```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG    GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG          GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
```

FILEID**DBGPERMOP

```
DDDDDDDD    BBBBBBBB     GGGGGGGG  PPPPPPPP   EEEEEEEEEE  RRRRRRRR    MM      MM    OOOOOO    PPPPPPPP
DDDDDDDD    BBBBBBBB     GGGGGGGG  PPPPPPPP   EEEEEEEEEE  RRRRRRRR    MM      MM    OOOOOO    PPPPPPPP
DD      DD  BB      BB  GG        PP      PP  EE          RR      RR  MMMM  MMMM  OO      OO  PP      PP
DD      DD  BB      BB  GG        PP      PP  EE          RR      RR  MMMM  MMMM  OO      OO  PP      PP
DD      DD  BB      BB  GG        PP      PP  EE          RR      RR  MM  MM  MM  OO      OO  PP      PP
DD      DD  BB      BB  GG        PP      PP  EE          RR      RR  MM  MM  MM  OO      OO  PP      PP
DD      DD  BBBBBBBB    GG        PPPPPPPP   EEEEEEEE    RRRRRRRR    MM      MM  OO      OO  PPPPPPPP
DD      DD  BBBBBBBB    GG        PPPPPPPP   EEEEEEEE    RRRRRRRR    MM      MM  OO      OO  PPPPPPPP
DD      DD  BB      BB  GG  GGGGGG  PP        EE          RR  RR      MM      MM  OO      OO  PP
DD      DD  BB      BB  GG  GGGGGG  PP        EE          RR  RR      MM      MM  OO      OO  PP
DD      DD  BB      BB  GG      GG  PP        EE          RR    RR    MM      MM  OO      OO  PP
DD      DD  BB      BB  GG      GG  PP        EE          RR    RR    MM      MM  OO      OO  PP
DDDDDDDD    BBBBBBBB     GGGGGG    PP        EEEEEEEEEE  RR      RR  MM      MM    OOOOOO    PP
DDDDDDDD    BBBBBBBB     GGGGGG    PP        EEEEEEEEEE  RR      RR  MM      MM    OOOOOO    PP
```

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

Table of contents

```
0000     1          .TITLE  DBGPERMOP
0000     2          .IDENT  /V04-000/
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ; WRITTEN BY
0000    28 ;       Ping Sager      Jun, 1982
0000    29 ;
0000    30 ; MODIFIED BY
0000    31 ;       Rich Title      Nov, 1982       Changed DBG$PERFORM_OPERATOR so
0000    32 ;                                       that it handles value descriptors
0000    33 ;                                       instead of VMS descriptors. This
0000    34 ;                                       is necessary for operations that
0000    35 ;                                       may use the TYPEID field, such
0000    36 ;                                       as the SIZEOF operator in C, where
0000    37 ;                                       it is necessary to look up the DST
0000    38 ;                                       information in order to determine
0000    39 ;                                       the result.
0000    40 ;       V. Holt         Mar, 1983       Added support for packed decimal.
0000    41 ;                                       Added support for PL/I bit-strings.
0000    42 ;
0000    43 ;       P. Sager        Jun, 1983       Fixed up PLI/I bit-string operations
0000    44 ;                                       and packed decimal comparisons.
0000    45 ;       P. Sager        Sep, 1983       Changed .LONG to .BLKL for operand
0000    46 ;                                       and work_octa.
0000    47 ;       K. Glossop      Oct, 1983       Fixed complex exponentiation.  (The
0000    48 ;                                       real and imaginary portions were being
0000    49 ;                                       pushed on the stack in reverse order.)
0000    50 ;                                       Also, a complex raised to a complex
0000    51 ;                                       power checks for (0,0) as the base
0000    52 ;                                       and signals the appropriate error rather
0000    53 ;                                       than causing an internal Debug coding
0000    54 ;                                       error.
0000    55 ;       B. Becker       Oct, 1983       Inserted calls to dbg$succ_enum and
0000    56 ;                                       dbg$pred_enum to implement those built-
0000    57 ;                                       in functions.
```

```
0000     58 ;       B. Becker       Dec, 1983       Inserted calls to fixed binary
0000     59 ;                                       arithmetic routines.
0000     60 ;       B. Becker       Dec, 1983       Added routines DBG$CVT_TUQUADWORD_TO_VALUE
0000     61 ;                                       and DBG$CVT_TUOCTAWORD_TO_VALUE to convert
0000     62 ;                                       to Unsigned Quad & Octawords.  To fixed the
0000     63 ;                                       bug DEP/QUAD I = 8000000000000000 for exampl
0000     64 ;       B. Becker       Jan, 1984       Made the MOD operation work correctly.
0000     65 ;       K. Glossop      Jan, 1984       Fixed G-Float complex add.
0000     66 ;
0000     67 ; MODULE FUNCTION
0000     68 ;       This module contains DBG$PERFORM_OPERATOR, the routine that
0000     69 ;       performs arithmetic operations.
0000     70 ;
0000     71 ; IMPORTANT NOTE
0000     72 ;       This module contains many definitions of constants and
0000     73 ;       offsets that are duplicates of those in DBGLIB.
0000     74 ;       This includes:
0000     75 ;       Case labels (e.g., ADD_L_L, ADD_B_B, etc.) - the order of these
0000     76 ;                       MUST be the same as the ORT$K_ADD_L_L etc. definitions
0000     77 ;                       in DBGLIB, or else the CASE statement will not work.
0000     78 ;       Offsets to fields within Value Descriptors (e.g., DBG$A_VALUE_VMSDESC)
0000     79 ;                       If the fields of a value descriptor change then these
0000     80 ;                       offsets must change accordingly.
0000     81 ;       Literal constants - there are 4 of these: ORT$K_MIN_ROUT,
0000     82 ;                       ORT$K_MAX_ROUT, DBG$K_BASIC, DBG$K_PLI_UBIT (defined
0000     83 ;                       below). These values must correspond to the value
0000     84 ;                       for the same constant in DBGLIB.
0000     85 ;       There are more than just these four needed. see below.
0000     86 ;
0000     87
```

```
0000     89 ;
0000     90 ; External routines
0000     91          .SBTTL  declarations
0000     92          .EXTRN  DBG$ABS_FIXED
0000     93          .EXTRN  DBG$ADD_FIXED_FIXED
0000     94          .EXTRN  DBG$BLISS_INDIRECTION
0000     95          .EXTRN  DBG$BLISS_BITSELECT
0000     96          .EXTRN  DBG$C_ADDRESS_OF
0000     97          .EXTRN  DBG$C_INDIRECTION
0000     98          .EXTRN  DBG$C_SIZEOF
0000     99          .EXTRN  DBG$C_ADD_TPTR_L
0000    100          .EXTRN  DBG$C_SUB_TPTR_L
0000    101          .EXTRN  DBG$C_SUB_TPTR_TPTR
0000    102          .EXTRN  DBG$DIV_FIXED_FIXED
0000    103          .EXTRN  DBG$EQL_FIXED_FIXED
0000    104          .EXTRN  DBG$GB_LANGUAGE
0000    105          .EXTRN  DBG$GEQ_FIXED_FIXED
0000    106          .EXTRN  DBG$GTR_FIXED_FIXED
0000    107          .EXTRN  DBG$LEQ_FIXED_FIXED
0000    108          .EXTRN  DBG$LSS_FIXED_FIXED
0000    109          .EXTRN  DBG$MUL_FIXED_FIXED
0000    110          .EXTRN  DBG$NEQ_FIXED_FIXED
0000    111          .EXTRN  DBG$PRED_ENUM
0000    112          .EXTRN  DBG$SUB_FIXED_FIXED
0000    113          .EXTRN  DBG$SUCC_ENUM
0000    114          .EXTRN  DBG$UNARY_MINUS_FIXED
0000    115          .EXTRN  DBG$UNARY_PLUS_FIXED
0000    116          .EXTRN  PLI$ANDBIT
0000    117          .EXTRN  PLI$CATBIT
0000    118          .EXTRN  PLI$CMPBIT
0000    119          .EXTRN  PLI$DIV_PK_LONG
0000    120          .EXTRN  PLI$NOTBIT
0000    121          .EXTRN  PLI$ORBIT
0000    122
0000    123
0000    124 ; Invoke data definitions
0000    125 ;
0000    126          $SHRDEF                 ; Shared error messages
0000    127          $SSDEF                  ; System error codes
0000    128          $DSCDEF                 ; Data definitions
0000    129          $STSDEF                 ; Status code fields
0000    130          $DBGDEF                 ; Debug definitions
0000    131          $CHFDEF                 ; Signal argument definitions
0000    132          $MTHDEF                 ; Math. Lib. definitions
0000    133          $STRDEF                 ; String definitions
0000    134
0000    135
0000    136 ; Literal constants - these are the same as those in DBGLIB - see note above.
0000    137 ; (These used to be obtained from DBGCOMLIB.MLB, which was built from
0000    138 ; DBGLIB.REQ. However, I decided that the overhead of building DBGCOMLIB
0000    139 ; just for these four constants was not worth it. Since we already have
0000    140 ; several other things that must be kept consistent across DBGLIB.REQ and
0000    141 ; this module, I just duplicated the definitions here.)
0000    142 ;
0000    143 ; As you have noted, it turned out there are more things needed from
0000    144 ; DBGLIB, since you have taken it out, so someone else duplicated some
0000    145 ; more constants.
```

```
                                        0000   146 ;
                                        0000   147
                             00000004   0000   148          dbg$k_basic = 4
                             0000000C   0000   149          dbg$k_pli_ubit = 12
                             0000000E   0000   150          dbg$k_pli_abit = 14
                             00000001   0000   151          ort$k_min_rout = 1
                             0000011C   0000   152          ort$k_max_rout = 284
                             00000004   0000   153          token$k_integer = 4
                             00000005   0000   154          token$k_hex_integer = 5
                             0000000A   0000   155          token$k_bin_integer = 10
                             0000000B   0000   156          token$k_oct_integer = 11
                                        0000   157
                                00000000   158          .PSECT  DBG$PLIT,BYTE,PIC,SHR,NOWRT
                                        0000   159 errmsg:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    0000   160          .ASCIC  /DBGPERMOP\DBG$PERFORM_OPERATOR unknown routine index/
4F 5F 4D 52 4F 46 52 45 50 24 47 42     000C
6E 6B 6E 75 20 52 4F 54 41 52 45 50     0018
20 65 6E 69 74 75 6F 72 20 6E 77 6F     0024
                  78 65 64 6E 69        0030
                                 34     0000
                                        0035   161 errmsg1:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    0035   162          .ASCIC  /DBGPERMOP\DBG$CVT_TRFA_TO_VALUE foreign tokencode/
5F 41 46 52 54 5F 54 56 43 24 47 42     0041
72 6F 66 20 45 55 4C 41 56 5F 4F 54     004D
6F 63 6E 65 6B 6F 74 20 6E 67 69 65     0059
                              65 64     0065
                                 31     0035
                                        0067   163 errmsg2:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    0067   164          .ASCIC  /DBGPERMOP\DBG$CVT_TQUADWORD_TO_VALUE foreign tokencode/
44 41 55 51 54 5F 54 56 43 24 47 42     0073
55 4C 41 56 5F 4F 54 5F 44 52 4F 57     007F
6F 74 20 6E 67 69 65 72 6F 66 20 45     008B
                  65 64 6F 63 6E 65 6B  0097
                                 36     0067
                                        009E   165 errmsg3:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    009E   166          .ASCIC  /DBGPERMOP\DBG$CVT_TUQUADWORD_TO_VALUE foreign tokencode/
41 55 51 55 54 5F 54 56 43 24 47 42     00AA
4C 41 56 5F 4F 54 5F 44 52 4F 57 44     00B6
74 20 6E 67 69 65 72 6F 66 20 45 55     00C2
               65 64 6F 63 6E 65 6B 6F  00CE
                                 37     009E
                                        00D6   167 errmsg4:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    00D6   168          .ASCIC  /DBGPERMOP\DBG$CVT_TOCTAWORD_TO_VALUE foreign tokencode/
41 54 43 4F 54 5F 54 56 43 24 47 42     00E2
55 4C 41 56 5F 4F 54 5F 44 52 4F 57     00EE
6F 74 20 6E 67 69 65 72 6F 66 20 45     00FA
                  65 64 6F 63 6E 65 6B  0106
                                 36     00D6
                                        010D   169 errmsg5:
44 5C 50 4F 4D 52 45 50 47 42 44 00'    010D   170          .ASCIC  /DBGPERMOP\DBG$CVT_TUOCTAWORD_TO_VALUE foreign tokencode/
54 43 4F 55 54 5F 54 56 43 24 47 42     0119
4C 41 56 5F 4F 54 5F 44 52 4F 57 41     0125
74 20 6E 67 69 65 72 6F 66 20 45 55     0131
               65 64 6F 63 6E 65 6B 6F  013D
                                 37     010D
                                        0145   171
                                        0145   172 ; Own Storage
```

```
                       0145   173 ;
                  00000000    174         .PSECT  DBG$OWN,NOEXE,LONG,PIC
                       0000   175 save_operator_entry:
00000000               0000   176         .LONG   0
                       0004   177 save_result:
00000000               0004   178         .LONG   0
                       0008   179 dope1:
00000000               0008   180         .LONG   0
                       000C   181 dope2:
00000000               000C   182         .LONG   0
                       0010   183 scratch1:
00000020               0010   184         .BLKL   4
                       0020   185 scratch2:
00000030               0020   186         .BLKL   4
                       0030   187 arg1_desc:
00000000               0030   188         .LONG   0
                       0034   189 arg1_valdesc:
00000000               0034   190         .LONG   0
                       0038   191 arg2_desc:
00000000               0038   192         .LONG   0
                       003C   193 arg2_valdesc:
00000000               003C   194         .LONG   0
                       0040   195 result_desc:
00000000               0040   196         .LONG   0
                       0044   197 result_valdesc:
00000000               0044   198         .LONG   0
                       0048   199 operand:
0000005C               0048   200         .BLKL   5
                       005C   201 work_octa:
0000006C               005C   202         .BLKL   4
                       006C   203
                  00000000    204         .PSECT  DBG$CODE,NOWRT,BYTE,PIC,SHR
```

```
                        .SBTTL  dbg$perform_operator
                    ;++
                    ;GLOBAL ROUTINE DBG$PERFORM_OPERATOR(OPERATOR_ENTRY, ROUT_INDEX,
                    ;                   LEFT_ARG, RIGHT_ARG, RESULT_ADDR): NOVALUE =
                    ;
                    ; FUNCTION
                    ;       Perform the operation indicated by ROUT_INDEX, and leave the result
                    ;       at RESULT_ADDR.
                    ;
                    ;       There are assumptions made in this routine:
                    ;         1.  Always check overflow conditions, and give informational message
                    ;         2.  The result will be the same type as the operand's type
                    ;         3.  Comparison result will be always a longword value
                    ;         4.  There is no mixed mode operation, ie., BU + B, or no B + L, etc.
