PTR = .STREAM_PTR + A_WORD;
3                 END;
3         INS_PC[0] = .INS_PC[0] + A_WORD*.CASE_FLAG              ! Advance over the table
2         END;
2
2 !++
2 ! Return a pointer to the beginning of the next instruction.
2 !--
2 RETURN(.STREAM_PTR);
1 END;
```

```
                                                                   .TITLE   PATMAC Instruction decoder
                                                                   .IDENT   \V04-000\

                                                                   .PSECT   _PAT$PLIT,NOWRT,NOEXE,0

                               00  20  20  02  00000 P.AAA:   .ASCII   <2>\  \<0>                                        ;
               00  00  00  5F  21  5F  21  04  00004 P.AAB:   .ASCII   <4>\!_!_\<0><0><0>                                ;

                                                                   .PSECT   _PAT$OWN,NOEXE,2

                                               00000 CASE_FLAG:
                                                                   .BLKB    4
                                               00004 MAP_FLAG:
                                                                   .BLKB    4

                                                    ISE$C_SIZE==          20
                                                    TXT$C_SIZE==          4
                                                    PAL$C_SIZE==          16
                                                    ASD$C_SIZE==          9
                                                    FWR$C_SIZE==          24
                                                                   .EXTRN   PAT$GET_VALUE, PAT$MAP_ADDR
                                                                   .EXTRN   R50ASC, PAT$FAO_PUT
                                                                   .EXTRN   PAT$OUT_NUM_VAL
                                                                   .EXTRN   PAT$OUT_SYM_VAL
                                                                   .EXTRN   PAT$OUT_PUT, PAT$CP_OUT_STR
                                                                   .EXTRN   PAT$GB_OPINFO, PAT$GB_MOD_PTR
                                                                   .EXTRN   PAT$GL_BUF_SIZ, PAT$GL_LAST_VAL
                                                                   .EXTRN   PAT$GB_OPINFO1, PAT$GB_OPINFO2

                                                                   .PSECT   _PAT$CODE,NOWRT,2

                                         0FFC  00000               .ENTRY   PAT$INS_DECODE, Save R2,R3,R4,R5,R6,R7,R8,- ; 1217
                                                                            R9,R10,R11
                      5B 00000000G   EF   9E  00002               MOVAB    PAT$GL_BUF_SIZ, R11                         ;
                      5A 00000000G   EF   9E  00009               MOVAB    PAT$GB_OPINFO2+4, R10                       ;
                      59 00000000G   EF   9E  00010               MOVAB    PAT$GB_OPINFO1+4, R9                        ;
                      58 00000000G   EF   9E  00017               MOVAB    PAT$CP_OUT_STR, R8                          ;
                      57 00000000'   EF   9E  0001E               MOVAB    MAP_FLAG, R7                                ;
                      5E             14   C2  00025               SUBL2    #20, SP                                     ;
                      54        0C   AC   D0  00028               MOVL     INS_PC, R4                                  ; 1300
                04    AC             64   D1  0002C               CMPL     (R4), STREAM_PNTR                           ;
                                     05   12  00030               BNEQ     1$                                          ;
                      67             01   D0  00032               MOVL     #1, MAP_FLAG                                ; 1302
                                     02   11  00035               BRB      2$                                          ;
                                     67   D4  00037 1$:           CLRL     MAP_FLAG                                    ; 1304
                      56        04   AC   D0  00039 2$:           MOVL     STREAM_PNTR, STREAM_PTR                     ; 1310
                      11             67   E9  0003D               BLBC     MAP_FLAG, 3$                                ; 1311
                                04   AE   9F  00040               PUSHAB   ISE_ADDR                                    ; 1313
                                0C   AE   9F  00043               PUSHAB   MAP_STREAM_PTR                              ;
                                     56   DD  00046               PUSHL    STREAM_PTR                                  ;
           00000000G  EF             03   FB  00048               CALLS    #3, PAT$MAP_ADDR                            ;
                                     05   11  0004F               BRB      4$                                          ;
                08    AE        04   AC   D0  00051 3$:           MOVL     STREAM_PNTR, MAP_STREAM_PTR                 ; 1315
                                FC   A7   D4  00056 4$:           CLRL     CASE_FLAG                                   ; 1320
                                10   AC   DD  00059               PUSHL    ASM_DIR_TBL                                 ; 1327
                                10   AE   9F  0005C               PUSHAB   ASD_TBL_PTR                                 ;
```

```
                                       64  DD 0005F         PUSHL   (R4)                                        ;
                  00000000V  EF        03  FB 00061         CALLS   #3, CHK_ASD_TBL                             ;
                             52        50  D0 00068         MOVL    R0, OPCODE                                  ;
                                       46  12 0006B         BNEQ    9$                                          ;
                             50   08   AE  D0 0006D         MOVL    MAP_STREAM_PTR, R0                          ; 1338
                             52        60  9A 00071         MOVZBL  (R0), OPCODE                                ;
                  000000FD   8F        52  D1 00074         CMPL    OPCODE, #253                                ; 1340
                                       0F  12 0007B         BNEQ    5$                                          ;
                             50   01   A0  9A 0007D         MOVZBL  1(R0), R0                                   ; 1343
                  50         50        08  78 00081         ASHL    #8, R0, R0                                  ;
                             52        50  C0 00085         ADDL2   R0, OPCODE                                  ;
                                       56  D6 00088         INCL    STREAM_PTR                                  ; 1344
                                       64  D6 0008A         INCL    (R4)                                        ; 1345
                         FD  8F        52  91 0008C 5$:     CMPB    OPCODE, #253                                ; 1351
                                       06  13 00090         BEQL    6$                                          ;
                             50      6942  7E 00092         MOVAQ   PAT$GB_OPINFO1+4[OPCODE], R0                ;
                                       09  11 00096         BRB     7$                                          ;
                  50         52   F8   8F  78 00098 6$:     ASHL    #-8, OPCODE, R0                             ;
                             50      6A40  7E 0009D         MOVAQ   PAT$GB_OPINFO2+4[R0], R0                    ;
FFFFFFFF   8F     60         04        00  EC 000A1 7$:     CMPV    #0, #4, (R0), #-1                           ;
                                       03  12 000AA         BNEQ    8$                                          ;
                                     0123  31 000AC         BRW     26$                                         ;
                                       56  D6 000AF 8$:     INCL    STREAM_PTR                                  ; 1358
                                       64  D6 000B1         INCL    (R4)                                        ; 1359
                                       68  DD 000B3 9$:     PUSHL   PAT$CP_OUT_STR                              ; 1367
                         FD  8F        52  91 000B5         CMPB    OPCODE, #253                                ;
                                       07  13 000B9         BEQL    10$                                         ;
                             50   FC A942  7E 000BB         MOVAQ   PAT$GB_OPINFO1[OPCODE], R0                  ;
                                       0A  11 000C0         BRB     11$                                         ;
                  50         52   F8   8F  78 000C2 10$:    ASHL    #-8, OPCODE, R0                             ;
                             50   FC AA40  7E 000C7         MOVAQ   PAT$GB_OPINFO2[R0], R0                      ;
                                       50  DD 000CC 11$:    PUSHL   R0                                          ;
                         08  AE        06  D0 000CE         MOVL    #6, 8(SP)                                   ;
                                  08   AE  9F 000D2         PUSHAB  8(SP)                                       ;
                  00000000G  EF        03  FB 000D5         CALLS   #3, R50ASC                                  ;
                             68        06  C0 000DC         ADDL2   #6, PAT$CP_OUT_STR                          ; 1368
                             6B        06  C0 000DF         ADDL2   #6, PAT$GL_BUF_SIZ                          ; 1369
                                00000000'  EF  9F 000E2     PUSHAB  P.AAA                                       ; 1371
                  00000000G  EF        01  FB 000E8         CALLS   #1, PAT$FAO_PUT                             ;
                  0000008F   8F        52  D1 000EF         CMPL    OPCODE, #143                                ; 1376
                                       12  13 000F6         BEQL    12$                                         ;
                  000000AF   8F        52  D1 000F8         CMPL    OPCODE, #175                                ;
                                       09  13 000FF         BEQL    12$                                         ;
                  000000CF   8F        52  D1 00101         CMPL    OPCODE, #207                                ;
                                       04  12 00108         BNEQ    13$                                         ;
                         FC  A7        01  CE 0010A 12$:    MNEGL   #1, CASE_FLAG                               ; 1378
                         FD  8F        52  91 0010E 13$:    CMPB    OPCODE, #253                                ; 1383
                                       06  13 00112         BEQL    14$                                         ;
                             50      6942  7E 00114         MOVAQ   PAT$GB_OPINFO1+4[OPCODE], R0                ;
                                       09  11 00118         BRB     15$                                         ;
                  50         52   F8   8F  78 0011A 14$:    ASHL    #-8, OPCODE, R0                             ;
                             50      6A40  7E 0011F         MOVAQ   PAT$GB_OPINFO2+4[R0], R0                    ;
           55     60         04        00  EE 00123 15$:    EXTV    #0, #4, (R0), OPRNDS                        ;
                  FFFFFFFE   8F        55  D1 00128         CMPL    OPRNDS, #-2                                 ;
                                       08  12 0012F         BNEQ    16$                                         ;
                             50   0C   AE  D0 00131         MOVL    ASD_TBL_PTR, R0                             ; 1385
                             55   08   A0  9A 00135         MOVZBL  8(R0), OPRNDS                               ;
```