                    ;             some sort of conversion will be done before the operation
                    ;         5.  Language rules always have precedence.  There will be a specical
                    ;             routine to perform the operation for that language
                    ;
                    ; INPUTS
                    ;       OPERATOR_ENTRY  - A pointer to the Operator Lexical Token Entry for
                    ;                         operator.
                    ;
                    ;       ROUT_INDEX      - A case index indicating which operation is to be
                    ;                         performed.  The possible values for this index are
                    ;                         defined in DBGLIB.
                    ;
                    ;       LEFT_ARG        - The address of the left operand (or, in the case of
                    ;                         unary operators, the only operand). This is a
                    ;                         pointer to a value descriptor.
                    ;
                    ;       RIGHT_ARG       - The address of the right operand (in the case of
                    ;                         unary operators, this operand is not used and its
                    ;                         value will be zero. This address is a pointer
                    ;                         to a value descriptor.
                    ;
                    ;       RESULT_ADDR     - A pointer to a longword which contains a pointer
                    ;                         to a Value Descriptor which should be filled in
                    ;                         with the result of the operation. The reason for
                    ;                         the extra level of indirection is that in some
                    ;                         cases we may build a new descriptor instead
                    ;                         of using the one that is passed to us. In this
                    ;                         case, we update RESULT_ADDR to point to the
                    ;                         new descriptor.
                    ;
                    ; OUTPUTS
                    ;       The result of the operation is left in the result value descriptor.
                    ;       No value is returned.
                    ;--

                    ; Parameter offsets
                    ;
00000002  0000      bitpos          = 2
00000004  0000      operator_entry  = 4
00000008  0000      rout_index      = 8
0000000C  0000      left_arg        = 12
00000010  0000      right_arg       = 16
```

```
                     00000014  0000    263          result_addr     = 20
                     0000000C  0000    264          token_name      = 12
                               0000    265
                               0000    266
                               0000    267 ; Register usage
                               0000    268 ;        r2      VMS Address of the left operand in value descriptor
                               0000    269 ;        r3      VMS Address of the right operand in value descriptor
                               0000    270 ;        r4      VMS Address of the result in vlaue descriptor
                               0000    271 ;        r5      Temporary result
                               0000    272 ;        r6      Temporary result
                               0000    273 ;        If the operands are packed, then r6 = arg1 vms desc.
                               0000    274 ;                                        r7 = arg2 vms desc.
                               0000    275 ;                                        r8 = result vms desc.
                               0000    276
                               0000    277 ; PSL
                               0000    278 ;        FU      Floating Underflow Trap enable
                               0000    279 ;        IV      Interger Overflow trap disable
                               0000    280 ;
                               0000    281
                               0000    282
                         8FFC  0000    283 .ENTRY  dbg$perform_operator,^M<DV,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
           6D  00001BBE'EF     DE  0002  284          MOVAL   trap_msg_handler,(FP)   ; Enable trap exception handler
                       0040 8F     B8  0009  285          BISPSW  #^X40                   ; Enable Floating Underflow Traps
               00000000'EF  04 AC  D0  000D  286          MOVL    operator_entry(AP),save_operator_entry
                               0015    287                                            ; Save operator entry pointer in static area
                               0015    288
                               0015    289 ;NOTE:
                               0015    290 ; The block of code below uses absolute offsets such as 16 to get at fields
                               0015    291 ; inside of a Value Descriptor. This is because of difficulties of getting
                               0015    292 ; the symbolic definitions from DBGLIB into this module.
                               0015    293 ;
                               0015    294 ; So, until this is fixed, the block of code below has to be changed whenever
                               0015    295 ; the definition of a Value Descriptor changes. This just affects the
                               0015    296 ; code between the asterisks.
                               0015    297 ;*****************************************************************************
                    52  0C AC  D0  0015  298          MOVL    left_arg(AP),r2         ; Get the address of the left argument
           00000034'EF     52  D0  0019  299          MOVL    r2,arg1_valdesc         ; Save the value descriptor address
                       52  14  C0  0020  300          ADDL2   #20,r2                  ; Get pointer to VMS descriptor
           00000030'EF     52  D0  0023  301          MOVL    r2,arg1_desc            ; Save the VMS descriptor address
                    52  04 A2  D0  002A  302          MOVL    4(r2),r2                ; Get the pointer to value
                    53  10 AC  D0  002E  303          MOVL    right_arg(AP),r3        ; Get the address of the right argument
                        53      D5  0032  304          TSTL    r3                      ; Test to see if there is a right argument
                        15      13  0034  305          BEQL    get_result_addr$        ; Get the result address
           0000003C'EF     53  D0  0036  306          MOVL    r3,arg2_valdesc         ; Save the value descriptor address
                       53  14  C0  003D  307          ADDL2   #20,r3                  ; Get pointer to VMS descriptor
           00000038'EF     53  D0  0040  308          MOVL    r3,arg2_desc            ; Save the VMS descriptor address
                    53  04 A3  D0  0047  309          MOVL    4(r3),r3                ; Get the pointer to value
                               004B    310 get_result_addr$:
                    54  14 AC  D0  004B  311          MOVL    result_addr(AP),r4      ; Get the address of the result
                        54      64  D0  004F  312          MOVL    (r4),r4                 ; Extra level of indirection for result
           00000044'EF     54  D0  0052  313          MOVL    r4,result_valdesc       ; Save result value descriptor address
                       54  14  C0  0059  314          ADDL2   #20,r4
           00000040'EF     54  D0  005C  315          MOVL    r4,result_desc          ; Save the VMS descriptor address
                    54  04 A4  D0  0063  316          MOVL    4(r4),r4                ; Get the pointer to value
           00000004'EF     54  D0  0067  317          MOVL    r4,save_result          ; Save the result pointer
                               006E    318 ;*****************************************************************************
                               006E    319
```

```
                     006E    320 ; Case on routine index to perform the actual operation
                     006E    321 ;
 011C 8F   01   08 AC  AF  006E    322          CASEW   rout_index(AP),#ort$k_min_rout,#ort$k_max_rout
                 029E' 0075    323 bgn:     .WORD   ADD_B_B         - bgn
                 02A8' 0077    324          .WORD   ADD_BO_BU       - bgn
                 02B2' 0079    325          .WORD   ADD_W_Q         - bgn
                 02BC' 007B    326          .WORD   ADD_WO_WU       - bgn
                 02C6' 007D    327          .WORD   ADD_L_L         - bgn
                 02D0' 007F    328          .WORD   ADD_LO_LU       - bgn
                 02DA' 0081    329          .WORD   ADD_F_F         - bgn
                 02DF' 0083    330          .WORD   ADD_D_D         - bgn
                 02E4' 0085    331          .WORD   ADD_G_G         - bgn
                 02EA' 0087    332          .WORD   ADD_H_H         - bgn
                 02F0' 0089    333          .WORD   ADD_FC_FC       - bgn
                 02FC' 008B    334          .WORD   ADD_DC_DC       - bgn
                 0308' 008D    335          .WORD   ADD_GC_GC       - bgn
                 0316' 008F    336          .WORD   ADD_HC_HC       - bgn
                 0324' 0091    337          .WORD   SUB_B_B         - bgn
                 032E' 0093    338          .WORD   SUB_BO_BU       - bgn
                 0338' 0095    339          .WORD   SUB_W_Q         - bgn
                 0342' 0097    340          .WORD   SUB_WO_WU       - bgn
                 034C' 0099    341          .WORD   SUB_L_L         - bgn
                 0356' 009B    342          .WORD   SUB_LO_LU       - bgn
                 0360' 009D    343          .WORD   SUB_F_F         - bgn
                 0365' 009F    344          .WORD   SUB_D_D         - bgn
                 036A' 00A1    345          .WORD   SUB_G_G         - bgn
                 0370' 00A3    346          .WORD   SUB_H_H         - bgn
                 0376' 00A5    347          .WORD   SUB_FC_FC       - bgn
                 0382' 00A7    348          .WORD   SUB_DC_DC       - bgn
                 038E' 00A9    349          .WORD   SUB_GC_GC       - bgn
                 039C' 00AB    350          .WORD   SUB_HC_HC       - bgn
                 03AA' 00AD    351          .WORD   DIV_B_B         - bgn
                 03B4' 00AF    352          .WORD   DIV_BO_BU       - bgn
                 03C7' 00B1    353          .WORD   DIV_W_Q         - bgn
                 03D1' 00B3    354          .WORD   DIV_WO_WU       - bgn
                 03E4' 00B5    355          .WORD   DIV_L_L         - bgn
                 03EE' 00B7    356          .WORD   DIV_LO_LU       - bgn
                 0416' 00B9    357          .WORD   DIV_F_F         - bgn
                 041B' 00BB    358          .WORD   DIV_D_D         - bgn
                 0420' 00BD    359          .WORD   DIV_G_G         - bgn
                 0426' 00BF    360          .WORD   DIV_H_H         - bgn
                 042C' 00C1    361          .WORD   DIV_FC_FC       - bgn
                 0465' 00C3    362          .WORD   DIV_DC_DC       - bgn
                 04D6' 00C5    363          .WORD   DIV_GC_GC       - bgn
                 0555' 00C7    364          .WORD   DIV_HC_HC       - bgn
                 05D4' 00C9    365          .WORD   MUL_B_B         - bgn
                 05DE' 00CB    366          .WORD   MUL_BO_BU       - bgn
                 05F7' 00CD    367          .WORD   MUL_W_Q         - bgn
                 0601' 00CF    368          .WORD   MUL_WO_WU       - bgn
                 061A' 00D1    369          .WORD   MUL_L_L         - bgn
                 0624' 00D3    370          .WORD   MUL_LO_LU       - bgn
                 0642' 00D5    371          .WORD   MUL_F_F         - bgn
                 0647' 00D7    372          .WORD   MUL_D_D         - bgn
                 064C' 00D9    373          .WORD   MUL_G_G         - bgn
                 0652' 00DB    374          .WORD   MUL_H_H         - bgn
                 0658' 00DD    375          .WORD   MUL_FC_FC       - bgn
                 0676' 00DF    376          .WORD   MUL_DC_DC       - bgn
```

```
06B4' 00E1   377        .WORD   MUL_GC_GC       - bgn
06F8' 00E3   378        .WORD   MUL_HC_HC       - bgn
073C' 00E5   379        .WORD   MOD_L_L         - bgn
0756' 00E7   380        .WORD   MOD_LO_LU       - bgn
076E' 00E9   381        .WORD   REM_L_L         - bgn
0779' 00EB   382        .WORD   REM_LO_LU       - bgn
0791' 00ED   383        .WORD   SHIFT_LEFT_L_L  - bgn
0799' 00EF   384        .WORD   SHIFT_RT_L_L    - bgn
07A8' 00F1   385        .WORD   SHIFT_RT_LO_LU  - bgn
07D0' 00F3   386        .WORD   POWER_W_W       - bgn
07E1' 00F5   387        .WORD   POWER_L_L       - bgn
07F0' 00F7   388        .WORD   POWER_F_L       - bgn
0801' 00F9   389        .WORD   POWER_D_L       - bgn
0812' 00FB   390        .WORD   POWER_G_L       - bgn
0825' 00FD   391        .WORD   POWER_H_L       - bgn
0838' 00FF   392        .WORD   POWER_FC_L      - bgn
0851' 0101   393        .WORD   POWER_DC_L      - bgn
086A' 0103   394        .WORD   POWER_GC_L      - bgn
0887' 0105   395        .WORD   POWER_F_F       - bgn
0898' 0107   396        .WORD   POWER_D_F       - bgn
08A9' 0109   397        .WORD   POWER_F_D       - bgn
08BA' 010B   398        .WORD   POWER_D_D       - bgn
08CB' 010D   399        .WORD   POWER_G_G       - bgn
08DF' 010F   400        .WORD   POWER_H_H       - bgn
08F3' 0111   401        .WORD   POWER_FC_FC     - bgn
0929' 0113   402        .WORD   POWER_DC_DC     - bgn
095F' 0115   403        .WORD   POWER_GC_GC     - bgn
099C' 0117   404        .WORD   CONCAT_T_T      - bgn
09DC' 0119   405        .WORD   EQL_VT_VT       - bgn
09EE' 011B   406        .WORD   EQL_T_T         - bgn
0A0D' 011D   407        .WORD   GEQ_VT_VT       - bgn
0A1F' 011F   408        .WORD   GEQ_T_T         - bgn
0A3E' 0121   409        .WORD   GTR_VT_VT       - bgn
0A50' 0123   410        .WORD   GTR_T_T         - bgn
0A6F' 0125   411        .WORD   LEQ_VT_VT       - bgn
0A81' 0127   412        .WORD   LEQ_T_T         - bgn
0AA0' 0129   413        .WORD   LSS_VT_VT       - bgn
0AB2' 012B   414        .WORD   LSS_T_T         - bgn
0AD1' 012D   415        .WORD   NEQ_VT_VT       - bgn
0AE3' 012F   416        .WORD   NEQ_T_T         - bgn
0B02' 0131   417        .WORD   EQL_B_B         - bgn
0B0F' 0133   418        .WORD   EQL_W_W         - bgn
0B1C' 0135   419        .WORD   EQL_L_L         - bgn
0B29' 0137   420        .WORD   EQL_F_F         - bgn
0B36' 0139   421        .WORD   EQL_D_D         - bgn
0B43' 013B   422        .WORD   EQL_G_G         - bgn
0B51' 013D   423        .WORD   EQL_H_H         - bgn
0B5F' 013F   424        .WORD   EQL_FC_FC       - bgn
0B73' 0141   425        .WORD   EQL_DC_DC       - bgn
0B87' 0143   426        .WORD   EQL_GC_GC       - bgn
0B9D' 0145   427        .WORD   EQL_HC_HC       - bgn
0BB3' 0147   428        .WORD   NEQ_B_B         - bgn
0BC0' 0149   429        .WORD   NEQ_W_W         - bgn
0BCD' 014B   430        .WORD   NEQ_L_L         - bgn
0BDA' 014D   431        .WORD   NEQ_F_F         - bgn
0BE7' 014F   432        .WORD   NEQ_D_D         - bgn
0BF4' 0151   433        .WORD   NEQ_G_G         - bgn
```

```
0C02' 0153   434        .WORD   NEQ_H_H         - bgn
0C10' 0155   435        .WORD   NEQ_FC_FC       - bgn
0C24' 0157   436        .WORD   NEQ_DC_DC       - bgn
0C38' 0159   437        .WORD   NEQ_GC_GC       - bgn
0C4E' 015B   438        .WORD   NEQ_HC_HC       - bgn
0C64' 015D   439        .WORD   GEQ_B_B         - bgn
0C71' 015F   440        .WORD   GEQ_W_W         - bgn
0C7E' 0161   441        .WORD   GEQ_L_L         - bgn
0C8B' 0163   442        .WORD   GEQ_LU_LU       - bgn
0C98' 0165   443        .WORD   GEQ_F_F         - bgn
0CA5' 0167   444        .WORD   GEQ_D_D         - bgn
0CB2' 0169   445        .WORD   GEQ_G_G         - bgn
0CC0' 016B   446        .WORD   GEQ_H_H         - bgn
0CCE' 016D   447        .WORD   GTR_B_B         - bgn
0CDB' 016F   448        .WORD   GTR_W_W         - bgn
0CE8' 0171   449        .WORD   GTR_L_L         - bgn
0CF5' 0173   450        .WORD   GTR_LU_LU       - bgn
0D02' 0175   451        .WORD   GTR_F_F         - bgn
0D0F' 0177   452        .WORD   GTR_D_D         - bgn
0D1C' 0179   453        .WORD   GTR_G_G         - bgn
0D2A' 017B   454        .WORD   GTR_H_H         - bgn
0D38' 017D   455        .WORD   LEQ_B_B         - bgn
0D45' 017F   456        .WORD   LEQ_W_W         - bgn
0D52' 0181   457        .WORD   LEQ_L_L         - bgn
0D5F' 0183   458        .WORD   LEQ_LU_LU       - bgn
0D6C' 0185   459        .WORD   LEQ_F_F         - bgn
0D79' 0187   460        .WORD   LEQ_D_D         - bgn
0D86' 0189   461        .WORD   LEQ_G_G         - bgn
0D94' 018B   462        .WORD   LEQ_H_H         - bgn
0DA2' 018D   463        .WORD   LSS_B_B         - bgn
0DAF' 018F   464        .WORD   LSS_W_W         - bgn
0DBC' 0191   465        .WORD   LSS_L_L         - bgn
0DC9' 0193   466        .WORD   LSS_LU_LU       - bgn
0DD6' 0195   467        .WORD   LSS_F_F         - bgn
0DE3' 0197   468        .WORD   LSS_D_D         - bgn
0DF0' 0199   469        .WORD   LSS_G_G         - bgn
0DFE' 019B   470        .WORD   LSS_H_H         - bgn
0E0C' 019D   471        .WORD   BIT_AND_B_B     - bgn
0E14' 019F   472        .WORD   BIT_AND_W_W     - bgn
0E1C' 01A1   473        .WORD   BIT_AND_L_L     - bgn
0E24' 01A3   474        .WORD   BIT_EQV_B_B     - bgn
0E2C' 01A5   475        .WORD   BIT_EQV_W_W     - bgn
0E34' 01A7   476        .WORD   BIT_EQV_L_L     - bgn
0E3C' 01A9   477        .WORD   BIT_NOT_B       - bgn
0E40' 01AB   478        .WORD   BIT_NOT_W       - bgn
0E44' 01AD   479        .WORD   BIT_NOT_L       - bgn
0E48' 01AF   480        .WORD   BIT_OR_B_B      - bgn
0E4D' 01B1   481        .WORD   BIT_OR_W_W      - bgn
0E52' 01B3   482        .WORD   BIT_OR_L_L      - bgn
0E57' 01B5   483        .WORD   BIT_XOR_B_B     - bgn
0E5C' 01B7   484        .WORD   BIT_XOR_W_W     - bgn
0E61' 01B9   485        .WORD   BIT_XOR_L_L     - bgn
0E7E' 01BB   486        .WORD   AND_B_B         - bgn
0E8A' 01BD   487        .WORD   AND_W_W         - bgn
0E96' 01BF   488        .WORD   AND_L_L         - bgn
0EA2' 01C1   489        .WORD   AND_D_D         - bgn
0EB1' 01C3   490        .WORD   NOT_B           - bgn
```

```
0EB8' 01C5   491         .WORD   NOT_W              - bgn
0EBF' 01C7   492         .WORD   NOT_L              - bgn
0EC6' 01C9   493         .WORD   NOT_D              - bgn
0ED1' 01CB   494         .WORD   OR_B_B             - bgn
0EDC' 01CD   495         .WORD   OR_W_W             - bgn
0EE7' 01CF   496         .WORD   OR_L_L             - bgn
0EF2' 01D1   497         .WORD   OR_D_D             - bgn
0F01' 01D3   498         .WORD   XOR_L_L            - bgn
0F17' 01D5   499         .WORD   UNARY_MINUS_B      - bgn
0F20' 01D7   500         .WORD   UNARY_MINUS_W      - bgn
0F29' 01D9   501         .WORD   UNARY_MINUS_L      - bgn
0F32' 01DB   502         .WORD   UNARY_MINUS_LU     - bgn
0F3B' 01DD   503         .WORD   UNARY_MINUS_F      - bgn
0F3F' 01DF   504         .WORD   UNARY_MINUS_D      - bgn
0F43' 01E1   505         .WORD   UNARY_MINUS_G      - bgn
0F48' 01E3   506         .WORD   UNARY_MINUS_H      - bgn
0F4D' 01E5   507         .WORD   UNARY_MINUS_FC     - bgn
0F56' 01E7   508         .WORD   UNARY_MINUS_DC     - bgn
0F5F' 01E9   509         .WORD   UNARY_MINUS_GC     - bgn
0F6A' 01EB   510         .WORD   UNARY_MINUS_HC     - bgn
0F75' 01ED   511         .WORD   UNARY_PLUS_B       - bgn
0F79' 01EF   512         .WORD   UNARY_PLUS_W       - bgn
0F7D' 01F1   513         .WORD   UNARY_PLUS_L       - bgn
0F81' 01F3   514         .WORD   UNARY_PLUS_F       - bgn
0F85' 01F5   515         .WORD   UNARY_PLUS_D       - bgn
0F89' 01F7   516         .WORD   UNARY_PLUS_G       - bgn
0F8E' 01F9   517         .WORD   UNARY_PLUS_H       - bgn
0F93' 01FB   518         .WORD   UNARY_PLUS_FC      - bgn
0F9C' 01FD   519         .WORD   UNARY_PLUS_DC      - bgn
0FA5' 01FF   520         .WORD   UNARY_PLUS_GC      - bgn
0FB0' 0201   521         .WORD   UNARY_PLUS_HC      - bgn
0FBB' 0203   522         .WORD   ABS_B              - bgn
0FC7' 0205   523         .WORD   ABS_W              - bgn
0FD3' 0207   524         .WORD   ABS_L              - bgn
0FDF' 0209   525         .WORD   ABS_F              - bgn
0FEB' 020B   526         .WORD   ABS_D              - bgn
0FF7' 020D   527         .WORD   ABS_G              - bgn
1006' 020F   528         .WORD   ABS_H              - bgn
1015' 0211   529         .WORD   INDIRECT_LU        - bgn
1026' 0213   530         .WORD   INDIRECT_TPTR      - bgn
1038' 0215   531         .WORD   BITSELECT          - bgn
104F' 0217   532         .WORD   DIFFERENCE_SET_SET - bgn
10A3' 0219   533         .WORD   EQL_SET_SET        - bgn
10EA' 021B   534         .WORD   GEQ_SET_SET        - bgn
1109' 021D   535         .WORD   IN_SET_SET         - bgn
1114' 021F   536         .WORD   INTERSECT_SET_SET  - bgn
1169' 0221   537         .WORD   LEQ_SET_SET        - bgn
11B7' 0223   538         .WORD   NEQ_SET_SET        - bgn
1212' 0225   539         .WORD   UNION_SET_SET      - bgn
1262' 0227   540         .WORD   ADDRESS_L          - bgn
1276' 0229   541         .WORD   SIZEOF_L           - bgn
1287' 022B   542         .WORD   ADD_TPTR_L         - bgn
12A1' 022D   543         .WORD   SUB_TPTR_L         - bgn
12BB' 022F   544         .WORD   SUB_TPTR_TPTR      - bgn
0E66' 0231   545         .WORD   BIT_IMP_B_B        - bgn
0E6E' 0233   546         .WORD   BIT_IMP_W_W        - bgn
0E76' 0235   547         .WORD   BIT_IMP_L_L        - bgn
```

```
12D5' 0237   548        .WORD   PRE_INCR_L      - bgn
12D5' 0239   549        .WORD   PRE_INCR_LU     - bgn
12E8' 023B   550        .WORD   PRE_INCR_D      - bgn
12FC' 023D   551        .WORD   PRE_INCR_TPTR   - bgn
131A' 023F   552        .WORD   POST_INCR_L     - bgn
131A' 0241   553        .WORD   POST_INCR_LU    - bgn
132B' 0243   554        .WORD   POST_INCR_D     - bgn
131A' 0245   555        .WORD   POST_INCR_TPTR  - bgn
133C' 0247   556        .WORD   PRE_DECR_L      - bgn
133C' 0249   557        .WORD   PRE_DECR_LU     - bgn
134F' 024B   558        .WORD   PRE_DECR_D      - bgn
1363' 024D   559        .WORD   PRE_DECR_TPTR   - bgn
1381' 024F   560        .WORD   POST_DECR_L     - bgn
1381' 0251   561        .WORD   POST_DECR_LU    - bgn
1392' 0253   562        .WORD   POST_DECR_D     - bgn
1381' 0255   563        .WORD   POST_DECR_TPTR  - bgn
13C7' 0257   564        .WORD   ADD_P_P         - bgn
13A3' 0259   565        .WORD   SUB_P_P         - bgn
143F' 025B   566        .WORD   MUL_P_P         - bgn
14E8' 025D   567        .WORD   DIV_P_P         - bgn
1571' 025F   568        .WORD   UNARY_PLUS_P    - bgn
1554' 0261   569        .WORD   UNARY_MINUS_P   - bgn
159B' 0263   570        .WORD   EQL_P_P         - bgn
15D3' 0265   571        .WORD   NEQ_P_P         - bgn
160B' 0267   572        .WORD   GTR_P_P         - bgn
161F' 0269   573        .WORD   GEQ_P_P         - bgn
1633' 026B   574        .WORD   LSS_P_P         - bgn
1647' 026D   575        .WORD   LEQ_P_P         - bgn
170E' 026F   576        .WORD   CONCAT_TF_TF    - bgn
172C' 0271   577        .WORD   EQL_TF_TF       - bgn
1812' 0273   578        .WORD   NEQ_TF_TF       - bgn
1781' 0275   579        .WORD   GTR_TF_TF       - bgn
1752' 0277   580        .WORD   GEQ_TF_TF       - bgn
17E1' 0279   581        .WORD   LSS_TF_TF       - bgn
17B2' 027B   582        .WORD   LEQ_TF_TF       - bgn
1838' 027D   583        .WORD   BIT_NOT_TF_TF   - bgn
184E' 027F   584        .WORD   BIT_AND_TF_TF   - bgn
186C' 0281   585        .WORD   BIT_OR_TF_TF    - bgn
18F6' 0283   586        .WORD   EQL_RFA_RFA     - bgn
190A' 0285   587        .WORD   NEQ_RFA_RFA     - bgn
191E' 0287   588        .WORD   UNARY_PLUS_Q    - bgn
1924' 0289   589        .WORD   UNARY_MINUS_Q   - bgn
1936' 028B   590        .WORD   UNARY_PLUS_O    - bgn
1941' 028D   591        .WORD   UNARY_MINUS_O   - bgn
1965' 028F   592        .WORD   SUCC_ENUM       - bgn                   ;BB001
1979' 0291   593        .WORD   PRED_ENUM       - bgn                   ;BB001
198D' 0293   594        .WORD   UNARY_PLUS_FIXED- bgn
19A1' 0295   595        .WORD   UNARY_MINUS_FIXED-bgn
19B5' 0297   596        .WORD   ABS_FIXED       - bgn
19C9' 0299   597        .WORD   ADD_FIXED_FIXED - bgn
19E3' 029B   598        .WORD   SUB_FIXED_FIXED - bgn
19FD' 029D   599        .WORD   MUL_FIXED_FIXED - bgn
1A17' 029F   600        .WORD   DIV_FIXED_FIXED - bgn
1A31' 02A1   601        .WORD   EQL_FIXED_FIXED - bgn
1A4B' 02A3   602        .WORD   NEQ_FIXED_FIXED - bgn
1A65' 02A5   603        .WORD   LSS_FIXED_FIXED - bgn
1A7F' 02A7   604        .WORD   GTR_FIXED_FIXED - bgn
```

```
                     1A99' 02A9    605          .WORD   LEQ_FIXED_FIXED - bgn
                     1AB3' 02AB    606          .WORD   GEQ_FIXED_FIXED - bgn
                      1892   31    02AD    607          BRW     unknown_rout_index
                             02B0    608
                             02B0    609 int_overflow:
      52  00000000'EF         D0    02B0    610          MOVL    save_operator_entry,r2
        52     0C A2          DE    02B7    611          MOVAL   token_name(r2),r2
                  52          DD    02BB    612          PUSHL   r2
                  01          DD    02BD    613          PUSHL   #1
            000286A3 8F       DD    02BF    614          PUSHL   #dbg$_iintovf
00000000'GF       03          FB    02C5    615          CALLS   #3,G^LIB$SIGNAL
                  04          02CC    616          RET
                             02CD    617
                             02CD    618 div_by_zero:
            00028240 8F       DD    02CD    619          PUSHL   #dbg$_divbyzero
00000000'GF       01          FB    02D3    620          CALLS   #1,G^LIB$SIGNAL
                  04          02DA    621          RET
                             02DB    622
                             02DB    623 shift_count_negative:
            00028EC0 8F       DD    02DB    624          PUSHL   #dbg$_sfcntneg
00000000'GF       01          FB    02E1    625          CALLS   #1,G^LIB$SIGNAL
                  04          02E8    626          RET
                             02E9    627
                             02E9    628 string_truncate:
      52  00000000'EF         D0    02E9    629          MOVL    save_operator_entry,r2
        52     0C A2          DE    02F0    630          MOVAL   token_name(r2),r2
                  52          DD    02F4    631          PUSHL   r2
                  01          DD    02F6    632          PUSHL   #1
            000286AB 8F       DD    02F8    633          PUSHL   #dbg$_istrtru
00000000'GF       03          FB    02FE    634          CALLS   #3,G^LIB$SIGNAL
                             0305    635
                             0305    636 branch_ret:
04    00000000'EF         91    0305    637          CMPB    DBG$GB_LANGUAGE, #DBG$K_BASIC
                  01          13    030C    638          BEQL    basic$
                  04          030E    639          RET
                             030F    640 basic$:
            64    64        CE    030F    641          MNEGL   (r4), (r4)
                  04          0312    642          RET
                             0313    643
                             0313    644
                             0313    645 ;*****************************************************************************
                             0313    646 ;
                             0313    647 ;       A  D  D    and    S  U  B  T  R  A  C  T
                             0313    648 ;
                             0313    649 ;*****************************************************************************
                             0313    650
                             0313    651
                             0313    652 ; Additive Operators - Addition
                             0313    653 ;
                             0313    654 add_b_b:
      64    63    62        81    0313    655          ADDB3   (r2),(r3),(r4)
                  01          1D    0317    656          BVS     add_b$                  ; Branch on V bit set
                  04          0319    657          RET
                             031A    658 add_b$:
                  FF93        31    031A    659          BRW     int_overflow
                             031D    660
                             031D    661 add_bu_bu:
```

```
                64   63   62   81   031D   662          ADDB3   (r2),(r3),(r4)
                          01   1F   0321   663          BCS     add_bu$                 ; Branch on C bit set
                               04   0323   664          RET
                               0324   665 add_bu$:
                     FF89      31   0324   666          BRW     int_overflow
                               0327   667
                               0327   668 add_w_w:
                64   63   62   A1   0327   669          ADDW3   (r2),(r3),(r4)
                          01   1D   032B   670          BVS     add_w$
                               04   032D   671          RET
                               032E   672 add_w$:
                     FF7F      31   032E   673          BRW     int_overflow
                               0331   674
                               0331   675 add_wu_wu:
                64   63   62   A1   0331   676          ADDW3   (r2),(r3),(r4)
                          01   1F   0335   677          BCS     add_wu$
                               04   0337   678          RET
                               0338   679 add_wu$:
                     FF75      31   0338   680          BRW     int_overflow
                               033B   681
                               033B   682 add_l_l:
                64   63   62   C1   033B   683          ADDL3   (r2),(r3),(r4)
                          01   1D   033F   684          BVS     add_l$
                               04   0341   685          RET
                               0342   686 add_l$:
                     FF6B      31   0342   687          BRW     int_overflow
                               0345   688
                               0345   689 add_lu_lu:
                64   63   62   C1   0345   690          ADDL3   (r2),(r3),(r4)
                          01   1F   0349   691          BCS     add_lu$
                               04   034B   692          RET
                               034C   693 add_lu$:
                     FF61      31   034C   694          BRW     int_overflow
                               034F   695
                               034F   696 add_f_f:
                64   63   62   41   034F   697          ADDF3   (r2),(r3),(r4)
                               04   0353   698          RET
                               0354   699 add_d_d:
                64   63   62   61   0354   700          ADDD3   (r2),(r3),(r4)
                               04   0358   701          RET
                               0359   702 add_g_g:
                64   63   62 41FD   0359   703          ADDG3   (r2),(r3),(r4)