```
                                      53  D4 00139 16$:    CLRL     I                                          ; 1386
                                      26  11 0013B         BRB      19$                                        ;
                                      14  BB 0013D 17$:    PUSHR    #^M<R2,R4>                                 ; 1389
                                      53  DD 0013F         PUSHL    I                                          ;
                                      56  DD 00141         PUSHL    STREAM_PTR                                 ;
             00000000V  EF            04  FB 00143         CALLS    #4, INS_OPERAND                            ;
                        56            50  D0 0014A         MOVL     R0, STREAM_PTR                             ;
                                      03  12 0014D         BNEQ     18$                                        ;
                                    0080  31 0014F         BRW      26$                                        ;
                                      53  D5 00152 18$:    TSTL     I                                          ; 1392
                                      0D  13 00154         BEQL     19$                                        ;
                        55            53  D1 00156         CMPL     I, OPRNDS                                  ;
                                      08  18 00159         BGEQ     19$                                        ;
                   00   B8            2C  90 0015B         MOVB     #44, @PAT$CP_OUT_STR                       ; 1394
                                      68  D6 0015F         INCL     PAT$CP_OUT_STR                             ;
                                      6B  D6 00161         INCL     PAT$GL_BUF_SIZ                             ;
        D6              53            55  F3 00163 19$:    AOBLEQ   OPRNDS, I, 17$                             ; 1386
                        63       14   AC  E9 00167         BLBC     CASE_TBL, 25$                              ; 1405
                        52       FC   A7  D0 0016B         MOVL     CASE_FLAG, R2                              ; 1411
                                      55  15 0016F         BLEQ     24$                                        ;
        55         08   AC            01  C1 00171         ADDL3    #1, OUTPUT_BUFFER, R5                      ; 1429
                                      53  D4 00176         CLRL     I                                          ;
                                      48  11 00178         BRB      23$                                        ;
                        10            67  E9 0017A 20$:    BLBC     MAP_FLAG, 21$                              ; 1423
                                 10   AE  9F 0017D         PUSHAB   CASE_OFFSET                                ; 1425
                                      02  DD 00180         PUSHL    #2                                         ;
                                      56  DD 00182         PUSHL    STREAM_PTR                                 ;
             00000000G  EF            03  FB 00184         CALLS    #3, PAT$GET_VALUE                          ;
                                      04  11 0018B         BRB      22$                                        ;
                   10   AE            66  B0 0018D 21$:    MOVW     (STREAM_PTR), CASE_OFFSET                  ; 1427
                                 08   AC  DD 00191 22$:    PUSHL    OUTPUT_BUFFER                              ; 1428
             00000000G  EF            01  FB 00194         CALLS    #1, PAT$OUT_PUT                            ;
                        68            55  D0 0019B         MOVL     R5, PAT$CP_OUT_STR                         ; 1429
                                      6B  D4 0019E         CLRL     PAT$GL_BUF_SIZ                             ; 1430
                         00000000'    EF  9F 001A0         PUSHAB   P.AAB                                      ; 1431
             00000000G  EF            01  FB 001A6         CALLS    #1, PAT$FAO_PUT                            ;
                        7E            04  7D 001AD         MOVQ     #4, -(SP)                                  ; 1432
                        50       18   AE  32 001B0         CVTWL    CASE_OFFSET, R0                            ;
        7E              64            50  C1 001B4         ADDL3    R0, (R4), -(SP)                            ;
             00000000G  EF            03  FB 001B8         CALLS    #3, PAT$OUT_SYM_VAL                        ;
                        56            02  C0 001BF         ADDL2    #2, STREAM_PTR                             ; 1433
        B4              53            52  F3 001C2 23$:    AOBLEQ   R2, I, 20$                                 ; 1416
                        50       FC   A7  D0 001C6 24$:    MOVL     CASE_FLAG, R0                              ; 1435
                        74          9440  3E 001CA         MOVAW    @(R4)+[R0], -(R4)                          ;
                        50            56  D0 001CE 25$:    MOVL     STREAM_PTR, R0                             ; 1441
                                          04 001D1         RET                                                 ;
                                      50  D4 001D2 26$:    CLRL     R0                                         ; 1442
                                          04 001D4         RET                                                 ;

; Routine Size:  469 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
1 %SBTTL 'INS_OPERAND - Output instruction''s operand'
1 ROUTINE INS_OPERAND( STREAM_PTR, INDEX, OPCODE, INS_PC ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Print out an instruction operand.
1 !
1 ! WARNING:
1 !
1 !       1) There is code in the 'DEFERRED' macro which will cease
1 !          to work when/if the representation of TRUE and FALSE are changed.
1 !       2) The local macros, below, check for the indicated addressing
1 !          modes only given that they appear in the code where they
1 !          do - i.e., the checks take advantage of what is known about
1 !          which cases already have been eliminated, etc.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       INS_OPERAND (STREAM_PTR, INDEX, OPCODE, INS_PC);
1 !
1 ! INPUTS:
1 !
1 !       STREAM_PTR      - A byte pointer to the first byte of the instruction stream
1 !                         which begins this operand.  This byte is the dominant
1 !                         mode.  This is an unmapped address.
1 !       INDEX           - Ordinal of which operand to decode.  This is needed to
1 !                         decide the 'CONTEXT' for this operand if PC-relative
1 !                         addressing mode is used.
1 !       OPCODE          - The opcode of instruction being decoded.
1 !                         (This parameter has already been validated.)
1 !       INS_PC          - PC for which this instruction was encoded
1 !       CASE_FLAG       - Non zero requests that this be loaded with the length
1 !                         of the case table (only if specified by a literal).
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       MAP_FLAG - TRUE if STREAM_PTR is an unmapped address (the PC),
1 !                  FALSE if STREAM_PTR is a temporary buffer address.
1
1 !
1 ! OUTPUTS:
1 !
1 !       The current operand is written into the current output buffer in
1 !       machine-language format.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       CASE_FLAG       - See INPUTS.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       -The instruction-stream byte pointer, incremented to reflect the number
1 !       of bytes used for this operand.  This pointer should point to the
1 !       beginning of either the next instruction, or the next operand,
1 !       depending on how many operands the current instruction has.
1 !       -If the operand cannot be decoded, FALSE is returned.
```

```
1 !
1 ! SIDE EFFECTS:
1 !
1 !        If the instruction pointer is updated incorrectly, then the supposed
1 !        next instruction will be wrong.  This will cause a completely misleading
1 !        'instruction' to be output on the next call to this routine.
1 !--
1
2 BEGIN
2
2 !++
2 ! Local macros used to check for the indicated addressing modes.
2 ! See 'WARNING:', above.
2 !--
2 MACRO
2         REGISTR(MODE) = (MODE EQL 5) %,                      ! Register mode addressing
2         DEFERRED(MODE) = ( MODE LSS 0 AND MODE ) %,          ! Those which begin with '@' are
2                                                              !  9 - @(RN)+,
2                                                              !  B - @BYTE(RN),
2                                                              !  D - @WORD(RN),
2                                                              !  F - @LONG(RN),
2                                                              ! or any of these + indexing
2                                                              ! The thing which is common to only these
2                                                              ! modes is that they all have the sign
2                                                              ! bit set and are odd!
2         AUTODEC(MODE) = (MODE EQL 7) %,                      ! See if mode is auto decrement.
2         AUTOINC(MODE) = (MODE LSS 0) %;                      ! mode is auto increment
2                                                              ! This check depends upon the fact that
2                                                              ! the mode was extracted with sign extension
2                                                              ! and that many of the other possibilities
2                                                              ! were already eliminated.
2
2 MAP
2         INS_PC : REF VECTOR[,LONG],                          ! Effect a REF LONG, enabling an update of t
2         STREAM_PTR : REF BLOCK[,BYTE];
2
2 LOCAL
2         STREAM_VALUE : BLOCK[4,BYTE],                        ! Values from instruction stream
2         NEW_STR_PTR,                                         ! New stream pointer
2         FLAG,                                                ! Indicates the type of displacement
2         DISPO : VECTOR[16,BYTE],                             ! The actual displacement
2         DISPL,                                               ! The low order longword of DISPO
2         DISP_SIZE,                                           ! The size, in bytes, of a displacement
2         DOM_OPRND,                                           ! Operand extracted from the
2                                                              !  dominant mode byte.  It may be Rn,
2                                                              !  Rx, or a literal (SRM notation).
2         DOM_MODE;                                            ! The primary addressing mode comes from
2                                                              !  this dominant byte as well.
2
2 !++
2 ! Consider the possibility of so-called 'branch type' addressing first before
2 ! anything else, because otherwise short literals cannot be differentiated
2 ! f.om byte displacement branches.
2 !--
3 IF( (NEW_STR_PTR = BRANCH_TYPE( .STREAM_PTR, .INDEX, .OPCODE, INS_PC[0] )) NEQ 0 )
2 THEN
2         RETURN( .NEW_STR_PTR );                              ! Success, return new stream pointer
```

```
2
2 !++
2 ! Extract the needed fields from the first byte of the operand specifier.
2 ! Extract some fields with sign extension because that makes various tests
2 ! more convenient.
2 !--
2 IF .MAP_FLAG                                                         ! Is the instruction at PC?
2 THEN
2         PAT$GET_VALUE(.STREAM_PTR, A_BYTE, STREAM_VALUE)             ! Yes, map address
2 ELSE
2         STREAM_VALUE = .STREAM_PTR[0, 0, (A_BYTE * BITS_PER_BYTE), 0]; ! No, get value from buffer
2 DOM_MODE = .STREAM_VALUE[ AMODE ];
2 DOM_OPRND = .STREAM_VALUE[ AREG ];
2
2 !++
2 ! Take special action for indexing mode.
2 !--
3 IF( .DOM_MODE EQL INDEXING_MODE )
2 THEN
3         BEGIN
3         !++
3         ! Handle indexing mode recursively.
3         !--
3         INS_PC[0] = .INS_PC[0] + 1;
4         IF( (STREAM_PTR = INS_OPERAND( STREAM_PTR[ NEXT_FIELD(1) ], .INDEX, .OPCODE, INS_PC[0] )) EQL 0 )
3         THEN
3                 RETURN(FALSE);                                       ! Read access failure
3         PUT_REG( .DOM_OPRND, SQUARE_BRACKETS );
3         RETURN( .STREAM_PTR );
2         END;
2
2 !++
2 ! Simple modes are easier:
2 !
2 ! First see if there will be a literal or displacement in the operand reference.
2 !--
3 IF( (STREAM_PTR = DISPLACEMENT( .STREAM_PTR, FLAG, DISPO, DISP_SIZE, .INDEX, .OPCODE, INS_PC[0] )) EQL 0 )
2 THEN
2         RETURN(FALSE);                                               ! Read access failure
2
2 DISPL = .DISPO< 0, MINU( .DISP_SIZE, A_LONGWORD) * BITS_PER_BYTE, 1>;
2
2 !++
2 ! Begin checking for the addressing modes which begin with special characters
2 ! that have to be printed first.  An attempt is made to handle different cases
2 ! first.
2 !--
3 IF (DEFERRED(.DOM_MODE))
2 THEN
3         PUTC('@')
2 ELSE
3         IF (AUTODEC(.DOM_MODE))
2         THEN
2                 PUTC('-');
2
2 !++
2 ! Next consider displacements or literals.  Whether or not this is the case
```