                               04   035E   704          RET
                               035F   705 add_h_h:
                64   63   62 61FD   035F   706          ADDH3   (r2),(r3),(r4)
                               04   0364   707          RET
                               0365   708
                               0365   709 add_fc_fc:
                64   63   62   41   0365   710          ADDF3   (r2),(r3),(r4)          ; Add Real Part
04 A4   04 A3     04 A2     41   0369   711          ADDF3   4(r2),4(r3),4(r4)       ; Add Imaginary Part
                               04   0370   712          RET
                               0371   713 add_dc_dc:
                64   63   62   61   0371   714          ADDD3   (r2),(r3),(r4)
08 A4   08 A3     08 A2     61   0375   715          ADDD3   8(r2),8(r3),8(r4)
                               04   037C   716          RET
                               037D   717 add_gc_gc:
                64   63   62 41FD   037D   718          ADDG3   (r2),(r3),(r4)
```

```
                08 A4   08 A3   08 A2 41FD  0382   719          ADDG3   8(r2),8(r3),8(r4)
                                      04  038A   720          RET
                                          038B   721 add_hc_hc:
                    64      63      62 61FD  038B   722          ADDH3   (r2),(r3),(r4)
                10 A4   10 A3   10 A2 61FD  0390   723          ADDH3   16(r2),16(r3),16(r4)
                                      04  0398   724          RET
                                          0399   725
                                          0399   726
                                          0399   727 ; Additive Operators - Subtraction
                                          0399   728 ;
                                          0399   729 sub_b_b:
                    64      62      63  83  0399   730          SUBB3   (r3),(r2),(r4)
                                      01  1D  039D   731          BVS     sub_b$
                                      04  039F   732          RET
                                          03A0   733 sub_b$:
                                    FF0D  31  03A0   734          BRW     int_overflow
                                          03A3   735
                                          03A3   736 sub_bu_bu:
                    64      62      63  83  03A3   737          SUBB3   (r3),(r2),(r4)
                                      01  1F  03A7   738          BCS     sub_bu$
                                      04  03A9   739          RET
                                          03AA   740 sub_bu$:
                                    FF03  31  03AA   741          BRW     int_overflow
                                          03AD   742
                                          03AD   743 sub_w_w:
                    64      62      63  A3  03AD   744          SUBW3   (r3),(r2),(r4)
                                      01  1D  03B1   745          BVS     sub_w$
                                      04  03B3   746          RET
                                          03B4   747 sub_w$:
                                    FEF9  31  03B4   748          BRW     int_overflow
                                          03B7   749
                                          03B7   750 sub_wu_wu:
                64      62      63  A3  03B7   751          SUBW3   (r3),(r2),(r4)
                                      01  1F  03BB   752          BCS     sub_wu$
                                      04  03BD   753          RET
                                          03BE   754 sub_wu$:
                                    FEEF  31  03BE   755          BRW     int_overflow
                                          03C1   756
                                          03C1   757 sub_l_l:
                64      62      63  C3  03C1   758          SUBL3   (r3),(r2),(r4)
                                      01  1D  03C5   759          BVS     sub_l$
                                      04  03C7   760          RET
                                          03C8   761 sub_l$:
                                    FEE5  31  03C8   762          BRW     int_overflow
                                          03CB   763
                                          03CB   764 sub_lu_lu:
                64      62      63  C3  03CB   765          SUBL3   (r3),(r2),(r4)
                                      01  1F  03CF   766          BCS     sub_lu$
                                      04  03D1   767          RET
                                          03D2   768 sub_lu$:
                                    FEDB  31  03D2   769          BRW     int_overflow
                                          03D5   770
                                          03D5   771 sub_f_f:
                64      62      63  43  03D5   772          SUBF3   (r3),(r2),(r4)
                                      04  03D9   773          RET
                                          03DA   774 sub_d_d:
                64      62      63  63  03DA   775          SUBD3   (r3),(r2),(r4)
```

```
                                04  03DE   776          RET
                                    03DF   777 sub_g_g:
                64   62   63 43FD  03DF   778          SUBG3   (r3),(r2),(r4)
                                04  03E4   779          RET
                                    03E5   780 sub_h_h:
                64   62   63 63FD  03E5   781          SUBH3   (r3),(r2),(r4)
                                04  03EA   782          RET
                                    03EB   783
                                    03EB   784 sub_fc_fc:
                64   62   63   43  03EB   785          SUBF3   (r3),(r2),(r4)
          04 A4 04 A2   04 A3   43  03EF   786          SUBF3   4(r3),4(r2),4(r4)
                                04  03F6   787          RET
                                    03F7   788 sub_dc_dc:
                64   62   63   63  03F7   789          SUBD3   (r3),(r2),(r4)
          08 A4 08 A2   08 A3   63  03FB   790          SUBD3   8(r3),8(r2),8(r4)
                                04  0402   791          RET
                                    0403   792 sub_gc_gc:
                64   62   63 43FD  0403   793          SUBG3   (r3),(r2),(r4)
          08 A4 08 A2   08 A3 43FD  0408   794          SUBG3   8(r3),8(r2),8(r4)
                                04  0410   795          RET
                                    0411   796 sub_hc_hc:
                64   62   63 63FD  0411   797          SUBH3   (r3),(r2),(r4)
          10 A4 10 A2   10 A3 63FD  0416   798          SUBH3   16(r3),16(r2),16(r4)
                                04  041E   799          RET
                                    041F   800
                                    041F   801
                                    041F   802 ;*****************************************************************************
                                    041F   803 ;
                                    041F   804 ;       D  I  V  I  D  E    and    M  U  L  T  I  P  L  Y
                                    041F   805 ;
                                    041F   806 ;*****************************************************************************
                                    041F   807
                                    041F   808
                                    041F   809 ; Multiplicative - Division
                                    041F   810 ;
                                    041F   811 div_b_b:
64   62   63   87                    041F   812          DIVB3   (r3),(r2),(r4)
               01   1D              0423   813          BVS     div_b$
                    04              0425   814          RET
                                    0426   815 div_b$:
               FE87 31              0426   816          BRW     int_overflow
                                    0429   817
                                    0429   818 div_bu_bu:
          52   62   9A              0429   819          MOVZBL  (r2),r2                 ; Zero-extend Dividend
          53   63   9A              042C   820          MOVZBL  (r3),r3                 ; Zero-extend Divisor
               08   13              042F   821          BEQL    div_bu$                 ; Test divisor is not zero
55        52   53   C7              0431   822          DIVL3   r3,r2,r5
          64   55   90              0435   823          MOVB    r5,(r4)
                    04              0438   824          RET
                                    0439   825 div_bu$:
               FE91 31              0439   826          BRW     div_by_zero
                                    043C   827
                                    043C   828 div_w_w:
64   62   63   A7                    043C   829          DIVW3   (r3),(r2),(r4)
               01   1D              0440   830          BVS     div_w$
                    04              0442   831          RET
                                    0443   832 div_w$:
```

```
                FE6A   31  0443   833          BRW     int_overflow
                       0446   834
                       0446   835 div_wu_wu:
           52    62    3C  0446   836          MOVZWL  (r2),r2
           53    63    3C  0449   837          MOVZWL  (r3),r3
                 08    13  044C   838          BEQL    div_wu$
     55    62    63    C7  044E   839          DIVL3   (r3),(r2),r5
     64    55    B0  0452   840          MOVW    r5,(r4)
                 04  0455   841          RET
                       0456   842 div_wu$:
                FE74   31  0456   843          BRW     div_by_zero
                       0459   844
                       0459   845 div_l_l:
     64    62    63    C7  0459   846          DIVL3   (r3),(r2),(r4)
                 01    1D  045D   847          BVS     div_l$
                 04  045F   848          RET
                       0460   849 div_l$:
                FE4D   31  0460   850          BRW     int_overflow
                       0463   851
                       0463   852 div_lu_lu:                        ;
                   63    D5  0463   853          TSTL    (r3)                    ; Check for divide by 0
                   1A    13  0465   854          BEQL    div_lu1$                ; Branch on divide by 0
                   0B    14  0467   855          BGTR    ext_prec_div            ; Branch on +Divisor
                   55    D4  0469   856          CLRL    r5                      ; Assume zero result
              62   63    D1  046B   857          CMPL    (r3),(r2)               ; Compare Divident and -Divisor
                       046E   858                                              ; -- (divided by large number)
                   17    1A  046E   859          BGTRU   set_quotient            ; Result is 0, set it
                   55    D6  0470   860          INCL    r5                      ; Result is 1
                   13    11  0472   861          BRB     set_quotient            ; Set it
                       0474   862 ext_prec_div:
              55   62    D0  0474   863          MOVL    (r2),r5                 ; Divident is quarward
                   56    D4  0477   864          CLRL    r6
     52    64    55    63    7B  0479   865          EDIV    (r3),r5,(r4),r2         ; Perform the DIV
                 04    1F  047E   866          BCS     div_lu2$
                 04  0480   867          RET
                       0481   868 div_lu1$:
                   FE49   31  0481   869          BRW     div_by_zero
                       0484   870 div_lu2$:
                   FE29   31  0484   871          BRW     int_overflow
                       0487   872
                       0487   873 set_quotient:
              64   55    D0  0487   874          MOVL    r5,(r4)                 ; Set result
                 04  048A   875          RET
                       048B   876
                       048B   877 div_f_f:
          64   62    63    47  048B   878          DIVF3   (r3),(r2),(r4)
                 04  048F   879          RET
                       0490   880
                       0490   881 div_d_d:
          64   62    63    67  0490   882          DIVD3   (r3),(r2),(r4)
                 04  0494   883          RET
                       0495   884
                       0495   885 div_g_g:
          64   62    63 47FD  0495   886          DIVG3   (r3),(r2),(r4)
                 04  049A   887          RET
                       049B   888
                       049B   889 div_h_h:
```

```
                64   62   63 67FD   049B   890          DIVH3   (r3),(r2),(r4)
                                04   04A0   891          RET
                                     04A1   892
                                     04A1   893 div_fc_fc:
                55   63   63   45   04A1   894          MULF3   (r3),(r3),r5          ; X = B<1>*B<1> + B<2>*B<2>
          56  04 A3   04 A3   45   04A5   895          MULF3   4(r3),4(r3),r6
                51   56   55   41   04AB   896          ADDF3   r5,r6,r1
                          51   53   04AF   897          TSTF    r1
                          24   13   04B1   898          BEQL    div_fc$
                55   62   63   45   04B3   899          MULF3   (r3),(r2),r5          ; Real Part Division
          56  04 A2   04 A3   45   04B7   900          MULF3   4(r3),4(r2),r6        ; C<1> = (A<1>*B<1> + A<2>*B<2>) / X
                     56   55   40   04BD   901          ADDF2   r5,r6
                64   56   51   47   04C0   902          DIVF3   r1,r6,(r4)
             55   04 A2   63   45   04C4   903          MULF3   (r3),4(r2),r5        ; Imaginary Part Division
             56   62   04 A3   45   04C9   904          MULF3   4(r3),(r2),r6        ; C<2> = (A<2>*B<1> - A<1>*B<2>) / X
                     55   56   42   04CE   905          SUBF2   r6,r5
             04 A4   55   51   47   04D1   906          DIVF3   r1,r5,4(r4)
                               04   04D6   907          RET
                                    04D7   908 div_fc$:
                     FDF3   31   04D7   909          BRW     div_by_zero
                                    04DA   910
                                    04DA   911 div_dc_dc:
       00000010'EF   63   63   65   04DA   912          MULD3   (r3),(r3),scratch1    ; X = B<1>*B<1> + B<2>*B<2>
   00000020'EF   08 A3   08 A3   65   04E2   913          MULD3   8(r3),8(r3),scratch2
55   00000020'EF   00000010'EF   61   04EC   914          ADDD3   scratch1,scratch2,r5
                          55   73   04F8   915          TSTD    r5
                          4C   13   04FA   916          BEQL    div_dc$
       00000010'EF   62   63   65   04FC   917          MULD3   (r3),(r2),scratch1    ; Real Part Division
   00000020'EF   08 A2   08 A3   65   0504   918          MULD3   8(r3),8(r2),scratch2  ; C<1> = (A<1>*B<1> + A<2>*B<2>) / X
     00000020'EF   00000010'EF   60   050E   919          ADDD2   scratch1,scratch2
        64   00000020'EF   55   67   0519   920          DIVD3   r5,scratch2,(r4)
     00000010'EF   08 A2   63   65   0521   921          MULD3   (r3),8(r2),scratch1   ; Imaginary Part Division
     00000020'EF   62   08 A3   65   052A   922          MULD3   8(r3),(r2),scratch2   ; C<2> = (A<2>*B<1> - A<1>*B<2>) / X
     00000010'EF   00000020'EF   62   0533   923          SUBD2   scratch2,scratch1
      08 A4   00000010'EF   55   67   053E   924          DIVD3   r5,scratch1,8(r4)
                               04   0547   925          RET
                                    0548   926 div_dc$:
                     FD82   31   0548   927          BRW     div_by_zero
                                    054B   928
                                    054B   929 div_gc_gc:
       00000010'EF   63   63 45FD   054B   930          MULG3   (r3),(r3),scratch1    ; X = B<1>*B<1> + B<2>*B<2>
   00000020'EF   08 A3   08 A3 45FD   0554   931          MULG3   8(r3),8(r3),scratch2
55   00000020'EF   00000010'EF 41FD   055F   932          ADDG3   scratch1,scratch2,r5
                000005C7'EF 53FD   056C   933          TSTG    div_gc$
       00000010'EF   62   63 45FD   0573   934          MULG3   (r3),(r2),scratch1    ; Real Part Division
   00000020'EF   08 A2   08 A3 45FD   057C   935          MULG3   8(r3),8(r2),scratch2  ; C<1> = (A<1>*B<1> + A<2>*B<2>) / X
     00000020'EF   00000010'EF 40FD   0587   936          ADDG2   scratch1,scratch2
        64   00000020'EF   55 47FD   0593   937          DIVG3   r5,scratch2,(r4)
     00000010'EF   08 A2   63 45FD   059C   938          MULG3   (r3),8(r2),scratch1   ; Imaginary Part Division
     00000020'EF   62   08 A3 45FD   05A6   939          MULG3   8(r3),(r2),scratch2   ; C<2> = (A<2>*B<1> - A<1>*B<2>) / X
     00000010'EF   00000020'EF 42FD   05B0   940          SUBG2   scratch2,scratch1
      08 A4   00000010'EF   55 47FD   05BC   941          DIVG3   r5,scratch1,8(r4)
                               04   05C6   942          RET
                                    05C7   943 div_gc$:
                     FD03   31   05C7   944          BRW     div_by_zero
                                    05CA   945
                                    05CA   946 div_hc_hc:
```

```
                00000010'EF   63    63 65FD  05CA   947          MULH3   (r3),(r3),scratch1      ; X = B<1>*B<1> + B<2>*B<2>
           00000020'EF   10 A3    10 A3 65FD  05D3   948          MULH3   16(r3),16(r3),scratch2
55            00000020'EF    00000010'EF 61FD  05DE   949          ADDH3   scratch1,scratch2,r5
                             00000646'EF 73FD  05EB   950          TSTH    div_hc$
                00000010'EF   62    63 65FD  05F2   951          MULH3   (r3),(r2),scratch1      ; Real Part Division
           00000020'EF   10 A2    10 A3 65FD  05FB   952          MULH3   16(r3),16(r2),scratch2  ; C<1> = (A<1>*B<1> + A<2>*B<2>) / X
             00000020'EF    00000010'EF 60FD  0606   953          ADDH2   scratch1,scratch2
                64    00000020'EF    55 67FD  0612   954          DIVH3   r5,scratch2,(r4)
              00000010'EF   10 A2    63 65FD  061B   955          MULH3   (r3),16(r2),scratch1    ; Imaginary Part Division
              00000020'EF   62    10 A3 65FD  0625   956          MULH3   16(r3),(r2),scratch2    ; C<2> = (A<2>*B<1> - A<1>*B<2>) / X
             00000010'EF    00000020'EF 62FD  062F   957          SUBH2   scratch2,scratch1
              10 A4    00000010'EF    55 67FD  063B   958          DIVH3   r5,scratch1,16(r4)
                                      04    0645   959          RET
                                            0646   960 div_hc$:
                                FC84    31  0646   961          BRW     div_by_zero
                                            0649   962
                                            0649   963
                                            0649   964 ; Multiplicative Operators - Multiply
                                            0649   965 ;
                                            0649   966 mul_b_b:
                          64    62    63    85  0649   967          MULB3   (r3),(r2),(r4)
                                      01    1D  064D   968          BVS     mul_b$
                                      04    064F   969          RET
                                            0650   970 mul_b$:
                                FC5D    31  0650   971          BRW     int_overflow
                                            0653   972
                                            0653   973 mul_bu_bu:
                          55    62    9A    0653   974          MOVZBL  (r2),r5                 ; Move to a larger quantity so overflow
                          56    63    9A    0656   975          MOVZBL  (r3),r6                 ; condition can be detected
                          55    56    C4    0659   976          MULL2   r6,r5
                000000FF  8F    55    D1    065C   977          CMPL    r5,#^XFF
                                      04    1A  0663   978          BGTRU   mul_bu$
                          64    55    90    0665   979          MOVB    r5,(r4)
                                      04    0668   980          RET
                                            0669   981 mul_bu$:
                                FC44    31  0669   982          BRW     int_overflow
                                            066C   983
                                            066C   984 mul_w_w:
                       64    62    63    A5  066C   985          MULW3   (r3),(r2),(r4)
                                      01    1D  0670   986          BVS     mul_w$
                                      04    0672   987          RET
                                            0673   988 mul_w$:
                                FC3A    31  0673   989          BRW     int_overflow
                                            0676   990
                                            0676   991 mul_wu_wu:
                          55    62    3C    0676   992          MOVZWL  (r2),r5
                          56    63    3C    0679   993          MOVZWL  (r3),r6
                          55    56    C4    067C   994          MULL2   r6,r5
                0000FFFF  8F    55    D1    067F   995          CMPL    r5,#^XFFFF
                                      04    1A  0686   996          BGTRU   mul_wu$
                          64    55    B0    0688   997          MOVW    r5,(r4)
                                      04    068B   998          RET
                                            068C   999 mul_wu$:
                                FC21    31  068C  1000          BRW     int_overflow
                                            068F  1001
                                            068F  1002 mul_l_l:
                       64    62    63    C5  068F  1003          MULL3   (r3),(r2),(r4)
```

```
                     01   1D  0693  1004          BVS     mul_l$
                          04  0695  1005          RET
                              0696  1006 mul_l$:
                     FC17 31  0696  1007          BRW     int_overflow
                              0699  1008
                              0699  1009 ; (note: in C, the code generated for this operation is much simplified -
                              0699  1010 ;        MULL3.  The reason to do so in here is to detect overflow condition.
                              0699  1011 ;
                              0699  1012 mul_lu_lu:                                ;
     55   00   62   63  7A    0699  1013          EMUL    (r3),(r2),#0,r5          ; Perform extended-precision multiplication
                     63  D5   069E  1014          TSTL    (r3)                     ; Test muptiplier
                     03  18   06A0  1015          BGEQ    positive_mulr            ; Multiplier >= 0
                56   62  C0   06A2  1016          ADDL2   (r2),r6                  ; Fudge, so overflow can be detected
                              06A5  1017 positive_mulr:
                     62  D5   06A5  1018          TSTL    (r2)                     ; Test multiplicand
                     03  18   06A7  1019          BGEQ    positive_muld            ; Multiplicand >= 0
                56   63  C0   06A9  1020          ADDL2   (r3),r6                  ; Fudge, so overflow can be detected
                              06AC  1021 positive_muld:
                     56  D5   06AC  1022          TSTL    r6                       ; Test Result[low]
                     04  12   06AE  1023          BNEQ    mul_lu$                  ; Result[low] not= zero, overflow
                64   55  D0   06B0  1024          MOVL    r5,(r4)                  ; Get Result[high] as the result
                     04  06B3  1025          RET
                              06B4  1026 mul_lu$:
                     FBF9 31  06B4  1027          BRW     int_overflow
                              06B7  1028
                              06B7  1029 mul_f_f:
           64   62   63  45  06B7  1030          MULF3   (r3),(r2),(r4)
                     04  06BB  1031          RET
                              06BC  1032 mul_d_d:
           64   62   63  65  06BC  1033          MULD3   (r3),(r2),(r4)
                     04  06C0  1034          RET
                              06C1  1035 mul_g_g:
           64   62   63 45FD  06C1  1036          MULG3   (r3),(r2),(r4)
                     04  06C6  1037          RET
                              06C7  1038 mul_h_h:
           64   62   63 65FD  06C7  1039          MULH3   (r3),(r2),(r4)
                     04  06CC  1040          RET
                              06CD  1041
                              06CD  1042 mul_fc_fc:
           55   62   63  45  06CD  1043          MULF3   (r3),(r2),r5             ; Real Part Multiply:
      56   04 A2   04 A3  45  06D1  1044          MULF3   4(r3),4(r2),r6           ; C<1> = A<1>*B<1> - A<2>*B<2>
           64   55   56  43  06D7  1045          SUBF3   r6,r5,(r4)               ; Real part result
           55   62   04 A3  45  06DB  1046          MULF3   4(r3),(r2),r5            ; Imaginary Part Multiply:
           56   04 A2   63  45  06E0  1047          MULF3   (r3),4(r2),r6            ; C<2> = A<1>*B<2> + A<2>*B<1>
           04 A4   56   55  41  06E5  1048          ADDF3   r5,r6,4(r4)              ; Imaginary part result
                     04  06EA  1049          RET
                              06EB  1050 mul_dc_dc:
     00000010'EF   62   63  65  06EB  1051          MULD3   (r3),(r2),scratch1       ; Real Part Multiply:
00000020'EF   08 A2   08 A3  65  06F3  1052          MULD3   8(r3),8(r2),scratch2     ; C<1> = A<1>*B<1> - A<2>*B<2>
64 00000010'EF   00000020'EF  63  06FD  1053          SUBD3   scratch2,scratch1,(r4)   ; Real part result
  00000010'EF   62   08 A3  65  0709  1054          MULD3   8(r3),(r2),scratch1      ; Imaginary Part Multiply:
  00000020'EF   08 A2   63  65  0712  1055          MULD3   (r3),8(r2),scratch2      ; C<2> = A<1>*B<2> + A<2>*B<1>
00000020'EF   00000010'EF  61  071B  1056          ADDD3   scratch1,scratch2,8(r4)  ; Imaginary part result
                   08 A4         0726
                     04  0728  1057          RET
                              0729  1058 mul_gc_gc:
     00000010'EF   62   63 45FD  0729  1059          MULG3   (r3),(r2),scratch1       ; Real Part Multiply:
```

```
   00000020'EF    08 A2    08 A3 45FD  0732  1060          MULG3   8(r3),8(r2),scratch2    ; C<1> = A<1>*B<1> - A<2>*B<2>
64   00000010'EF    00000020'EF 43FD  073D  1061          SUBG3   scratch2,scratch1,(r4)  ; Real part result
     00000010'EF    62    08 A3 45FD  074A  1062          MULG3   8(r3),(r2),scratch1     ; Imaginary Part Multiply:
     00000020'EF    08 A2    63 45FD  0754  1063          MULG3   (r3),8(r2),scratch2     ; C<2> = A<1>*B<2> + A<2>*B<1>
   00000020'EF    00000010'EF 41FD  075E  1064          ADDG3   scratch1,scratch2,8(r4) ; Imaginary part result
                       08 A4        076A
                                04  076C  1065          RET
                                    076D  1066 mul_hc_hc:
        00000010'EF    62    63 65FD  076D  1067          MULH3   (r3),(r2),scratch1      ; Real Part Multiply:
   00000020'EF    10 A2    10 A3 65FD  0776  1068          MULH3   16(r3),16(r2),scratch2  ; C<1> = A<1>*B<1> - A<2>*B<2>
64   00000010'EF    00000020'EF 63FD  0781  1069          SUBH3   scratch2,scratch1,(r4)  ; Real part result
     00000010'EF    62    10 A3 65FD  078E  1070          MULH3   16(r3),(r2),scratch1    ; Imaginary Part Multiply:
     00000020'EF    10 A2    63 65FD  0798  1071          MULH3   (r3),16(r2),scratch2    ; C<2> = A<1>*B<2> + A<2>*B<1>
   00000020'EF    00000010'EF 61FD  07A2  1072          ADDH3   scratch1,scratch2,16(r4); Imaginary part result
                       10 A4        07AE
                                04  07B0  1073          RET
                                    07B1  1074
                                    07B1  1075
                                    07B1  1076 ;****************************************************************************
                                    07B1  1077 ;
                                    07B1  1078 ;       M O D U L U S    and    R E M A I N D E R
                                    07B1  1079 ;
                                    07B1  1080 ;****************************************************************************
                                    07B1  1081
                                    07B1  1082
                                    07B1  1083 ; Multiplicative - MOD operation
                                    07B1  1084 ;
                                    07B1  1085 mod_l_l:
64   62   00   00   7A              07B1  1086          EMUL    #0,#0,(r2),(r4)
64   64   64   63   7B              07B6  1087          EDIV    (r3),(r4),(r4),(r4)
                  64   D5          07BB  1088          TSTL    (r4)
                  0B   13          07BD  1089          BEQL    mod_l_ret$
          55   62   63   CD        07BF  1090          XORL3   (r3),(r2),r5
                  55   D5          07C3  1091          TSTL    r5
                  03   18          07C5  1092          BGEQ    mod_l_ret$
               64   63   C0        07C7  1093          ADDL2   (r3),(r4)
                                    07CA  1094 mod_l_ret$:
                                04  07CA  1095          RET
                                    07CB  1096
                                    07CB  1097 mod_lu_lu:
                  63   D5          07CB  1098          TSTL    (r3)
                  0B   18          07CD  1099          BGEQ    mod_lu$
          64   62   63   C3        07CF  1100          SUBL3   (r3),(r2),(r4)
                  0D   1E          07D3  1101          BGEQU   mod_lu_ret$
               64   62   D0        07D5  1102          MOVL    (r2),(r4)
                  08   11          07D8  1103          BRB     mod_lu_ret$
                                    07DA  1104 mod_lu$:
               64   62   D0        07DA  1105          MOVL    (r2),(r4)
64   64   64   63   7B              07DD  1106          EDIV    (r3),(r4),(r4),(r4)
                                    07E2  1107 mod_lu_ret$:
                                04  07E2  1108          RET
                                    07E3  1109
                                    07E3  1110
                                    07E3  1111 ; Remainder Operators
                                    07E3  1112 ;
                                    07E3  1113 rem_l_l:
55   62   00   00   7A              07E3  1114          EMUL    #0,#0,(r2),r5
```

```
                64   55   55   63   7B  07E8  1115          EDIV    (r3),r5,r5,(r4)
                                   04  07ED  1116          RET
                                       07EE  1117
                                       07EE  1118 rem_lu_lu:
                          63   D5   07EE  1119          TSTL    (r3)
                          0B   18   07F0  1120          BGEQ    rem_lu$
                     64   62   63   C3   07F2  1121          SUBL3   (r3),(r2),(r4)
                          0D   1E   07F6  1122          BGEQU   rem_lu_ret$
                     64   62   D0   07F8  1123          MOVL    (r2),(r4)