```
2 ! has already been determined in the call to 'DISPLACEMENT', above.
2 !--
3 IF (.FLAG)
2 THEN
3         BEGIN
3         !++
3         ! There is a literal, so print it.  The flag value returned by routine
3         ! DISPLACEMENT distinguishes when there should be a '#' as opposed to
3         ! when the number is actually a displacement off a register.
3         !--
4         IF (.FLAG GTR 0)
3         THEN
4                 BEGIN
4                 IF .DISP_SIZE GTR A_LONGWORD                    ! **** Temp
4                 THEN
4                         !++
4                         ! Literals bigger than a longword are not yet supported.
4                         !--
5                         BEGIN
5                         DISP_SIZE = A_LONGWORD;
5                         PUTC('?');
4                         END;
4                 PUTC('#');
4
4                 !++
4                 ! Except for @# mode, make .DOM_OPRND NEQ PC_REG so that later
4                 ! only checking that will also tell us that .FLAG is GTR 0.
4                 !--
5                 IF (NOT DEFERRED (.DOM_MODE))
4                 THEN
4                         DOM_OPRND = PC_REG + 1;
5                 IF (.CASE_FLAG NEQ 0) AND (.INDEX EQL 3)
4                 THEN
4                         CASE_FLAG = .DISPL + 1;
4                 END
3         ELSE
4                 BEGIN
4                 OWN
4                         DISPL_ID : VECTOR[4,BYTE]
4                         INITIAL( BYTE( 'B', 'W', '?', 'L') );
4
4                 !++
4                 ! Print an indication of the displacement size.
4                 !--
4                 PAT$FAO_PUT( UPLIT( %ASCIC '!AD^' ), 1, DISPL_ID[ .DISP_SIZE - 1 ] );
3                 END;
3
3         !++
3         ! If the register is the PC, then the absolute address is output.
3         !--
4         IF (.FLAG LSS 0) AND (.DOM_OPRND EQL PC_REG)
3         THEN
4                 BEGIN
4                 !++
4                 ! Pick up the displacement and make it into an effective address.
4                 !--
4                 DISP_SIZE = A_LONGWORD;
```

```
4                       DISPL = .DISPL + .INS_PC[0];
3                       END;
3
3
3               !++
3               ! Output here is the same as non-EFFECTIVE unless the (REG) is PC.
3               !--
4               IF( .DOM_OPRND EQL PC_REG )
3               THEN
3                       PAT$OUT_SYM_VAL( .DISPL, LONG_LENGTH, NO_OVERRIDE )
3               ELSE
4                       BEGIN
4                       !++
4                       ! Literals or real (non-PC) displacement modes.
4                       !--
4                       PAT$OUT_NUM_VAL(.DISPO, .DISP_SIZE, NO_OVERRIDE, TRUE);
5                       IF( .FLAG LSS 0 )
4                       THEN
4                               PUT_REG( .DOM_OPRND, ROUND_BRACKETS );
3                       END;
3               END
2       ELSE
2               !++
2               ! No literal or displacement therefore the operand must be a type of
2               ! register reference.  Sort out the few cases and print them.
2               !--
3               IF (REGISTR(.DOM_MODE))
2               THEN
2                       PUT_REG( .DOM_OPRND, NO_BRACKETS )
2               ELSE
3                       BEGIN
3                       PUT_REG( .DOM_OPRND, ROUND_BRACKETS );
4                       IF( AUTOINC( .DOM_MODE ) )
3                       THEN
3                               PUTC('+');
2                       END;
2       RETURN(.STREAM_PTR);                                    ! Return the new byte stream pointer
1       END;


                                                        .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

        00  00  00  5E  44  41  21  04  0000C P.AAC:    .ASCII  <4>\!AD^\<0><0><0>                      ;

                                                        .PSECT  _PAT$OWN,NOEXE,2

                                  42  00008 DISPL_ID:
                                                        .ASCII  \B\                                     ;
                                  57  00009             .ASCII  \W\                                     ;
                                  3F  0000A             .ASCII  \?\                                     ;
                                  4C  0000B             .ASCII  \L\                                     ;


                                                        .PSECT  _PAT$CODE,NOWRT,2
                                                        07FC 00000 INS_OPERAND:
                                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10              ; 1444
                                        5A 00000000V EF   9E 00002         MOVAB   PUT_REG, R10                                  ;
                                        59 00000000G EF   9E 00009         MOVAB   PAT$GL_BUF_SIZ, R9                            ;
                                        58 00000000' EF   9E 00010         MOVAB   CASE_FLAG, R8                                 ;
                                        57 00000000G EF   9E 00017         MOVAB   PAT$CP_OUT_STR, R7                            ;
                                        5E           1C   C2 0001E         SUBL2   #28, SP                                       ;
                                        54       10  AC   D0 00021         MOVL    INS_PC, R4                                    ; 1554
                                                     54   DD 00025         PUSHL   R4                                            ;
                                        7E       08  AC   7D 00027         MOVQ    INDEX, -(SP)                                  ;
                                                 04  AC   DD 0002B         PUSHL   STREAM_PTR                                    ;
                            00000000V   EF           04   FB 0002E         CALLS   #4, BRANCH_TYPE                               ;
                                                     50   D5 00035         TSTL    NEW_STR_PTR                                   ;
                                                     01   13 00037         BEQL    1$                                            ;
                                                          04 00039         RET                                                   ;
                                        10       04  A8   E9 0003A 1$:     BLBC    MAP_FLAG, 2$                                  ; 1563
                                                     5E   DD 0003E         PUSHL   SP                                            ; 1565
                                                     01   DD 00040         PUSHL   #1                                            ;
                                                 04  AC   DD 00042         PUSHL   STREAM_PTR                                    ;
                            00000000G   EF           03   FB 00045         CALLS   #3, PAT$GET_VALUE                             ;
                                                     04   11 0004C         BRB     3$                                            ;
                                        6E       04  BC   9A 0004E 2$:     MOVZBL  @STREAM_PTR, STREAM_VALUE                     ; 1567
           53            6E             04           04   EE 00052 3$:     EXTV    #4, #4, STREAM_VALUE, DOM_MODE                ; 1568
           52            6E             04           00   EF 00057         EXTZV   #0, #4, STREAM_VALUE, DOM_OPRND               ; 1569
                                        04           53   D1 0005C         CMPL    DOM_MODE, #4                                  ; 1574
                                                     1C   12 0005F         BNEQ    4$                                            ;
                                                     64   D6 00061         INCL    (R4)                                          ; 1580
                                                     54   DD 00063         PUSHL   R4                                            ; 1581
                                        7E       08  AC   7D 00065         MOVQ    INDEX, -(SP)                                  ;
                         7E       04    AC           01   C1 00069         ADDL3   #1, STREAM_PTR, -(SP)                         ;
                                  8E    AF           04   FB 0006E         CALLS   #4, INS_OPERAND                               ;
                                  04    AC           50   D0 00072         MOVL    R0, STREAM_PTR                                ;
                                                     22   13 00076         BEQL    5$                                            ;
                                                     02   DD 00078         PUSHL   #2                                            ; 1584
                                                   00F5   31 0007A         BRW     20$                                           ;
                                                     54   DD 0007D 4$:     PUSHL   R4                                            ; 1593
                                        7E       08  AC   7D 0007F         MOVQ    INDEX, -(SP)                                  ;
                                                 10  AE   9F 00083         PUSHAB  DISP_SIZE                                     ;
                                                 1C  AE   9F 00086         PUSHAB  DISP0                                         ;
                                                 1C  AE   9F 00089         PUSHAB  FLAG                                          ;
                                                 04  AC   DD 0008C         PUSHL   STREAM_PTR                                    ;
                            00000000V   EF           07   FB 0008F         CALLS   #7, DISPLACEMENT                              ;
                                  04    AC           50   D0 00096         MOVL    R0, STREAM_PTR                                ;
                                                     03   12 0009A 5$:     BNEQ    6$                                            ;
                                                   00F1   31 0009C         BRW     23$                                           ;
                                        50       04  AE   D0 0009F 6$:     MOVL    DISP_SIZE, R0                                 ; 1597
                                        04           50   D1 000A3         CMPL    R0, #4                                        ;
                                                     03   1B 000A6         BLEQU   7$                                            ;
                                        50           04   D0 000A8         MOVL    #4, R0                                        ;
                                        50           08   C4 000AB 7$:     MULL2   #8, R0                                        ;
           55        0C  AE             50           00   EE 000AE         EXTV    #0, R0, DISP0, DISPL                          ;
                                                     56   D4 000B4         CLRL    R6                                            ; 1604
                                                     53   D5 000B6         TSTL    DOM_MODE                                      ;
                                                     0C   18 000B8         BGEQ    8$                                            ;
                                                     56   D6 000BA         INCL    R6                                            ;
                                        07           53   E9 000BC         BLBC    DOM_MODE, 8$                                  ;
                                  00    B7       40  8F   90 000BF         MOVB    #64, @PAT$CP_OUT_STR                          ; 1606
```

```
                                09 11 000C4        BRB     9$                                  ;
               07               53 D1 000C6 8$:    CMPL    DOM_MODE, #7                        ; 1608
                                08 12 000C9        BNEQ    10$                                 ;
         00    B7               2D 90 000CB        MOVB    #45, @PAT$CP_OUT_STR                ; 1610
                                67 D6 000CF 9$:    INCL    PAT$CP_OUT_STR                      ;
                                69 D6 000D1        INCL    PAT$GL_BUF_SIZ                      ;
               03        08     AE E8 000D3 10$:   BLBS    FLAG, 11$                           ; 1616
                              0091 31 000D7        BRW     19$                                 ;
                         08     AE D5 000DA 11$:   TSTL    FLAG                                ; 1624
                                33 15 000DD        BLEQ    15$                                 ;
               04        04     AE D1 000DF        CMPL    DISP_SIZE, #4                       ; 1627
                                0C 15 000E3        BLEQ    12$                                 ;
         04    AE               04 D0 000E5        MOVL    #4, DISP_SIZE                       ; 1633
         00    B7               3F 90 000E9        MOVB    #63, @PAT$CP_OUT_STR                ; 1634
                                67 D6 000ED        INCL    PAT$CP_OUT_STR                      ;
                                69 D6 000EF        INCL    PAT$GL_BUF_SIZ                      ;
         00    B7               23 90 000F1 12$:   MOVB    #35, @PAT$CP_OUT_STR                ; 1636
                                67 D6 000F5        INCL    PAT$CP_OUT_STR                      ;
                                69 D6 000F7        INCL    PAT$GL_BUF_SIZ                      ;
               03               56 E9 000F9        BLBC    R6, 13$                             ; 1642
               03               53 E8 000FC        BLBS    DOM_MODE, 14$                       ;
               52               10 D0 000FF 13$:   MOVL    #16, DOM_OPRND                      ; 1644
                                68 D5 00102 14$:   TSTL    CASE_FLAG                           ; 1645
                                25 13 00104        BEQL    16$                                 ;
               03        08     AC D1 00106        CMPL    INDEX, #3                           ;
                                1F 12 0010A        BNEQ    16$                                 ;
               68        01     A5 9E 0010C        MOVAB   1(R5), CASE_FLAG                    ; 1647
                                19 11 00110        BRB     16$                                 ; 1624
               50        08     A8 9E 00112 15$:   MOVAB   DISPL_ID, R0                        ; 1658
                         04 BE40   9F 00116        PUSHAB  @DISP_SIZE[R0]                      ;
                                6E D7 0011A        DECL    (SP)                                ;
                                01 DD 0011C        PUSHL   #1                                  ;
                      00000000' EF 9F 0011E        PUSHAB  P.AAC                               ;
   00000000G   EF               03 FB 00124        CALLS   #3, PAT$FAO_PUT                     ;
                                53 D4 0012B 16$:   CLRL    R3                                  ; 1664
                         08     AE D5 0012D        TSTL    FLAG                                ;
                                0E 18 00130        BGEQ    17$                                 ;
                                53 D6 00132        INCL    R3                                  ;
               0F               52 D1 00134        CMPL    DOM_OPRND, #15                      ;
                                07 12 00137        BNEQ    17$                                 ;
         04    AE               04 D0 00139        MOVL    #4, DISP_SIZE                       ; 1670
               55               64 C0 0013D        ADDL2   (R4), DISPL                         ; 1671
               0F               52 D1 00140 17$:   CMPL    DOM_OPRND, #15                      ; 1678
                                0E 12 00143        BNEQ    18$                                 ;
               7E               04 7D 00145        MOVQ    #4, -(SP)                           ; 1680
                                55 DD 00148        PUSHL   DISPL                               ;
   00000000G   EF               03 FB 0014A        CALLS   #3, PAT$OUT_SYM_VAL                 ;
                                38 11 00151        BRB     22$                                 ;
                                01 DD 00153 18$:   PUSHL   #1                                  ; 1686
                                7E D4 00155        CLRL    -(SP)                               ;
                         0C     AE DD 00157        PUSHL   DISP_SIZE                           ;
                         18     AE DD 0015A        PUSHL   DISPO                               ;
   00000000G   EF               04 FB 0015D        CALLS   #4, PAT$OUT_NUM_VAL                 ;
               24               53 E9 00164        BLBC    R3, 22$                             ; 1687
                                7E D4 00167        CLRL    -(SP)                               ; 1689
                                07 11 00169        BRB     20$                                 ;
               05               53 D1 0016B 19$:   CMPL    DOM_MODE, #5                        ; 1697
```

```
                                09 12 0016E         BNEQ    21$                              ;
                                01 DD 00170         PUSHL   #1                               ; 1699
                                52 DD 00172 20$:    PUSHL   DOM_OPRND                        ;
                    6A          02 FB 00174         CALLS   #2, PUT_REG                      ;
                                12 11 00177         BRB     22$                              ;
                                7E D4 00179 21$:    CLRL    -(SP)                            ; 1702
                                52 DD 0017B         PUSHL   DOM_OPRND                        ;
                    6A          02 FB 0017D         CALLS   #2, PUT_REG                      ;
                    08          56 E9 00180         BLBC    R6, 22$                          ; 1703
              00    B7          2B 90 00183         MOVB    #43, @PAT$CP_OUT_STR             ; 1705
                                67 D6 00187         INCL    PAT$CP_OUT_STR                   ;
                                69 D6 00189         INCL    PAT$GL_BUF_SIZ                   ;
                    50      04  AC D0 0018B 22$:    MOVL    STREAM_PTR, R0                   ; 1707
                                   04 0018F         RET                                      ;
                                50 D4 00190 23$:    CLRL    R0                               ; 1708
                                   04 00192         RET                                      ;

; Routine Size:  403 bytes,    Routine Base:  _PAT$CODE + 01D5
```

```
1 %SBTTL 'BRANCH_TYPE - Handle branch operands'
1 ROUTINE BRANCH_TYPE( STREAM_PTR, INDEX, OPCODE, INS_PC ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       DECIDE IF THE CURRENT OPERAND IS USING BRANCH TYPE
1 !       ADDRESSING.  IF SO, PRINT OUT THE REFERENCE AND
1 !       LOOK AFTER ALL THE DETAILS.  OTHERWISE RETURN FALSE.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       BRANCH_TYPE ();
1 !
1 ! INPUTS:
1 !
1 !       STREAM_PTR      - AN UNMAPPED POINTER TO THE CURRENT DOMINANT
1 !                         MODE BYTE.
1 !       INDEX           - WHICH OPERAND (ORDINAL) BEING DECODED.
1 !       OPCODE          -THE OPCODE OF INSTRUCTION BEING DECODED.
1 !                        (This parameter has already been validated.)
1 !       INS_PC          - THE PC FOR WHICH THE INSTRUCTION WAS ENCODED
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$GB_OPINFC - THE OPCODE INFORMATION TABLE.
1 !
1 ! OUTPUTS:
1 !
1 !       IF THE CURRENT OPERAND IS A REFERENCE USING BRANCH TYPE
1 !       ADDRESSING, THIS REFERENCE IS PRINTED.  OTHERWISE THE
1 !       ROUTINE DOES NO OUTPUT.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       MAP_FLAG - TRUE IF STREAM_PTR IS EQUAL TO PC,
1 !                  FALSE IF STREAM_PTR IS A BUFFER.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       FALSE - IF THE CURRENT OPERAND IS NOT A BRANCH TYPE
1 !               (i.e. If the calling routine should continue on
1 !                further to decode the instruction.)
1 !       TRUE - non-zero, THE ADDRESS OF THE NEXT INSTRUCTION IS RETURNED.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE.
1 !--
1
2 BEGIN
2
2 MAP
2       INS_PC : REF VECTOR[,LONG],
2       STREAM_PTR : REF BLOCK[,BYTE];
2
2 LOCAL
```

```
2           N_OPS,                                                  ! NUMBER OF OPERANDS FOR CURRENT OPCODE
2           DISP_SIZE,                                              ! SIZE OF BRANCH OPERAND, IN BYTES.
2           DISPL,                                                  ! THE ACTUAL BRANCH DISPLACEMENT.
2           STREAM_VALUE;                                           ! VALUE OF BYTE STREAM FOR INSTRUCTION
2
2   !++
2   ! THERE IS NO POINT IN EVEN CONSIDERING BRANCH TYPE ADDRESSING UNLESS THIS IS
2   ! THE LAST OPERAND FOR THIS INSTRUCTION.
2   !--
2   IF ((N_OPS = .PAT$GB_OPINFO[ .OPCODE, OP_NUMOPS ]) NEQ .INDEX) AND
3      (.PAT$GB_OPINFO[.OPCODE, OP_NUMOPS] NEQ ASM_DIR_OP)
3
2   THEN
2           RETURN(FALSE);
2
2   !--
2   ! 0 IN THE OP_BR_TYPE FIELD INDICATES OPCODE HAS NO BRANCH TYPE OPERANDS.
2   !--
3   IF( (DISP_SIZE = .PAT$GB_OPINFO[ .OPCODE, OP_BR_TYPE ]) EQL NO_BR )
2   THEN
3           RETURN(FALSE)
2   ELSE
3           IF (.DISP_SIZE EQLU BR_LG)
2           THEN
2                   DISP_SIZE = A_LONGWORD;
2
2   !++
2   ! SUCCESS -- THIS IS A CASE OF BRANCH TYPE ADDRESSING. HANDLE THIS
2   ! BY EXTRACTING THE FIELD, (WITH SIGN EXTENSION AS PER SRM),  PRINTING
2   ! OUT THE REFERENCE, AND RETURNING A POINTER TO THE NEXT INSTRUCTION.  ALSO
2   ! UPDATE THE VARIABLE WHICH THE USER ACCESSES AS '\' - THE LAST VALUE DISPLAYED.
2   ! IN THIS CASE IT IS DEFINED AS THE VALUE TO BE THE BRANCH ADDRESS.
2   !--
2   IF .MAP_FLAG                                                    ! IS THE INSTRUCTION AT THE PC?
2   THEN
2           PAT$GET_VALUE(.STREAM_PTR, .DISP_SIZE, STREAM_VALUE)    ! YES, MAP ADDRESS
2   ELSE
2           STREAM_VALUE = .STREAM_PTR[0, 0, .DISP_SIZE*BITS_PER_BYTE, 1]; ! NO, GET VALUE FROM BUFFER
3   IF (.PAT$GB_OPINFO[.OPCODE, OP_NUMOPS] NEQ ASM_DIR_OP)
2   THEN
2           DISPL = .STREAM_VALUE<0,.DISP_SIZE*BITS_PER_BYTE,1>
2   ELSE
2           DISPL = .STREAM_VALUE<0,.DISP_SIZE*BITS_PER_BYTE,0>;
2   STREAM_PTR = .STREAM_PTR + .DISP_SIZE;
2   INS_PC[0] = .INS_PC[0] + .DISP_SIZE;
2   PAT$GL_LAST_VAL = .DISPL + .INS_PC[0];
2
2   !++
2   ! Put out the absolute branch operand.
2   !--
3   IF (.PAT$GB_OPINFO[.OPCODE, OP_NUMOPS] NEQ ASM_DIR_OP)
2   THEN
2           DISPL = .DISPL + .INS_PC[0];
2   PAT$OUT_SYM_VAL(.DISPL, LONG_LENGTH, NO_OVERRIDE);
2   RETURN( .STREAM_PTR );
1   END;
```