                          08   11   07FB  1124          BRB     rem_lu_ret$
                                       07FD  1125 rem_lu$:
                     64   62   D0   07FD  1126          MOVL    (r2),(r4)
                64   64   64   63   7B  0800  1127          EDIV    (r3),(r4),(r4),(r4)
                                       0805  1128 rem_lu_ret$:
                                   04  0805  1129          RET
                                       0806  1130
                                       0806  1131
                                       0806  1132 ;*****************************************************************************
                                       0806  1133 ;
                                       0806  1134 ;       S  H  I  F  T
                                       0806  1135 ;
                                       0806  1136 ;*****************************************************************************
                                       0806  1137
                                       0806  1138
                                       0806  1139 ; Shift Operators
                                       0806  1140 ;
                                       0806  1141 shift_left_l_l:                          ; same as shift_left_lu_lu
                     64   62   63   78  0806  1142          ASHL    (r3),(r2),(r4)
                                   04  080A  1143          RET
                                       080B  1144 shift_left_l$:
                          FACD   31   080B  1145          BRW     shift_count_negative
                                       080E  1146
                                       080E  1147 shift_rt_l_l:
                          63   D5   080E  1148          TSTL    (r3)
                          08   19   0810  1149          BLSS    shift_rt_l$
                     53   63   CE   0812  1150          MNEGL   (r3),r3
                64   62   53   78  0815  1151          ASHL    r3,(r2),(r4)
                                   04  0819  1152          RET
                                       081A  1153 shift_rt_l$:
                          FABE   31   081A  1154          BRW     shift_count_negative
                                       081D  1155
                                       081D  1156 shift_rt_lu_lu:
                          63   D5   081D  1157          TSTL    (r3)
                          21   19   081F  1158          BLSS    shift_rt_lu$
                     53   63   CE   0821  1159          MNEGL   (r3),r3
                          53   D5   0824  1160          TSTL    r3
                          16   13   0826  1161          BEQL    set_shift_result
                     55   53   D0   0828  1162          MOVL    r3,r5
                          55   D6   082B  1163          INCL    r5
56   04C4B400 8F   55   78  082D  1164          ASHL    r5,#80000000,r6
          55   62   53   78  0835  1165          ASHL    r3,(r2),r5
          64   55   56   CB  0839  1166          BICL3   r6,r5,(r4)
                                   04  083D  1167          RET
                                       083E  1168 set_shift_result:
                     64   62   D0   083E  1169          MOVL    (r2),(r4)
                                   04  0841  1170          RET
                                       0842  1171 shift_rt_lu$:
```

```
                FA96    31  0842  1172          BRW     shift_count_negative
                            0845  1173
                            0845  1174
                            0845  1175 ;*****************************************************************************
                            0845  1176 ;
                            0845  1177 ;       P  O  W  E  R
                            0845  1178 ;
                            0845  1179 ;*****************************************************************************
                            0845  1180
                            0845  1181
                            0845  1182 ; Power of Operators
                            0845  1183 ;
                            0845  1184 power_w_w:
            7E    63    3C  0845  1185          MOVZWL  (r3),-(SP)
            7E    62    3C  0848  1186          MOVZWL  (r2),-(SP)
00000000'GF    02    FB  084B  1187          CALLS   #2,G^OTS$POWII
            64    50    B0  0852  1188          MOVW    r0,(r4)
                        04  0855  1189          RET
                            0856  1190
                            0856  1191 power_l_l:
               63    DD   0856  1192          PUSHL   (r3)
               62    DD   0858  1193          PUSHL   (r2)
00000000'GF    02    FB   085A  1194          CALLS   #2,G^OTS$POWJJ
            64    50    D0  0861  1195          MOVL    r0,(r4)
                        04  0864  1196          RET
                            0865  1197
                            0865  1198 power_f_l:
            7E    63    D0  0865  1199          MOVL    (r3),-(SP)
            7E    62    50  0868  1200          MOVF    (r2),-(SP)
00000000'GF    02    FB  086B  1201          CALLS   #2,G^OTS$POWRJ
            64    50    50  0872  1202          MOVF    r0,(r4)
                        04  0875  1203          RET
                            0876  1204
                            0876  1205 power_d_l:
            7E    63    D0  0876  1206          MOVL    (r3),-(SP)
            7E    62    70  0879  1207          MOVD    (r2),-(SP)
00000000'GF    03    FB  087C  1208          CALLS   #3,G^OTS$POWDJ
            64    50    70  0883  1209          MOVD    r0,(r4)
                        04  0886  1210          RET
                            0887  1211
                            0887  1212 power_g_l:
            7E    63    D0  0887  1213          MOVL    (r3),-(SP)
            7E    62 50FD   088A  1214          MOVG    (r2),-(SP)
00000000'GF    03    FB  088E  1215          CALLS   #3,G^OTS$POWGJ
            64    50 50FD   0895  1216          MOVG    r0,(r4)
                        04  0899  1217          RET
                            089A  1218
                            089A  1219 power_h_l:
            7E    63    D0  089A  1220          MOVL    (r3),-(SP)
            7E    62 70FD   089D  1221          MOVH    (r2),-(SP)
00000000'GF    05    FB  08A1  1222          CALLS   #5,G^OTS$POWHJ_R3
            64    50 70FD   08A8  1223          MOVH    r0,(r4)
                        04  08AC  1224          RET
                            08AD  1225
                            08AD  1226 ; Note that any time a library routine is called, the imaginary portion ;KG001
                            08AD  1227 ; of the real number must be pushed on the stack before the real part.  ;KG001
                            08AD  1228
```

```
                        08AD  1229 power_fc_l:
        7E  63  D0      08AD  1230          MOVL    (r3),-(SP)
      7E  04 A2  50     08B0  1231          MOVF    4(r2),-(SP)                     ;KG001
        7E  62  50      08B4  1232          MOVF    (r2),-(SP)                      ;KG001
00000000'GF  03  FB     08B7  1233          CALLS   #3,G^OTS$POWCJ
        64  50  50      08BE  1234          MOVF    r0,(r4)
     04 A4  51  50      08C1  1235          MOVF    r1,4(r4)
                04      08C5  1236          RET
                        08C6  1237
                        08C6  1238 power_dc_l:
        7E  63  D0      08C6  1239          MOVL    (r3),-(SP)
      7E  08 A2  70     08C9  1240          MOVD    8(r2),-(SP)                     ;KG001
        7E  62  70      08CD  1241          MOVD    (r2),-(SP)                      ;KG001
00000000'GF  05  FB     08D0  1242          CALLS   #5,G^OTS$POWCDJ_R3
        64  50  70      08D7  1243          MOVD    r0,(r4)
     08 A4  52  70      08DA  1244          MOVD    r2,8(r4)
                04      08DE  1245          RET
                        08DF  1246
                        08DF  1247 power_gc_l:
        7E  63  D0      08DF  1248          MOVL    (r3),-(SP)
      7E  08 A2 50FD    08E2  1249          MOVG    8(r2),-(SP)                     ;KG001
        7E  62 50FD     08E7  1250          MOVG    (r2),-(SP)                      ;KG001
00000000'GF  05  FB     08EB  1251          CALLS   #5,G^OTS$POWCGJ_R3
        64  50 50FD     08F2  1252          MOVG    r0,(r4)
     08 A4  52 50FD     08F6  1253          MOVG    r2,8(r4)
                04      08FB  1254          RET
                        08FC  1255
                        08FC  1256 power_f_f:
        7E  63  50      08FC  1257          MOVF    (r3),-(SP)
        7E  62  50      08FF  1258          MOVF    (r2),-(SP)
00000000'GF  02  FB     0902  1259          CALLS   #2,G^OTS$POWRR
        64  50  50      0909  1260          MOVF    r0,(r4)
                04      090C  1261          RET
                        090D  1262
                        090D  1263 power_d_f:
        7E  63  50      090D  1264          MOVF    (r3),-(SP)
        7E  62  70      0910  1265          MOVD    (r2),-(SP)
00000000'GF  03  FB     0913  1266          CALLS   #3,G^OTS$POWDR
        64  50  70      091A  1267          MOVD    r0,(r4)
                04      091D  1268          RET
                        091E  1269
                        091E  1270 power_f_d:
        7E  63  70      091E  1271          MOVD    (r3),-(SP)
        7E  62  50      0921  1272          MOVF    (r2),-(SP)
00000000'GF  03  FB     0924  1273          CALLS   #3,G^OTS$POWRD
        64  50  70      092B  1274          MOVD    r0,(r4)
                04      092E  1275          RET
                        092F  1276
                        092F  1277 power_d_d:
        7E  63  70      092F  1278          MOVD    (r3),-(SP)
        7E  62  70      0932  1279          MOVD    (r2),-(SP)
00000000'GF  04  FB     0935  1280          CALLS   #4,G^OTS$POWDD
        64  50  70      093C  1281          MOVD    r0,(r4)
                04      093F  1282          RET
                        0940  1283
                        0940  1284 power_g_g:
        7E  63 50FD     0940  1285          MOVG    (r3),-(SP)
```

```
                7E    62 50FD   0944  1286          MOVG    (r2),-(SP)
00000000'       GF    04   FB   0948  1287          CALLS   #4,G^OTS$POWGG
                64    50 50FD   094F  1288          MOVG    r0,(r4)
                         04     0953  1289          RET
                                0954  1290
                                0954  1291 power_h_h:
                7E    63 70FD   0954  1292          MOVH    (r3),-(SP)
                7E    62 70FD   0958  1293          MOVH    (r2),-(SP)
00000000'       GF    08   FB   095C  1294          CALLS   #8,G^OTS$POWHH_R3
                64    50 70FD   0963  1295          MOVH    r0,(r4)
                         04     0967  1296          RET
                                0968  1297
                                0968  1298 power_fc_fc:
             7E   04 A3    50   0968  1299          MOVF    4(r3),-(SP)
                7E    63    50   096C  1300          MOVF    (r3),-(SP)
             7E   04 A2    50   096F  1301          MOVF    4(r2),-(SP)
                   12    13    0973  1302          BEQL    1$                      ; Check if the first operand=0  ;KG001
                7E    62    50   0975  1303          MOVF    (r2),-(SP)
00000000'       GF    04   FB   0978  1304 2$:       CALLS   #4,G^OTS$POWCC                                  ;KG001
                64    50    50   097F  1305          MOVF    r0,(r4)
             04 A4    51    50   0982  1306          MOVF    r1,4(r4)
                         04     0986  1307          RET
                                0987  1308
                7E    62    50   0987  1309 1$:       MOVF    (r2),-(SP)              ; Check the second              ;KG001
                   EC    12    098A  1310          BNEQ    2$                      ; Are both = 0?                 ;KG001
             5E   08 AE    DE   098C  1311          MOVAF   8(SP),SP                ; Yes, get rid of args.         ;KG001
          00028ED8 8F      DD   0990  1312          PUSHL   #dbg$_undexpn           ; Signal Undefined Exp.         ;KG001
00000000'       GF    01   FB   0996  1313          CALLS   #1,G^LIB$SIGNAL         ; Signal it                     ;KG001
                         04     099D  1314          RET                             ; And exit                      ;KG001
                                099E  1315
                                099E  1316 power_dc_dc:
             7E   08 A3    70   099E  1317          MOVD    8(r3),-(SP)             ; Push imaginary part first     ;KG001
                7E    63    70   09A2  1318          MOVD    (r3),-(SP)              ; Push the real part last       ;KG001
             7E   08 A2    70   09A5  1319          MOVD    8(r2),-(SP)                                             ;KG001
                   12    13    09A9  1320          BEQL    1$                      ; Check if first op=0           ;KG001
                7E    62    70   09AB  1321          MOVD    (r2),-(SP)                                              ;KG001
00000000'       GF    08   FB   09AE  1322 2$:       CALLS   #8,G^OTS$POWCDCD_R3                                     ;KG001
                64    50    70   09B5  1323          MOVD    r0,(r4)
             08 A4    52    70   09B8  1324          MOVD    r2,8(r4)
                         04     09BC  1325          RET
                                09BD  1326
                7E    62    70   09BD  1327 1$:       MOVD    (r2),-(SP)              ; Same as F, almost             ;KG001
                   EC    12    09C0  1328          BNEQ    2$                                                      ;KG001
             5E   08 AE    7E   09C2  1329          MOVAD   8(SP),SP                                                ;KG001
          00028ED8 8F      DD   09C6  1330          PUSHL   #dbg$_undexpn                                           ;KG001
00000000'       GF    01   FB   09CC  1331          CALLS   #1,G^LIB$SIGNAL                                         ;KG001
                         04     09D3  1332          RET                                                             ;KG001
                                09D4  1333
                                09D4  1334 power_gc_gc:
             7E   08 A3 50FD   09D4  1335          MOVG    8(r3),-(SP)             ; See comments for FC&DC        ;KG001
                7E    63 50FD   09D9  1336          MOVG    (r3),-(SP)                                              ;KG001
             7E   08 A2 50FD   09DD  1337          MOVG    8(r2),-(SP)                                             ;KG001
                   15    13    09E2  1338          BEQL    1$                                                      ;KG001
                7E    62 50FD   09E4  1339          MOVG    (r2),-(SP)                                              ;KG001
00000000'       GF    08   FB   09E8  1340 2$:       CALLS   #8,G^OTS$POWCGCG_R3                                     ;KG001
                64    50 50FD   09EF  1341          MOVG    r0,(r4)
             08 A4    52 50FD   09F3  1342          MOVG    r2,8(r4)
```

```
                     04  09F8  1343          RET
                         09F9  1344
          7E   62 50FD   09F9  1345 1$:      MOVG    (r2),-(SP)                      ;KG001
               E9 12     09FD  1346          BNEQ    2$                              ;KG001
       5E   08 AE        7E    09FF  1347    MOVAG   8(SP),SP                        ;KG001
    00028ED8 8F          DD    0A03  1348    PUSHL   #dbg$_undexpn                   ;KG001