```
                                                     00FC 00000 BRANCH_TYPE:
                                                                       .WORD   Save R2,R3,R4,R5,R6,R7                          ; 1710
                                  57 00000000G  EF   9E 00002          MOVAB   PAT$GB_OPINFO1+4, R7                            ;
                                  56 00000000G  EF   9E 00009          MOVAB   PAT$GB_OPINFO2+4, R6                            ;
                                  5E            04   C2 00010          SUBL2   #4, SP                                          ;
                                  53        0C  AC   D0 00013          MOVL    OPCODE, R3                                      ; 1775
                                                55   D4 00017          CLRL    R5                                              ;
                         FD       8F            53   91 00019          CMPB    R3, #253                                        ;
                                                08   13 0001D          BEQL    1$                                              ;
                                                55   D6 0001F          INCL    R5                                              ;
                                  50          6743   7E 00021          MOVAQ   PAT$GB_OPINFO1+4[R3], R0                        ;
                                                09   11 00025          BRB     2$                                              ;
                 50               53        F8  8F   78 00027 1$:      ASHL    #-8, R3, R0                                     ;
                                  50          6640   7E 0002C          MOVAQ   PAT$GB_OPINFO2+4[R0], R0                        ;
       50        60               04            00   EE 00030 2$:      EXTV    #0, #4, (R0), N_OPS                             ;
                         08       AC            50   D1 00035          CMPL    N_OPS, INDEX                                    ;
                                                20   13 00039          BEQL    6$                                              ;
                                  06            55   E9 0003B          BLBC    R5, 3$                                          ; 1776
                                  50          6743   7E 0003E          MOVAQ   PAT$GB_OPINFO1+4[R3], R0                        ;
                                                09   11 00042          BRB     4$                                              ;
                 50               53        F8  8F   78 00044 3$:      ASHL    #-8, R3, R0                                     ;
                                  50          6640   7E 00049          MOVAQ   PAT$GB_OPINFO2+4[R0], R0                        ;
FFFFFFFE   8F    60               04            00   EC 0004D 4$:      CMPV    #0, #4, (R0), #-2                               ;
                                                03   13 00056          BEQL    6$                                              ;
                                              00B4   31 00058 5$:      BRW     19$                                             ;
                                  07            55   E9 0005B 6$:      BLBC    R5, 7$                                          ; 1784
                                  50       03 A743   7E 0005E          MOVAQ   PAT$GB_OPINFO1+7[R3], R0                        ;
                                                0A   11 00063          BRB     8$                                              ;
                 50               53        F8  8F   78 00065 7$:      ASHL    #-8, R3, R0                                     ;
                                  50       03 A640   7E 0006A          MOVAQ   PAT$GB_OPINFO2+7[R0], R0                        ;
       52        60               04            04   EF 0006F 8$:      EXTZV   #4, #4, (R0), DISP_SIZE                         ;
                                                E2   13 00074          BEQL    5$                                              ;
                                  03            52   D1 00076          CMPL    DISP_SIZE, #3                                   ; 1788
                                                03   12 00079          BNEQ    9$                                              ;
                                  52            04   D0 0007B          MOVL    #4, DISP_SIZE                                   ; 1790
                                  10 00000000'  EF   E9 0007E 9$:      BLBC    MAP_FLAG, 10$                                   ; 1799
                                          4004  8F   BB 00085          PUSHR   #^M<R2,SP>                                      ; 1801
                                            04  AC   DD 00089          PUSHL   STREAM_PTR                                      ;
                      00000000G   EF            03   FB 0008C          CALLS   #3, PAT$GET_VALUE                               ;
                                                0A   11 00093          BRB     11$                                             ;
                 5C               52            03   78 00095 10$:     ASHL    #3, DISP_SIZE, R0                               ; 1803
       6E    04  BC               50            00   EE 00099          EXTV    #0, R0, @STREAM_PTR, STREAM_VALUE               ;
                 54               52            03   78 0009F 11$:     ASHL    #3, DISP_SIZE, R4                               ; 1806
                                  06            55   E9 000A3          BLBC    R5, 12$                                         ;
                                  50          6743   7E 000A6          MOVAQ   PAT$GB_OPINFO1+4[R3], R0                        ; 1804
                                                09   11 000AA          BRB     13$                                             ;
                 50               53        F8  8F   78 000AC 12$:     ASHL    #-8, R3, R0                                     ;
                                  50          6640   7E 000B1          MOVAQ   PAT$GB_OPINFO2+4[R0], R0                        ;
FFFFFFFE   8F    60               04            00   EC 000B5 13$:     CMPV    #0, #4, (R0), #-2                               ;
                                  07            13      000BE          BEQL    14$                                             ;
       51        6E               54            00   EE 000C0          EXTV    #0, R4, STREAM_VALUE, DISPL                     ; 1806
                                                05   11 000C5          BRB     15$                                             ;
       51        6E               54            00   EF 000C7 14$:     EXTZV   #0, R4, STREAM_VALUE, DISPL                     ; 1808
```

```
                              04   AC          52  C0 000CC 15$:     ADDL2   DISP_SIZE, STREAM_PTR                              ; 1809
                              10   BC          52  C0 000D0          ADDL2   DISP_SIZE, @INS_PC                                 ; 1810
              00000000G  EF        51     10   BC  C1 000D4          ADDL3   @INS_PC, DISPL, PAT$GL_LAST_VAL                    ; 1811
                                   06          55  E9 000DD          BLBC    R5, 16$                                            ; 1816
                                   50        6743  7E 000E0          MOVAQ   PAT$GB_OPINFO1+4[R3], R0                           ;
                                               09  11 000E4          BRB     17$                                                ;
                         50        53     F8   8F  78 000E6 16$:     ASHL    #-8, R3, R0                                        ;
                                   50        6640  7E 000EB          MOVAQ   PAT$GB_OPINFO2+4[R0], R0                           ;
FFFFFFFE   8F            60        04          00  EC 000EF 17$:     CMPV    #0, #4, (R0), #-2                                  ;
                                               04  13 000F8          BEQL    18$                                                ;
                                   51     10   BC  C0 000FA          ADDL2   @INS_PC, DISPL                                     ; 1818
                                   7E          04  7D 000FE 18$:     MOVQ    #4, -(SP)                                          ; 1819
                                               51  DD 00101          PUSHL   DISPL                                              ;
                        00000000G  EF          03  FB 00103          CALLS   #3, PAT$OUT_SYM_VAL                                ;
                                   50     04   AC  D0 0010A          MOVL    STREAM_PTR, R0                                     ; 1820
                                                   04 0010E          RET                                                        ;
                                               50  D4 0010F 19$:     CLRL    R0                                                 ; 1821
                                                   04 00111          RET                                                        ;

; Routine Size:  274 bytes,    Routine Base:  _PAT$CODE + 0368
```

```
1 %SBTTL 'PUT_REG - Print a register name'
1 ROUTINE PUT_REG( REG, CS_FLAG ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       THIS ROUTINE TAKES ONE PARAMETER WHICH IT ASSUMES IS
1 !       THE NUMBER OF A VAX REGISTER.  IT THEN PRINTS OUT
1 !       'R' FOLLOWED BY THE NUMBER (IN DECIMAL), UNLESS THE
1 !       REGISTER NUMBER IS 'SPECIAL'.  THE SPECIAL REGISTERS INCLUDE:
1 !
1 !       REGISTER NUMBER         SPECIAL NAME
1 !
1 !               12                  AP
1 !               13                  FP
1 !               14                  SP
1 !               15                  PC
1 !
1 !       An additional parameter is used as a flag to indicate
1 !       whether the register reference should be enclosed in
1 !       round/square brackets or not.
1 !
1 ! INPUTS:
1 !
1 !       REG - The register number.
1 !       CS_FLAG - A flag to control printing before/after REG.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE.
1 !
1 ! OUTPUTS:
1 !
1 !       THE REGISTER REFERENCE IS PRINTED.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE.
1 !--
1
2 BEGIN
2
2 LOCAL
2       INDEX;
2
2 OWN
2       ENCLOSING_CS : VECTOR[4,WORD]                           ! Enclosing strings for REG.
2                      INITIAL( WORD(    %ASCIC '(',
2                                        %ASCIC ')',
  2                                           %ASCIC '[',
  2                                           %ASCIC ']'),
  2         PUT_CS    : VECTOR[4,BYTE]                                ! FAO formatting string.
  2                     INITIAL( %ASCIC '!AC' ),
  2         REGNAMES  : VECTOR[4,WORD]                                ! SPECIAL REGISTER NAMES.
  2                     INITIAL( WORD( 'AP', 'FP', 'SP', 'PC') );
  2
  2 !++
  2 ! IF ANY ENCLOSING STRINGS SHOULD BE OUTPUT, THEN CS_FLAG PROVIDED
  2 ! AN INDEX INTO THE VECTOR OF STRINGS.
  2 !--
  3 IF( (INDEX = .CS_FLAG) NEQ NO_BRACKETS )
  2 THEN
  2         PAT$FAO_PUT( PUT_CS, ENCLOSING_CS[.INDEX] );
  2
  2 !++
  2 ! Now print the actual register reference.
  2 !--
  3 IF( .REG LSS AP_REG )
  2 THEN
  2         PAT$FAO_PUT( UPLIT ( %ASCIC 'R!UB' ), .REG )
  2 ELSE
  2         !++
  2         ! The reserved registers have special names which
  2         ! are extracted from the above vector.
  2         !--
  2         PAT$FAO_PUT( UPLIT ( %ASCIC '!AD' ), 2, REGNAMES[.REG-12] );
  2
  2 !++
  2 ! Check for any enclosing string, right parentheses or bracket.
  2 !--
  3 IF( .INDEX NEQ NO_BRACKETS )
  2 THEN
  2         PAT$FAO_PUT( PUT_CS, ENCLOSING_CS[.INDEX+1] );
  1 END;


                                                            .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

            00  00  00  42  55  21  52  04  00014 P.AAD:   .ASCII  <4>\R!UB\<0><0><0>             ;
                            44  41  21  03  0001C P.AAE:   .ASCII  <3>\!AD\                       ;

                                                            .PSECT  _PAT$OWN,NOEXE,2

                                    28  01  0000C ENCLOSING_CS:
                                                           .ASCII  <1>\(\                         ;
                                    29  01  0000E          .ASCII  <1>\)\                         ;
                                    5B  01  00010          .ASCII  <1>\[\                         ;
                                    5D  01  00012          .ASCII  <1>\]\                         ;
                            43  41  21  03  00014 PUT_CS:  .ASCII  <3>\!AC\                       ;
                                    50  41  00018 REGNAMES:
                                                           .ASCII  \AP\                           ;
                                    50  46  0001A          .ASCII  \FP\                           ;
                                    50  53  0001C          .ASCII  \SP\                           ;
                                    43  50  0001E          .ASCII  \PC\                           ;
```

```
                                                                    .PSECT   _PAT$CODE,NOWRT,2

                                      001C 00000 PUT_REG:.WORD      Save R2,R3,R4                                   ; 1823
                  54 00000000G  EF     9E 00002         MOVAB      PAT$FAO_PUT, R4                                  ;
                  53 00000000'  EF     9E 00009         MOVAB      PUT_CS, R3                                       ;
                  52        08  AC     D0 00010         MOVL       CS_FLAG, INDEX                                   ; 1890
                  01            52     D1 00014         CMPL       INDEX, #1                                        ;
                                09     13 00017         BEQL       1$                                               ;
                          F8 A342      3F 00019         PUSHAW     ENCLOSING_CS[INDEX]                              ; 1892
                                53     DD 0001D         PUSHL      R3                                               ;
                  64            02     FB 0001F         CALLS      #2, PAT$FAO_PUT                                  ;
                  0C        04  AC     D1 00022 1$:     CMPL       REG, #12                                         ; 1897
                                0E     18 00026         BGEQ       2$                                               ;
                            04  AC     DD 00028         PUSHL      REG                                              ; 1899
                     00000000'  EF     9F 0002B         PUSHAB     P.AAD                                            ;
                  64            02     FB 00031         CALLS      #2, PAT$FAO_PUT                                  ;
                                13     11 00034         BRB        3$                                               ;
                  50        04  AC     D0 00036 2$:     MOVL       REG, R0                                          ; 1905
                          EC A340      3F 0003A         PUSHAW     REGNAMES-24[R0]                                  ;
                                02     DD 0003E         PUSHL      #2                                               ;
                     00000000'  EF     9F 00040         PUSHAB     P.AAE                                            ;
                  64            03     FB 00046         CALLS      #3, PAT$FAO_PUT                                  ;
                  01            52     D1 00049 3$:     CMPL       INDEX, #1                                        ; 1910
                                09     13 0004C         BEQL       4$                                               ;
                          FA A342      3F 0004E         PUSHAW     ENCLOSING_CS+2[INDEX]                            ; 1912
                                53     DD 00052         PUSHL      R3                                               ;
                  64            02     FB 00054         CALLS      #2, PAT$FAO_PUT                                  ;
                                       04 00057 4$:     RET                                                         ; 1913

; Routine Size:  88 bytes,    Routine Base:  _PAT$CODE + 047A
```

```
1 %SBTTL 'DISPLACEMENT - Determine size of operand'
1 ROUTINE DISPLACEMENT( STREAM_PTR, FLAG, DISPO, PTR_DISP_SIZE, INDEX, OPCODE, INS_PC ) =
1
1 !++
1 !  FUNCTIONAL DESCRIPTION:
1 !
1 !       DECIDE IF THERE IS A DISPLACEMENT FOR THE CURRENT
1 !       OPERAND OF THE CURRENT INSTRUCTION.  IF THERE IS,
1 !       EXTRACT IT FROM THE INSTRUCTION STREAM AND RETURN AN
1 !       INDICATION OF THE CASE DETECTED.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       DISPLACEMENT ();
1 !
1 ! INPUTS:
1 !
1 !       STREAM_PTR      - POINTER TO THE BEGINNING OF THE CURRENT OPERAND SPECIFIER.
1 !       FLAG            - POINTER TO THE RETURN LOCATION FOR ONE OF THE 3 FLAGS
1 !                         IF THERE IS A DISPLACEMENT OR LITERAL ASSOCIATED
1 !                         WITH THIS OPERAND REFERENCE.
1 !       DISPO           - A POINTER TO THE RETURN BUFFER FOR THE ACTUAL
1 !                         DISPLACEMENT OR LITERAL.
1 !       PTR_DISP_SIZE   - ADDRESS TO CONTAIN RETURNED VALUE OF NUMBER
1 !                         OF BYTES ACTUALLY NEEDED FOR THE DISPLACEMENT.
1 !                         This is done strictly for the benefit of FAO,
1 !                         which would fill out output fields with 0s
1 !                         otherwise, giving misleading output.
1 !       INDEX           - THE ORDINAL OF THE OPERAND BEING DECODED.
1 !       OPCODE          -  THE OPCODE OF THE INSTRUCTION BEING DECODED.
1 !                        (This parameter has already been validated.)
1 !       INS_PC          - THE PC FOR WHICH THE INSTRUCTION WAS ENCODED.
1 !
1 ! OUTPUTS:
1 !
1 !       1) A VALUE OF -1, 0, OR 1 IS RETURNED VIA THE LONGWORD
1 !          POINTER, FLAG.  0 IS RETURNED IF NO DISPLACEMENT IS TO BE
1 !          ASSOCIATED WITH THIS OPERAND REFERENCE.  OTHERWISE 1 OR -1
1 !          IS RETURNED TO SEPARATE THE CASES LISTED BELOW.  THIS PROVIDES A WAY
1 !          TO PRINT '#' BEFORE SOME LITERALS, (E.G. MOVL #01,R0), AND TO INDICATE
1 !          WHEN TO PRINT '(RN)' AFTER THE DISPLACEMENT, ETC.  THIS INFORMATION
1 !          IS RETURNED TO PRECLUDE TESTING FOR IT AGAIN.
1 !
1 !       2) IF THERE IS A DISPLACEMENT, ITS VALUE IS RETURNED
1 !          TO THE BUFFER POINTED AT BY 'DISPO'.  IF THE FLAG WHICH IS
1 !          RETURNED (SEE ABOVE) IS 0, THE BUFFER POINTED
1 !          TO BY 'DISPO' IS UNCHANGED.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       -THE (NEW) BYTE STREAM POINTER WHICH WILL THEN POINT TO THE
1 !        BEGINNING OF THE NEXT INSTRUCTION OR OPERAND REFERENCE.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE.
1 !--
```

```
1
2 BEGIN
2
2 MAP
2         INS_PC          :       REF VECTOR[,LONG],
2         PTR_DISP_SIZE   :       REF VECTOR[,LONG],
2         DISPO           :       REF VECTOR[,BYTE],
2         FLAG            :       REF VECTOR[,LONG],
2         STREAM_PTR      :       REF BLOCK[,BYTE];
2
2 LOCAL
2         MAP_STREAM_PTR : REF BLOCK[,BYTE],                        ! MAPPED ADDRESS OF BYTE STREAM
2         STREAM_VALUE : BLOCK[4,BYTE],                             ! VALUES FROM BYTE STREAM
2         MODE,                                                     ! DOMINANT ADDRESSING MODE
2         F,                                                        ! FLAG VALUE RETURNED
2         DISP_SIZE;                                                ! SIZE, IN BYTES, OF THE DISPLACEMENT
2
2 !++
2 !  ASSUME THERE IS NO DISPLACMENT, BUT THEN CHECK FOR THE CASES:
2 !
2 !       1) LITERAL MODE - DOMINANT MODE IS 0, 1, 2, OR 3.
2 !
2 !       2) BYTE, WORD, OR LONGWORD, DISPLACEMENT OR DEFERRED
2 !          DISPLACEMENT MODES.
2 !
2 !       3) MODE 8 WHEN REG IS PC (IE #CONST, INSTEAD OF (PC)+ )
2 !         OR
2 !          MODE 9 WHEN REG IS PC (IE @#ADDRESS, INSTEAD OF @(PC)+ )
2 !
2 !  CASES 1 AND 3 ARE TYPE 1, WHILE CASE 2 IS TYPE -1.
2 !--
2 F = 0;
2 DISP_SIZE = 0;
2 IF .MAP_FLAG                                                      ! IS INSTRUCTION AT PC?
2 THEN
2         PAT$GET_VALUE(.STREAM_PTR, A_BYTE, STREAM_VALUE)          ! YES, MAP ADDRESS
2 ELSE
2         STREAM_VALUE = .STREAM_PTR[0, 0, (A_BYTE * BITS_PER_BYTE), 0]; ! NO, GET VALUE FROM BUFFER
3 IF( (MODE = .STREAM_VALUE[ DSP[_MODE ]) LSS INDEXING_MODE )
2 THEN
3         BEGIN
3         !++
3         ! CASE 1: LITERAL MODE ADDRESSING
3         !         THE LITERAL IS A 6-BIT FIELD WHICH MUST BE EXTRACTED WITHOUT
3         !         SIGN-EXTENSION FROM THE ADDRESSING MODE BYTE.  EXTRACT THIS
3         !         FIELD, PASS IT BACK, AND SET A FLAG TO INDICATE WHICH CASE
3         !         WAS FOUND.  NOTE THAT THE FLAG VALUE BEING 1 ALSO MEANS THAT
3         !         IT CAN BE USED FOR THE DISP_SIZE VALUE RETURNED.
3         !--
3         DISPO[0] = .STREAM_VALUE[SHORT_LITERAL];
3         F = 1;
3         END
2 ELSE
3         IF( .MODE GTR AT_PC_REL_MODE )
2         THEN
3                 BEGIN
3                 !++
3                       ! CASE 2: DISPLACEMENT OR DEFERRED DISPLACEMENT MODE.  THERE
3                       !         ONLY REMAINS TO DECIDE HOW MUCH OF THE BYTE STREAM TO
3                       !         EXTRACT.  THIS IS DONE SIMPLY BY LOOKING AT WHAT THE
3                       !         'MODE' FIELD OF THE DOMINANT BYTE IS.  TO
3                       !         DIFFERENTIATE HEX A OR B, C OR D, AND E OR F,
3                       !         SO JUST LOOK FOR THE MOST CONVENIENT BITS TO CHECK.
3                       !         START BY ASSUMING BYTE DISPLACEMENT (HEX A OR B), THEN
3                       !         SORT OUT THE OTHER TWO CASES.
3                       !--
3                       F = -1;
3                       DISP_SIZE = A_BYTE;
4                       IF( .STREAM_VALUE[ DOM_MOD_FIELD ] LSS 0 )
3                       THEN
5                               DISP_SIZE = (IF (.STREAM_VALUE[DOM_MOD_FIELD])
4                                            THEN A_LONGWORD            ! MODE IS HEX E OR F.
3                                            ELSE A_WORD);              ! MODE IS HEX C OR D.
3                       END
2               ELSE
3                       IF ((.STREAM_VALUE[OPERAND_VALUE] EQL PC_REG) AND
3                          (.MODE EQL PC_REL_MODE OR .MODE EQL AT_PC_REL_MODE))
2                       THEN
3                               BEGIN
3                               !++
3                               ! CASE 3: SPECIAL CASE NOTATION FOR PC MODES.
3                               !         THE ONLY DIFFICULTY IS DECIDING HOW
3                               !         MUCH OF THE BYTE STREAM TO 'EAT UP'.
3                               !         @#ADDRESS ALWAYS HAS LONGWORD CONTEXT,
3                               !         WHILE THE CONTEXT OF #CONST DEPENDS ON
3                               !         THE OPCODE AND THE OPERAND ORDINAL.
3                               F = 1;
4                               IF( .MODE EQL AT_PC_REL_MODE )
3                               THEN
3                                       DISP_SIZE = A_LONGWORD
3                               ELSE
3                                       DISP_SIZE = INS_CONTEXT( .INDEX, .OPCODE );
2                               END;
2       !++
2       ! AT THIS POINT .DISP_SIZE IS THE NUMBER OF BYTES PAST THE MODE BYTE
2       ! CONSUMED FROM THE INSTRUCTION STREAM, IF ANY.  JUST GO AHEAD AND CONSUME THESE
2       ! BYTES, PASS BACK THE DISPLACEMENT, AND RETURN THE NEW INSTRUCTION-STREAM BYTE
2       ! POINTER.  EVEN IF THERE IS NO DISPLACEMENT, AT LEAST THE DOMINANT ADDRESSING
2       ! MODE BYTE WAS CONSUMED.  THE NEXT COMMAND INCREMENTS THE STREAM_PTR
2       ! ADDRESS BY ONE.
2       !--
2       STREAM_PTR = STREAM_PTR[NEXT_FIELD(1)];
2       INS_PC[0] = .INS_PC[0] + 1;
2
2       !++
2       ! Pass back the flag now so that F can be re-used as a temporary, below.
2       !--
2       FLAG[0] = .F;
3       IF( .DISP_SIZE NEQ 0 )
2       THEN
3               BEGIN
3               F = .DISP_SIZE;
3
3               !++
; 1047          2085  3             ! PASS BACK THE LITERAL OR DISPLACEMENT.  NOTE THE SIGN EXTENSION.
; 1048          2086  3             !--
; 1049          2087  3             IF .MAP_FLAG                                            ! IS INSTRUCTION AT PC?
; 1050          2088  3             THEN
; 1051          2089  3                     PAT$GET_VALUE(.STREAM_PTR, .DISP_SIZE, .DISPO)  . YES, MAP ADDRESS
; 1052          2090  3             ELSE
; 1053          2091  3                     INCR I FROM 0 TO .DISP_SIZE - 1 DO              ! NO GET VALUE FROM BUFFER
; 1054          2092  3                             DISPO[.I] = .STREAM_PTR[ .I, 0, BITS_PER_BYTE, 0];
; 1055          2093  2             END;
; 1056          2094  2
; 1057          2095  2 !++
; 1058          2096  2 ! Pass back an indication of the number of bytes actually consumed for the
; 1059          2097  2 ! displacement or literal for the benefit of FAO output.
; 1060          2098  2 !--
; 1061          2099  2 PTR_DISP_SIZE[0] = .F;
; 1062          2100  2 INS_PC[0] = .INS_PC[0] + .DISP_SIZE;
; 1063          2101  2 RETURN( STREAM_PTR[ NEXT_FIELD( .DISP_SIZE ) ] );
; 1064          2102  1 END;



                                               003C 00000 DISPLACEMENT:
                                                                 .WORD   Save R2,R3,R4,R5                                   ; 1915
                          55 00000000'  EF  9E 00002             MOVAB   MAP_FLAG, R5                                       ;
                          54 00000000G  EF  9E 00009             MOVAB   PAT$GET_VALUE, R4                                  ;
                          5E            04  C2 00010             SUBL2   #4, SP                                             ;
                                        52  7C 00013             CLRQ    DISP_SIZE                                          ; 2003
                          0C            65  E9 00015             BLBC    MAP_FLAG, 1$                                       ; 2004
                                        5E  DD 00018             PUSHL   SP                                                 ; 2006
                                        01  DD 0001A             PUSHL   #1                                                 ;
                                   04   AC  DD 0001C             PUSHL   STREAM_PTR                                         ;
                          64            03  FB 0001F             CALLS   #3, PAT$GET_VALUE                                  ;
                                        04  11 00022             BRB     2$                                                 ;
                          6E       04   BC  9A 00024 1$:         MOVZBL  @STREAM_PTR, STREAM_VALUE                          ; 2008
     50          6E       04            04  EF 00028 2$:         EXTZV   #4, #4, STREAM_VALUE, MODE                         ; 2009
                          04            50  D1 0002D             CMPL    MODE, #4                                           ;
                          0E            18 00030                 BGEQ    3$                                                 ;
     51          6E       06            00  EF 00032             EXTZV   #0, #6, STREAM_VALUE, R1                           ; 2020
                     0C   BC            51  90 00037             MOVB    R1, @DISPO                                         ;
                          53            01  D0 0003B             MOVL    #1, F                                              ; 2021
                                        47  11 0003E             BRB     8$                                                 ; 2009
                          09            50  D1 00040 3$:         CMPL    MODE, #9                                           ; 2024
                                        16  15 00043             BLEQ    4$                                                 ;
                          53            01  CE 00045             MNEGL   #1, F                                              ; 2037
                          52            01  D0 00048             MOVL    #1, DISP_SIZE                                      ; 2038
     00          6E       02            05  EC 0004B             CMPV    #5, #2, STREAM_VALUE, #0                           ; 2039
                                        35  18 00050             BGEQ    8$                                                 ;
                 1E       6E            05  E0 00052             BBS     #5, STREAM_VALUE, 6$                               ; 2041
                          52            02  D0 00056             MOVL    #2, DISP_SIZE                                      ;
                                        2C  11 00059             BRB     8$                                                 ;
     0F          6E       04            00  ED 0005B 4$:         CMPZV   #0, #4, STREAM_VALUE, #15                          ; 2046
                                        25  12 00060             BNEQ    8$                                                 ;
                          08            50  D1 00062             CMPL    MODE, #8                                           ; 2047
                                        05  13 00065             BEQL    5$                                                 ;
                          09            50  D1 00067             CMPL    MODE, #9                                           ;
```

```
                                 1B  12 0006A        BNEQ    8$                                  ;
                      53         01  D0 0006C 5$:    MOVL    #1, F                               ; 2057
                      09         50  D1 0006F        CMPL    MODE, #9                            ; 2058
                                 05  12 00072        BNEQ    7$                                  ;
                      52         04  D0 00074 6$:    MOVL    #4, DISP_SIZE                       ; 2060
                                 0E  11 00077        BRB     8$                                  ;
                      7E     14  AC  7D 00079 7$:    MOVQ    INDEX, -(SP)                        ; 2062
            00000000V EF         02  FB 0007D        CALLS   #2, INS_CONTEXT                     ;
                      52         50  D0 00084        MOVL    R0, DISP_SIZE                       ;
                             04  AC  D6 00087 8$:    INCL    STREAM_PTR                          ; 2072
                             1C  BC  D6 0008A        INCL    @INS_PC                             ; 2073
                08    BC         53  D0 0008D        MOVL    F, @FLAG                            ; 2078
                                 52  D5 00091        TSTL    DISP_SIZE                           ; 2079
                                 23  13 00093        BEQL    12$                                 ;
                      53         52  D0 00095        MOVL    DISP_SIZE, F                        ; 2082
                      0D         65  E9 00098        BLBC    MAP_FLAG, 9$                        ; 2087
                             0C  AC  DD 0009B        PUSHL   DISP0                               ; 2089
                                 52  DD 0009E        PUSHL   DISP_SIZE                           ;
                             04  AC  DD 000A0        PUSHL   STREAM_PTR                          ;
                      64         03  FB 000A3        CALLS   #3, PAT$GET_VALUE                   ;
                                 10  11 000A6        BRB     12$                                 ;
                      50         01  CE 000A8 9$:    MNEGL   #1, I                               ; 2091
                                 07  11 000AB        BRB     11$                                 ;
                0C BC40       04 BC40  90 000AD 10$:   MOVB    @STREAM_PTR[I], @DISP0[I]           ; 2092
        F5            50         52  F2 000B4 11$:   AOBLSS  DISP_SIZE, I, 10$                   ;
                10    BC         53  D0 000B8 12$:   MOVL    F, @PTR_DISP_SIZE                   ; 2099
                1C    BC         52  C0 000BC        ADDL2   DISP_SIZE, @INS_PC                  ; 2100
        50            52     04  AC  C1 000C0        ADDL3   STREAM_PTR, DISP_SIZE, R0           ; 2101
                                     04 000C5        RET                                         ; 2102

; Routine Size:  198 bytes,    Routine Base:  _PAT$CODE + 04D2
```

```
%SBTTL 'INS_CONTEXT - Determine operand type'
ROUTINE INS_CONTEXT( INDEX, OPCODE ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       THIS ROUTINE DECIDES WHAT CONTEXT APPLIES TO THE GIVEN
!       OPERAND FOR A SPECIFIC OPCODE.  IT IS USED TO DETERMINE
!       WHETHER A PC-RELATIVE MODE FOR THIS OPERAND WOULD
!       REQUIRE A BYTE, WORD, LONGWORD, OR QUADWORD OPERAND.
!
! CALLING SEQUENCE:
!
!       INS_CONTEXT ();
!
! INPUTS:
!
!       INDEX - OPERAND IS BEING DECODED.  THIS NUMBER
!               MUST BE 1, 2, ... 6.
!       OPCODE  -THE OPCODE OF THE INSTRUCTION BEING DECODED.
!                (This parameter has already been validated.)
!
! IMPLICIT INPUTS:
!
!       None.
!
! OUTPUTS:
!
!       NONE.
!
! IMPLICIT OUTPUTS:
!
!       NONE.
!
! ROUTINE VALUE:
!
!       IF SOME ERROR IS DETECTED, RETURN FALSE.  OTHERWISE RETURN
!       THE NUMBER OF BYTES FROM THE INSTRUCTION STREAM THAT THE CURRENT
!       OPERAND REFERENCE SHOULD CONSUME.  THIS NUMBER WILL BE:
!
!     NUMBER     OP_CONTEXT       NAME FROM OPI MACRO DEFINITION
!   OF BYTES          VALUE
!
!          1              0                BYT
!          2              1                WRD
!          4              2                LNG
!          8              3                QAD
!         16              4                OCT
!
!   THE VALUE, 0 TO 4, STORED IN THE OP_CONTEXT FIELD IS THE POWER OF TWO
!   WHICH WILL YIELD THE 'NUMBER OF BYTES' ENTRY, ABOVE.
!
! SIDE EFFECTS:
!
!       NONE.
!--

```

```
; 1123          2160  2 BEGIN
; 1124          2161  2 !++
; 1125          2162  2 ! CHECK FOR ANY OF THE FOLLOWING ERROR CONDITIONS:
; 1126          2163  2 !       1) INSUFFICIENT INFORMATION ABOUT IT.
; 1127          2164  2 !          (IE - IT IS RESERVED OR YET TO BE DEFINED).
; 1128          2165  2 !       2)  CONFLICTING INFORMATION ABOUT NUMBER OF OPERANDS FOR OPCODE.
; 1129          2166  2 !          THIS CHECK IS NECESSARY BECAUSE THE 'NUL' ENTRY IN THE OPINFO
; 1130          2167  2 !          MACROS RESULTS IN THE SAME VALUE BEING ENCODED AS THE 'BYT' ONES DO.
; 1131          2168  2 !          THIS ERROR CAN BE CAUGHT AT THIS POINT (BY LOOKING AT THE OP_NUMOPS ENTRY FOR
; 1132          2169  2 !          THIS OPCODE), IT DID NOT SEEM WORTH TAKING UP MORE BITS IN THE OPINFO
; 1133          2170  2 !          TABLE TO DIFFERENTIATE 'NUL' AND THE OTHERS.
; 1134          2171  2 !--
; 1135          2172  3 IF (.PAT$GB_OPINFO[.OPCODE, OP_NUMOPS] EQL NOT_AN_OP)          ! ERROR 1, SEE ABOVE.
; 1136          2173  2 THEN
; 1137          2174  2         RETURN(FALSE);
; 1138          2175  3 IF ((.PAT$GB_OPINFO[.OPCODE, OP_NUMOPS] LSS .INDEX) OR
; 1139          2176  3    (.INDEX LEQ 0))                                              ! ERROR 2, SEE ABOVE.
; 1140          2177  2 THEN
; 1141          2178  2         RETURN(FALSE);
; 1142          2179  2
; 1143          2180  2 !++
; 1144          2181  2 ! NOW IT IS JUST A MATTER OF LOOKING INTO OUR OPINFO TABLE TO REQUIRE 0 THROUGH
; 1145          2182  2 ! 4.  THIS JUST HAPPENS TO BE THE POWER OF 2 WHICH NEEDED TO CALCULATE THE
; 1146          2183  2 ! NUMBER OF BYTES OCCUPIED BY THE CORRESPONDING OPERAND.
; 1147          2184  2 !--
; 1148          2185  2 RETURN (( 1 ^ .PAT$GB_OPINFO[.OPCODE, OP_CONTEXT(.INDEX)]));
; 1149          2186  1 END;



                                                  001C 00000 INS_CONTEXT:
                                                                    .WORD   Save R2,R3,R4                          ; 2104
                        54 00000000G  EF  9E 00002                 MOVAB   PAT$GB_OPINFO1+4, R4                   ;
                        53 00000000G  EF  9E 00009                 MOVAB   PAT$GB_OPINFO2+4, R3                   ;
                        51         08  AC  D0 00010                 MOVL    OPCODE, R1                             ; 2172
                                       52  D4 00014                 CLRL    R2                                     ;
                   FD   8F             51  91 00016                 CMPB    R1, #253                               ;
                                       08  13 0001A                 BEQL    1$                                     ;
                                       52  D6 0001C                 INCL    R2                                     ;
                        50           6441  7E 0001E                 MOVAQ   PAT$GB_OPINFO1+4[R1], R0               ;
                                       09  11 00022                 BRB     2$                                     ;
              50        51      F8     8F  78 00024 1$:             ASHL    #-8, R1, R0                            ;
                        50           6340  7E 00029                 MOVAQ   PAT$GB_OPINFO2+4[R0], R0               ;
FFFFFFFF   8F         60        04     00  EC 0002D 2$:             CMPV    #0, #4, (R0), #-1                      ;
                                       40  13 00036                 BEQL    7$                                     ;
                        06             52  E9 00038                 BLBC    R2, 3$                                 ; 2175
                        50           6441  7E 0003B                 MOVAQ   PAT$GB_OPINFO1+4[R1], R0               ;
                                       09  11 0003F                 BRB     4$                                     ;
              50        51      F8     8F  78 00041 3$:             ASHL    #-8, R1, R0                            ;
                        50           6340  7E 00046                 MOVAQ   PAT$GB_OPINFO2+4[R0], R0               ;
      04   AC         60        04     00  EC 0004A 4$:             CMPV    #0, #4, (R0), INDEX                    ;
                                       26  19 00050                 BLSS    7$                                     ;
                                04     AC  D5 00052                 TSTL    INDEX                                  ; 2176
                                       21  15 00055                 BLEQ    7$                                     ;
                        06             52  E9 00057                 BLBC    R2, 5$                                 ; 2185
```

```
                          50            6441  7E 0005A          MOVAQ   PAT$GB_OPINFO1+4[R1], R0                          ;
                                          09  11 0005E          BRB     6$                                                ;
                    50          51    F8  8F  78 00060 5$:      ASHL    #-8, R1, R0                                       ;
                                50      6340  7E 00065          MOVAQ   PAT$GB_OPINFO2+4[R0], R0                          ;
                    52     04   AC        02  78 00069 6$:      ASHL    #2, INDEX, R2                                     ;
        51          60          04        52  EF 0006E          EXTZV   R2, #4, (R0), R1                                  ;
                    50          01        51  78 00073          ASHL    R1, #1, R0                                        ;
                                              04 00077          RET                                                       ;
                                          50  D4 00078 7$:      CLRL    R0                                                ; 2186
                                              04 0007A          RET                                                       ;

; Routine Size:  123 bytes,    Routine Base:  _PAT$CODE + 0598
```

```
1 ROUTINE CHK_ASD_TBL( INS_PC, ASD_ENTRY_PTR, ASM_DIR_TBL ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 ! This routine determines if the PC to be decoded is a known assembler
1 ! directive.  If there is no assembler directive correlation table or the flag
1 ! specifies not to check it, then FALSE is returned.  Otherwise, the table is
1 ! searched to see if it contains the given PC.  If not FALSE is returned.  If
1 ! the PC is within the table, then the offset into the OPINFO table to the
1 ! directive is returned.  Also, in this case, a pointer into the assembler
1 ! directive table is returned.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CHK_ASD_TBL( INS_PC, ASD_ENTRY_PTR, ASM_DIR_TBL )
1 !
1 ! INPUTS:
1 !
1 !       INS_PC - the PC to search the ASD table for
1 !       ASD_ENTRY_PTR - location to place ASD table pointer if found
1 !       ASM_DIR_TBL - flag indicating whether or not to search ASD table,
1 !                       FALSE = do not search, TRUE = search
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The ASD table must have been initialized.
1 !
1 ! OUTPUTS:
1 !
1 !       The pointer into the ASD table is set to 0 or the appropriate entry.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       The returned value is either:
1 !
1 !               FALSE, if the PC is not in the table or there was no ASD table.
1 !               otherwise, it is the "OPCODE" offset into the OPINFO table.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE.
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       ASM_DIR_TBL : REF BLOCK[,BYTE],                         ! Descriptor for assembler directive table
2       ASD_ENTRY_PTR : REF VECTOR[,LONG];                      ! Pointer to set if PC is found in ASD table
2
2 LOCAL
2       OPINFO_PTR : REF BLOCK[,BYTE],                          ! Local pointer into OPINFO table from ASD e
```

```
2         ASD_SIZE,                                          ! Remaining length of ASD table to search
2         ASD_PTR : REF BLOCK[,BYTE];                        ! Local pointer into ASD table for search
2
2 !++
2 ! Check if an ASD table was built.  If not, this means that there was
2 ! no assembler directive in the instructions just deposited.
2 !--
3 IF (.ASM_DIR_TBL EQL 0)
2 THEN
2         RETURN(FALSE);
3 IF ((ASD_PTR = .ASM_DIR_TBL[DSC$A_POINTER]) EQL 0)
2 THEN
2         RETURN(FALSE);
2
2 !++
2 ! Loop, searching the assembler directive table, ASD, for the PC provided.
2 ! If it is located, then set the ASD_ENTRY_PTR pointer and return the opcode
2 ! offset into the OPINFO table.  If this routine falls out of the loop, then
2 ! the PC was not in the table and FALSE is returned.
2 !--
2 ASD_SIZE = .ASM_DIR_TBL[DSC$W_LENGTH];
2 ASD_ENTRY_PTR[0] = 0;
2 WHILE .ASD_SIZE GEQ ASD$C_SIZE
2 DO
3         BEGIN
4         IF (.INS_PC EQL .ASD_PTR[ASD$L_PC])
3         THEN
4                 BEGIN
4                 !++
4                 ! PC was found in the ASD table.  Set up return values.
4                 !--
4                 ASD_ENTRY_PTR[0] = .ASD_PTR;
4                 OPINFO_PTR = .ASD_PTR[ASD$L_OPINFO];
6                 RETURN (  IF (.OPINFO_PTR[OP_BR_TYPE] EQL BR_BY)
6                           THEN  (SIZOPINFO1 - 3)
6                           ELSE  IF (.OPINFO_PTR[OP_BR_TYPE] EQL BR_WD)
6                                 THEN (SIZOPINFO1 - 2)
4                                 ELSE (SIZOPINFO1 - 1));
4                 END
3         ELSE
4                 BEGIN
4                 !++
4                 ! PC is not equal to this entry.  Update to next entry in table.
4                 !--
4                 ASD_SIZE = .ASD_SIZE - ASD$C_SIZE;
4                 ASD_PTR = CH$PTR (.ASD_PTR, ASD$C_SIZE);
3                 END;
2         END;
2
2 !++
2 ! PC was not in the table.  Return FALSE (indicating not found) .
2 !--
2 RETURN(FALSE);
1 END;
```

```
                                                      000C 00000 CHK_ASD_TBL:
                                                                         .WORD    Save R2,R3                                  ; 2187
                                   50       0C    AC   D0 00002          MOVL     ASM_DIR_TBL, R0                             ; 2251
                                                  49   13 00006          BEQL     5$                                          ;
                                   51       04    A0   D0 00008          MOVL     4(R0), ASD_PTR                              ; 2254
                                                  43   13 0000C          BEQL     5$                                          ;
                                   53             60   3C 0000E          MOVZWL   (R0), ASD_SIZE                              ; 2264
                                            08    BC   D4 00011          CLRL     @ASD_ENTRY_PTR                              ; 2265
                                   09             53   D1 00014 1$:      CMPL     ASD_SIZE, #9                                ; 2266
                                                  38   19 00017          BLSS     5$                                          ;
                                   61       04    AC   D1 00019          CMPL     INS_PC, (ASD_PTR)                           ; 2269
                                                  2A   12 0001D          BNEQ     4$                                          ;
                              08   BC             51   D0 0001F          MOVL     ASD_PTR, @ASD_ENTRY_PTR                     ; 2275
                                   52       04    A1   D0 00023          MOVL     4(ASD_PTR), OPINFO_PTR                      ; 2276
        01         07   A2         04             04   ED 00027          CMPZV    #4, #4, 7(OPINFO_PTR), #1                   ; 2277
                                                  06   12 0002D          BNEQ     2$                                          ;
                                   50     0100    8F   3C 0002F          MOVZWL   #256, R0                                    ; 2278
                                                  04   00034             RET                                                  ;
        02         07   A2         04             04   ED 00035 2$:      CMPZV    #4, #4, 7(OPINFO_PTR), #2                   ; 2279
                                                  06   12 0003B          BNEQ     3$                                          ;
                                   50     0101    8F   3C 0003D          MOVZWL   #257, R0                                    ; 2280
                                                  04   00042             RET                                                  ;
                                   50     0102    8F   3C 00043 3$:      MOVZWL   #258, R0                                    ; 2281
                                                  04   00048             RET                                                  ; 2277
                                   53             09   C2 00049 4$:      SUBL2    #9, ASD_SIZE                                ; 2288
                                   51             09   C0 0004C          ADDL2    #9, ASD_PTR                                 ; 2289
                                                  C3   11 0004F          BRB      1$                                          ; 2266
                                                  50   D4 00051 5$:      CLRL     R0                                          ; 2297
                                                  04   00053             RET                                                  ;

; Routine Size:  84 bytes,    Routine Base:  _PAT$CODE + 0613
; 1263          2298  1 END
; 1264          2299  0 ELUDOM



;                                        PSECT SUMMARY
;
;       Name                       Bytes                           Attributes
;
;   _PAT$OWN                          32  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _PAT$PLIT                         32  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;   _PAT$CODE                       1639  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   .  ABS  .                          0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)



;                                Library Statistics
;
;                                               -------- Symbols --------      Pages       Processing
;       File                                    Total    Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1            9776         2         0        581         00:01.0



;                                        COMMAND QUALIFIEFS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATMAC/OBJ=OBJ$:PATMAC MSRC$:PATMAC/UPDATE=(ENH$:PATMAC)

; Size:          1639 code + 64 data bytes
; Run Time:         00:33.6
; Elapsed Time:     01:59.6
; Lines/CPU Min:     4106
; Lexemes/CPU-Min: 21354
; Memory Used:  193 pages
; Compilation Complete
```

0302 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PATPAR
LIS

PATMAC
LIS

PATMAI
LIS

PATLST
LIS

PATIO
LIS

PATLEX
LIS

PATMOD
LIS

PATMSG
LIS