00000000'GF  01          FB    0A09  1349    CALLS   #1,G^LIB$SIGNAL                 ;KG001
                     04  0A10  1350          RET                                     ;KG001
                         0A11  1351
                         0A11  1352
                         0A11  1353 ;*****************************************************************************
                         0A11  1354 ;
                         0A11  1355 ;       C H A R   AND   B I T   S T R I N G
                         0A11  1356 ;
                         0A11  1357 ;*****************************************************************************
                         0A11  1358
                         0A11  1359
                         0A11  1360 ; Concatenation
                         0A11  1361 ;
                         0A11  1362 concat_t_t:
    00000038'EF          DD    0A11  1363          PUSHL   arg2_desc
    00000030'EF          DD    0A17  1364          PUSHL   arg1_desc
    00000040'EF          DD    0A1D  1365          PUSHL   result_desc
00000000'GF  03          FB    0A23  1366          CALLS   #3,G^STR$CONCAT
          01 50          D1    0A2A  1367          CMPL    r0,#ss$_normal
             01          12    0A2D  1368          BNEQ    concat_t$
                     04  0A2F  1369          RET
                         0A30  1370 concat_t$:
51  00000030'FF         3C    0A30  1371          MOVZWL  @arg1_desc,r1
52  00000038'FF         3C    0A37  1372          MOVZWL  @arg2_desc,r2
53  00000040'FF         3C    0A3E  1373          MOVZWL  @result_desc,r3
          52 51          C0    0A45  1374          ADDL2   r1,r2
          53 52          D1    0A48  1375          CMPL    r2,r3
             03          15    0A4B  1376          BLEQ    concat_ok$
           F899          31    0A4D  1377          BRW     string_truncate
                         0A50  1378 concat_ok$:
                     04  0A50  1379          RET
                         0A51  1380
                         0A51  1381 eql_vt_vt:
          63 62          B1    0A51  1382          CMPW    (r2),(r3)
             0D          13    0A54  1383          BEQL    eql_t_t
    0002866B 8F          DD    0A56  1384          PUSHL   #dbg$_strngpad
00000000'GF  01          FB    0A5C  1385          CALLS   #1,G^LIB$SIGNAL
                         0A63  1386 eql_t_t:
          64 01          D0    0A63  1387          MOVL    #1,(r4)
    00000038'EF          DD    0A66  1388          PUSHL   arg2_desc
    00000030'EF          DD    0A6C  1389          PUSHL   arg1_desc
00000000'GF  02          FB    0A72  1390          CALLS   #2,G^STR$COMPARE
             50          D5    0A79  1391          TSTL    r0
             02          13    0A7B  1392          BEQL    eql_t$
             64          D4    0A7D  1393          CLRL    (r4)
                         0A7F  1394 eql_t$:
           F883          31    0A7F  1395          BRW     branch_ret
                         0A82  1396
                         0A82  1397
                         0A82  1398 geq_vt_vt:
          63 62          B1    0A82  1399          CMPW    (r2),(r3)
```

```
                 0D   13   0A85  1400          BEQL    geq_t_t
          0002866B 8F   DD   0A87  1401          PUSHL   #dbg$_strngpad
  00000000'GF     01   FB   0A8D  1402          CALLS   #1,G^LIB$SIGNAL
                       0A94  1403  geq_t_t:
                 64   01   D0   0A94  1404          MOVL    #1,(r4)
          00000038'EF   DD   0A97  1405          PUSHL   arg2_desc
          00000030'EF   DD   0A9D  1406          PUSHL   arg1_desc
  00000000'GF     02   FB   0AA3  1407          CALLS   #2,G^STR$COMPARE
                 50   D5   0AAA  1408          TSTL    r0
                 02   18   0AAC  1409          BGEQ    geq_t$
                 64   D4   0AAE  1410          CLRL    (r4)
                       0AB0  1411  geq_t$:
                F852  31   0AB0  1412          BRW     branch_ret
                       0AB3  1413
                       0AB3  1414  gtr_vt_vt:
              63 62    B1   0AB3  1415          CMPW    (r2),(r3)
                 0D   13   0AB6  1416          BEQL    gtr_t_t
          0002866B 8F   DD   0AB8  1417          PUSHL   #dbg$_strngpad
  00000000'GF     01   FB   0ABE  1418          CALLS   #1,G^LIB$SIGNAL
                       0AC5  1419  gtr_t_t:
                 64   01   D0   0AC5  1420          MOVL    #1,(r4)
          00000038'EF   DD   0AC8  1421          PUSHL   arg2_desc
          00000030'EF   DD   0ACE  1422          PUSHL   arg1_desc
  00000000'GF     02   FB   0AD4  1423          CALLS   #2,G^STR$COMPARE
                 50   D5   0ADB  1424          TSTL    r0
                 02   14   0ADD  1425          BGTR    gtr_t$
                 64   D4   0ADF  1426          CLRL    (r4)
                       0AE1  1427  gtr_t$:
                F821  31   0AE1  1428          BRW     branch_ret
                       0AE4  1429
                       0AE4  1430
                       0AE4  1431  leq_vt_vt:
              63 62    B1   0AE4  1432          CMPW    (r2),(r3)
                 0D   13   0AE7  1433          BEQL    leq_t_t
          0002866B 8F   DD   0AE9  1434          PUSHL   #dbg$_strngpad
  00000000'GF     01   FB   0AEF  1435          CALLS   #1,G^LIB$SIGNAL
                       0AF6  1436  leq_t_t:
                 64   01   D0   0AF6  1437          MOVL    #1,(r4)
          00000038'EF   DD   0AF9  1438          PUSHL   arg2_desc
          00000030'EF   DD   0AFF  1439          PUSHL   arg1_desc
  00000000'GF     02   FB   0B05  1440          CALLS   #2,G^STR$COMPARE
                 50   D5   0B0C  1441          TSTL    r0
                 02   15   0B0E  1442          BLEQ    leq_t$
                 64   D4   0B10  1443          CLRL    (r4)
                       0B12  1444  leq_t$:
                F7F0  31   0B12  1445          BRW     branch_ret
                       0B15  1446
                       0B15  1447
                       0B15  1448  lss_vt_vt:
              63 62    B1   0B15  1449          CMPW    (r2),(r3)
                 0D   13   0B18  1450          BEQL    lss_t_t
          0002866B 8F   DD   0B1A  1451          PUSHL   #dbg$_strngpad
  00000000'GF     01   FB   0B20  1452          CALLS   #1,G^LIB$SIGNAL
                       0B27  1453  lss_t_t:
                 64   01   D0   0B27  1454          MOVL    #1,(r4)
          00000038'EF   DD   0B2A  1455          PUSHL   arg2_desc
          00000030'EF   DD   0B30  1456          PUSHL   arg1_desc
```

```
00000000'GF    02   FB   0B36   1457          CALLS   #2,G^STR$COMPARE
               50   D5   0B3D   1458          TSTL    r0
               02   19   0B3F   1459          BLSS    lss_t$
               64   D4   0B41   1460          CLRL    (r4)
                         0B43   1461 lss_t$:
             F7BF   31   0B43   1462          BRW     branch_ret
                         0B46   1463
                         0B46   1464
                         0B46   1465 neq_vt_vt:
         63    62   B1   0B46   1466          CMPW    (r2),(r3)
               0D   13   0B49   1467          BEQL    neq_t_t
     0002866B  8F   DD   0B4B   1468          PUSHL   #dbg$_strngpad
00000000'GF    01   FB   0B51   1469          CALLS   #1,G^LIB$SIGNAL
                         0B58   1470 neq_t_t:
               64   D4   0B58   1471          CLRL    (r4)
     00000038'EF    DD   0B5A   1472          PUSHL   arg2_desc
     00000030'EF    DD   0B60   1473          PUSHL   arg1_desc
00000000'GF    02   FB   0B66   1474          CALLS   #2,G^STR$COMPARE
               50   D5   0B6D   1475          TSTL    r0
               03   13   0B6F   1476          BEQL    neq_t$
         64    01   D0   0B71   1477          MOVL    #1,(r4)
                         0B74   1478 neq_t$:
             F78E   31   0B74   1479          BRW     branch_ret
                         0B77   1480
                         0B77   1481
                         0B77   1482 ;*****************************************************************************
                         0B77   1483 ;
                         0B77   1484 ;       E  Q  L    and    N  E  Q
                         0B77   1485 ;
                         0B77   1486 ;*****************************************************************************
                         0B77   1487
                         0B77   1488
                         0B77   1489 ; Equality Operators
                         0B77   1490 ;
                         0B77   1491 eql_b_b:
         64    D4   0B77   1492          CLRL    (r4)
   63    62    91   0B79   1493          CMPB    (r2),(r3)
         03    12   0B7C   1494          BNEQ    eql_b$
   64    01    D0   0B7E   1495          MOVL    #1,(r4)
                         0B81   1496 eql_b$:
     F781    31   0B81   1497          BRW     branch_ret
                         0B84   1498
                         0B84   1499 eql_w_w:
         64    D4   0B84   1500          CLRL    (r4)
   63    62    B1   0B86   1501          CMPW    (r2),(r3)
         03    12   0B89   1502          BNEQ    eql_w$
   64    01    D0   0B8B   1503          MOVL    #1,(r4)
                         0B8E   1504 eql_w$:
     F774    31   0B8E   1505          BRW     branch_ret
                         0B91   1506
                         0B91   1507 eql_l_l:
         64    D4   0B91   1508          CLRL    (r4)
   63    62    D1   0B93   1509          CMPL    (r2),(r3)
         03    12   0B96   1510          BNEQ    eql_l$
   64    01    D0   0B98   1511          MOVL    #1,(r4)
                         0B9B   1512 eql_l$:
     F767    31   0B9B   1513          BRW     branch_ret
```

```
                        0B9E  1514
                        0B9E  1515 eql_f_f:
          64    D4      0B9E  1516          CLRL    (r4)
    63    62    51      0BA0  1517          CMPF    (r2),(r3)
          03    12      0BA3  1518          BNEQ    eql_f$
    64    01    D0      0BA5  1519          MOVL    #1,(r4)
                        0BA8  1520 eql_f$:
       F75A     31      0BA8  1521          BRW     branch_ret
                        0BAB  1522
                        0BAB  1523 eql_d_d:
          64    D4      0BAB  1524          CLRL    (r4)
    63    62    71      0BAD  1525          CMPD    (r2),(r3)
          03    12      0BB0  1526          BNEQ    eql_d$
    64    01    D0      0BB2  1527          MOVL    #1,(r4)
                        0BB5  1528 eql_d$:
       F74D     31      0BB5  1529          BRW     branch_ret
                        0BB8  1530
                        0BB8  1531 eql_g_g:
          64    D4      0BB8  1532          CLRL    (r4)
    63    62 51FD       0BBA  1533          CMPG    (r2),(r3)
          03    12      0BBE  1534          BNEQ    eql_g$
    64    01    D0      0BC0  1535          MOVL    #1,(r4)
                        0BC3  1536 eql_g$:
       F73F     31      0BC3  1537          BRW     branch_ret
                        0BC6  1538
                        0BC6  1539 eql_h_h:
          64    D4      0BC6  1540          CLRL    (r4)
    63    62 71FD       0BC8  1541          CMPH    (r2),(r3)
          03    12      0BCC  1542          BNEQ    eql_h$
    64    01    D0      0BCE  1543          MOVL    #1,(r4)
                        0BD1  1544 eql_h$:
       F731     31      0BD1  1545          BRW     branch_ret
                        0BD4  1546
                        0BD4  1547 eql_fc_fc:
          64    D4      0BD4  1548          CLRL    (r4)
    63    62    51      0BD6  1549          CMPF    (r2),(r3)
          0A    12      0BD9  1550          BNEQ    eql_fc$
04 A3 04 A2    51      0BDB  1551          CMPF    4(r2),4(r3)
          03    12      0BE0  1552          BNEQ    eql_fc$
    64    01    D0      0BE2  1553          MOVL    #1,(r4)
                        0BE5  1554 eql_fc$:
       F71D     31      0BE5  1555          BRW     branch_ret
                        0BE8  1556
                        0BE8  1557 eql_dc_dc:
          64    D4      0BE8  1558          CLRL    (r4)
    63    62    71      0BEA  1559          CMPD    (r2),(r3)
          0A    12      0BED  1560          BNEQ    eql_dc$
08 A3 08 A2    71      0BEF  1561          CMPD    8(r2),8(r3)
          03    12      0BF4  1562          BNEQ    eql_dc$
    64    01    D0      0BF6  1563          MOVL    #1,(r4)
                        0BF9  1564 eql_dc$:
       F709     31      0BF9  1565          BRW     branch_ret
                        0BFC  1566
                        0BFC  1567 eql_gc_gc:
          64    D4      0BFC  1568          CLRL    (r4)
    63    62 51FD       0BFE  1569          CMPG    (r2),(r3)
          0B    12      0C02  1570          BNEQ    eql_gc$
```

```
08 A3   08 A2 51FD   0C04  1571          CMPG    8(r2),8(r3)
              03   12   0C0A  1572          BNEQ    eql_gc$
        64    01   D0   0C0C  1573          MOVL    #1,(r4)
                         0C0F  1574 eql_gc$:
           F6F3   31   0C0F  1575          BRW     branch_ret
                         0C12  1576
                         0C12  1577 eql_hc_hc:
        64    D4   0C12  1578          CLRL    (r4)
     63 62 71FD   0C14  1579          CMPH    (r2),(r3)
              0B   12   0C18  1580          BNEQ    eql_hc$
10 A3   10 A2 71FD   0C1A  1581          CMPH    16(r2),16(r3)
              03   12   0C20  1582          BNEQ    eql_hc$
        64    01   D0   0C22  1583          MOVL    #1,(r4)
                         0C25  1584 eql_hc$:
           F6DD   31   0C25  1585          BRW     branch_ret
                         0C28  1586
                         0C28  1587
                         0C28  1588 neq_b_b:
        64    D4   0C28  1589          CLRL    (r4)
     63 62    91   0C2A  1590          CMPB    (r2),(r3)
              03   13   0C2D  1591          BEQL    neq_b$
        64    01   D0   0C2F  1592          MOVL    #1,(r4)
                         0C32  1593 neq_b$:
           F6D0   31   0C32  1594          BRW     branch_ret
                         0C35  1595
                         0C35  1596 neq_w_w:
        64    D4   0C35  1597          CLRL    (r4)
     63 62    B1   0C37  1598          CMPW    (r2),(r3)
              03   13   0C3A  1599          BEQL    neq_w$
        64    01   D0   0C3C  1600          MOVL    #1,(r4)
                         0C3F  1601 neq_w$:
           F6C3   31   0C3F  1602          BRW     branch_ret
                         0C42  1603
                         0C42  1604 neq_l_l:
        64    D4   0C42  1605          CLRL    (r4)
     63 62    D1   0C44  1606          CMPL    (r2),(r3)
              03   13   0C47  1607          BEQL    neq_l$
        64    01   D0   0C49  1608          MOVL    #1,(r4)
                         0C4C  1609 neq_l$:
           F6B6   31   0C4C  1610          BRW     branch_ret
                         0C4F  1611
                         0C4F  1612 neq_f_f:
        64    D4   0C4F  1613          CLRL    (r4)
     63 62    51   0C51  1614          CMPF    (r2),(r3)
              03   13   0C54  1615          BEQL    neq_f$
        64    01   D0   0C56  1616          MOVL    #1,(r4)
                         0C59  1617 neq_f$:
           F6A9   31   0C59  1618          BRW     branch_ret
                         0C5C  1619
                         0C5C  1620 neq_d_d:
        64    D4   0C5C  1621          CLRL    (r4)
     63 62    71   0C5E  1622          CMPD    (r2),(r3)
              03   13   0C61  1623          BEQL    neq_d$
        64    01   D0   0C63  1624          MOVL    #1,(r4)
                         0C66  1625 neq_d$:
           F69C   31   0C66  1626          BRW     branch_ret
                         0C69  1627
```

```
                        0C69  1628 neq_g_g:
            64   D4     0C69  1629          CLRL    (r4)
       63   62 51FD     0C6B  1630          CMPG    (r2),(r3)
            03   13     0C6F  1631          BEQL    neq_g$
       64   01   D0     0C71  1632          MOVL    #1,(r4)
                        0C74  1633 neq_g$:
          F68E   31     0C74  1634          BRW     branch_ret
                        0C77  1635
                        0C77  1636 neq_h_h:
            64   D4     0C77  1637          CLRL    (r4)
       63   62 71FD     0C79  1638          CMPH    (r2),(r3)
            03   13     0C7D  1639          BEQL    neq_h$
       64   01   D0     0C7F  1640          MOVL    #1,(r4)
                        0C82  1641 neq_h$:
          F680   31     0C82  1642          BRW     branch_ret
                        0C85  1643
                        0C85  1644 neq_fc_fc:
       63   62   51     0C85  1645          CMPF    (r2),(r3)
            02   12     0C88  1646          BNEQ    test_imaginaryf
            64   D4     0C8A  1647          CLRL    (r4)
                        0C8C  1648 test_imaginaryf:
04 A3  04 A2   51     0C8C  1649          CMPF    4(r2),4(r3)
            03   13     0C91  1650          BEQL    neq_fc$
       64   01   D0     0C93  1651          MOVL    #1,(r4)
                        0C96  1652 neq_fc$:
          F66C   31     0C96  1653          BRW     branch_ret
                        0C99  1654
                        0C99  1655 neq_dc_dc:
       63   62   71     0C99  1656          CMPD    (r2),(r3)
            02   12     0C9C  1657          BNEQ    test_imaginaryd
            64   D4     0C9E  1658          CLRL    (r4)
                        0CA0  1659 test_imaginaryd:
08 A3  08 A2   71     0CA0  1660          CMPD    8(r2),8(r3)
            03   13     0CA5  1661          BEQL    neq_dc$
       64   01   D0     0CA7  1662          MOVL    #1,(r4)
                        0CAA  1663 neq_dc$:
          F658   31     0CAA  1664          BRW     branch_ret
                        0CAD  1665
                        0CAD  1666 neq_gc_gc:
       63   62 51FD     0CAD  1667          CMPG    (r2),(r3)
            02   12     0CB1  1668          BNEQ    test_imaginaryg
            64   D4     0CB3  1669          CLRL    (r4)
                        0CB5  1670 test_imaginaryg:
08 A3  08 A2 51FD     0CB5  1671          CMPG    8(r2),8(r3)
            03   13     0CBB  1672          BEQL    neq_gc$
       64   01   D0     0CBD  1673          MOVL    #1,(r4)
                        0CC0  1674 neq_gc$:
          F642   31     0CC0  1675          BRW     branch_ret
                        0CC3  1676
                        0CC3  1677 neq_hc_hc:
       63   62 71FD     0CC3  1678          CMPH    (r2),(r3)
            02   12     0CC7  1679          BNEQ    test_imaginaryh
            64   D4     0CC9  1680          CLRL    (r4)
                        0CCB  1681 test_imaginaryh:
10 A3  10 A2 71FD     0CCB  1682          CMPH    16(r2),16(r3)
            03   13     0CD1  1683          BEQL    neq_hc$
       64   01   D0     0CD3  1684          MOVL    #1,(r4)
```

```
                     0CD6  1685 neq_hc$:
    F62C    31       0CD6  1686         BRW     branch_ret
                     0CD9  1687
                     0CD9  1688
                     0CD9  1689 ;*****************************************************************************
                     0CD9  1690 ;
                     0CD9  1691 ;       G  E  Q    and    G  T  R    and    L  E  Q    and     L  S  S
                     0CD9  1692 ;
                     0CD9  1693 ;*****************************************************************************
                     0CD9  1694
                     0CD9  1695
                     0CD9  1696 ; Relational Operators
                     0CD9  1697 ;
                     0CD9  1698 geq_b_b:
64   01    D0       0CD9  1699         MOVL    #1,(r4)
63   62    91       0CDC  1700         CMPB    (r2),(r3)
     02    18       0CDF  1701         BGEQ    geq_b$
     64    D4       0CE1  1702         CLRL    (r4)
                     0CE3  1703 geq_b$:
    F61F    31       0CE3  1704         BRW     branch_ret
                     0CE6  1705
                     0CE6  1706 geq_w_w:
64   01    D0       0CE6  1707         MOVL    #1,(r4)
63   62    B1       0CE9  1708         CMPW    (r2),(r3)
     02    18       0CEC  1709         BGEQ    geq_w$
     64    D4       0CEE  1710         CLRL    (r4)
                     0CF0  1711 geq_w$:
    F612    31       0CF0  1712         BRW     branch_ret
                     0CF3  1713
                     0CF3  1714 geq_l_l:
64   01    D0       0CF3  1715         MOVL    #1,(r4)
63   62    D1       0CF6  1716         CMPL    (r2),(r3)
     02    18       0CF9  1717         BGEQ    geq_l$
     64    D4       0CFB  1718         CLRL    (r4)
                     0CFD  1719 geq_l$:
    F605    31       0CFD  1720         BRW     branch_ret
                     0D00  1721
                     0D00  1722 geq_lu_lu:
64   01    D0       0D00  1723         MOVL    #1,(r4)
63   62    D1       0D03  1724         CMPL    (r2),(r3)
     02    1E       0D06  1725         BGEQU   geq_lu$
     64    D4       0D08  1726         CLRL    (r4)
                     0D0A  1727 geq_lu$:
    F5F8    31       0D0A  1728         BRW     branch_ret
                     0D0D  1729
                     0D0D  1730 geq_f_f:
64   01    D0       0D0D  1731         MOVL    #1,(r4)
63   62    51       0D10  1732         CMPF    (r2),(r3)
     02    18       0D13  1733         BGEQ    geq_f$
     64    D4       0D15  1734         CLRL    (r4)
                     0D17  1735 geq_f$:
    F5EB    31       0D17  1736         BRW     branch_ret
                     0D1A  1737
                     0D1A  1738 geq_d_d:
64   01    D0       0D1A  1739         MOVL    #1,(r4)
63   62    71       0D1D  1740         CMPD    (r2),(r3)
     02    18       0D20  1741         BGEQ    geq_d$
```

```
                64    D4   0D22  1742             CLRL    (r4)
                           0D24  1743 geq_d$:
              F5DE    31   0D24  1744             BRW     branch_ret
                           0D27  1745
                           0D27  1746 geq_g_g:
          64    01    D0   0D27  1747             MOVL    #1,(r4)
          63    62 51FD    0D2A  1748             CMPG    (r2),(r3)
                02    18   0D2E  1749             BGEQ    geq_g$
                64    D4   0D30  1750             CLRL    (r4)
                           0D32  1751 geq_g$:
              F5D0    31   0D32  1752             BRW     branch_ret
                           0D35  1753
                           0D35  1754 geq_h_h:
          64    01    D0   0D35  1755             MOVL    #1,(r4)
          63    62 71FD    0D38  1756             CMPH    (r2),(r3)
                02    18   0D3C  1757             BGEQ    geq_h$
                64    D4   0D3E  1758             CLRL    (r4)
                           0D40  1759 geq_h$:
              F5C2    31   0D40  1760             BRW     branch_ret
                           0D43  1761
                           0D43  1762
                           0D43  1763 gtr_b_b:
          64    01    D0   0D43  1764             MOVL    #1,(r4)
          63    62    91   0D46  1765             CMPB    (r2),(r3)
                02    14   0D49  1766             BGTR    gtr_b$
                64    D4   0D4B  1767             CLRL    (r4)
                           0D4D  1768 gtr_b$:
              F5B5    31   0D4D  1769             BRW     branch_ret
                           0D50  1770
                           0D50  1771 gtr_w_w:
          64    01    D0   0D50  1772             MOVL    #1,(r4)
          63    62    B1   0D53  1773             CMPW    (r2),(r3)
                02    14   0D56  1774             BGTR    gtr_w$
                64    D4   0D58  1775             CLRL    (r4)
                           0D5A  1776 gtr_w$:
              F5A8    31   0D5A  1777             BRW     branch_ret
                           0D5D  1778
                           0D5D  1779 gtr_l_l:
          64    01    D0   0D5D  1780             MOVL    #1,(r4)
          63    62    D1   0D60  1781             CMPL    (r2),(r3)
                02    14   0D63  1782             BGTR    gtr_l$
                64    D4   0D65  1783             CLRL    (r4)
                           0D67  1784 gtr_l$:
              F59B    31   0D67  1785             BRW     branch_ret
                           0D6A  1786
                           0D6A  1787 gtr_lu_lu:
          64    01    D0   0D6A  1788             MOVL    #1,(r4)
          63    62    D1   0D6D  1789             CMPL    (r2),(r3)
                02    1A   0D70  1790             BGTRU   gtr_lu$
                64    D4   0D72  1791             CLRL    (r4)
                           0D74  1792 gtr_lu$:
              F58E    31   0D74  1793             BRW     branch_ret
                           0D77  1794
                           0D77  1795 gtr_f_f:
          64    01    D0   0D77  1796             MOVL    #1,(r4)
          63    62    51   0D7A  1797             CMPF    (r2),(r3)
                02    14   0D7D  1798             BGTR    gtr_f$
```

```
                64   D4   0D7F  1799          CLRL    (r4)
                          0D81  1800 gtr_f$:
              F581   31   0D81  1801          BRW     branch_ret
                          0D84  1802
                          0D84  1803 gtr_d_d:
64    01   D0     0D84  1804          MOVL    #1,(r4)
63    62   71     0D87  1805          CMPD    (r2),(r3)
      02   14     0D8A  1806          BGTR    gtr_d$
      64   D4     0D8C  1807          CLRL    (r4)
                          0D8E  1808 gtr_d$:
              F574   31   0D8E  1809          BRW     branch_ret
                          0D91  1810
                          0D91  1811 gtr_g_g:
64    01   D0     0D91  1812          MOVL    #1,(r4)
63    62 51FD     0D94  1813          CMPG    (r2),(r3)
      02   14     0D98  1814          BGTR    gtr_g$
      64   D4     0D9A  1815          CLRL    (r4)
                          0D9C  1816 gtr_g$:
              F566   31   0D9C  1817          BRW     branch_ret
                          0D9F  1818
                          0D9F  1819 gtr_h_h:
64    01   D0     0D9F  1820          MOVL    #1,(r4)
63    62 71FD     0DA2  1821          CMPH    (r2),(r3)
      02   14     0DA6  1822          BGTR    gtr_h$
      64   D4     0DA8  1823          CLRL    (r4)
                          0DAA  1824 gtr_h$:
              F558   31   0DAA  1825          BRW     branch_ret
                          0DAD  1826
                          0DAD  1827
                          0DAD  1828 leq_b_b:
64    01   D0     0DAD  1829          MOVL    #1,(r4)
63    62   91     0DB0  1830          CMPB    (r2),(r3)
      02   15     0DB3  1831          BLEQ    leq_b$
      64   D4     0DB5  1832          CLRL    (r4)
                          0DB7  1833 leq_b$:
              F54B   31   0DB7  1834          BRW     branch_ret
                          0DBA  1835
                          0DBA  1836 leq_w_w:
64    01   D0     0DBA  1837          MOVL    #1,(r4)
63    62   B1     0DBD  1838          CMPW    (r2),(r3)
      02   15     0DC0  1839          BLEQ    leq_w$
      64   D4     0DC2  1840          CLRL    (r4)
                          0DC4  1841 leq_w$:
              F53E   31   0DC4  1842          BRW     branch_ret
                          0DC7  1843
                          0DC7  1844 leq_l_l:
64    01   D0     0DC7  1845          MOVL    #1,(r4)
63    62   D1     0DCA  1846          CMPL    (r2),(r3)
      02   15     0DCD  1847          BLEQ    leq_l$
      64   D4     0DCF  1848          CLRL    (r4)
                          0DD1  1849 leq_l$:
              F531   31   0DD1  1850          BRW     branch_ret
                          0DD4  1851
                          0DD4  1852 leq_lu_lu:
64    01   D0     0DD4  1853          MOVL    #1,(r4)
63    62   D1     0DD7  1854          CMPL    (r2),(r3)
      02   1B     0DDA  1855          BLEQU   leq_lu$
```

```
                64    D4   0DDC  1856                CLRL    (r4)
                           0DDE  1857 leq_lu$:
            F524    31     0DDE  1858                BRW     branch_ret
                           0DE1  1859
                           0DE1  1860 leq_f_f:
64    01    D0           0DE1  1861                MOVL    #1,(r4)
63    62    51           0DE4  1862                CMPF    (r2),(r3)
      02    15           0DE7  1863                BLEQ    leq_f$
      64    D4           0DE9  1864                CLRL    (r4)
                           0DEB  1865 leq_f$:
            F517    31     0DEB  1866                BRW     branch_ret
                           0DEE  1867
                           0DEE  1868 leq_d_d:
64    01    D0           0DEE  1869                MOVL    #1,(r4)
63    62    71           0DF1  1870                CMPD    (r2),(r3)
      02    15           0DF4  1871                BLEQ    leq_d$
      64    D4           0DF6  1872                CLRL    (r4)
                           0DF8  1873 leq_d$:
            F50A    31     0DF8  1874                BRW     branch_ret
                           0DFB  1875
                           0DFB  1876 leq_g_g:
64    01    D0           0DFB  1877                MOVL    #1,(r4)
63    62 51FD            0DFE  1878                CMPG    (r2),(r3)
      02    15           0E02  1879                BLEQ    leq_g$
      64    D4           0E04  1880                CLRL    (r4)
                           0E06  1881 leq_g$:
            F4FC    31     0E06  1882                BRW     branch_ret
                           0E09  1883
                           0E09  1884 leq_h_h:
64    01    D0           0E09  1885                MOVL    #1,(r4)
63    62 71FD            0E0C  1886                CMPH    (r2),(r3)
      02    15           0E10  1887                BLEQ    leq_h$
      64    D4           0E12  1888                CLRL    (r4)
                           0E14  1889 leq_h$:
            F4EE    31     0E14  1890                BRW     branch_ret
                           0E17  1891
                           0E17  1892
                           0E17  1893 lss_b_b:
64    01    D0           0E17  1894                MOVL    #1,(r4)
63    62    91           0E1A  1895                CMPB    (r2),(r3)
      02    19           0E1D  1896                BLSS    lss_b$
      64    D4           0E1F  1897                CLRL    (r4)
                           0E21  1898 lss_b$:
            F4E1    31     0E21  1899                BRW     branch_ret
                           0E24  1900
                           0E24  1901 lss_w_w:
64    01    D0           0E24  1902                MOVL    #1,(r4)
63    62    B1           0E27  1903                CMPW    (r2),(r3)
      02    19           0E2A  1904                BLSS    lss_w$
      64    D4           0E2C  1905                CLRL    (r4)
                           0E2E  1906 lss_w$:
            F4D4    31     0E2E  1907                BRW     branch_ret
                           0E31  1908
                           0E31  1909 lss_l_l:
64    01    D0           0E31  1910                MOVL    #1,(r4)
63    62    D1           0E34  1911                CMPL    (r2),(r3)
      02    19           0E37  1912                BLSS    lss_l$
```

```
                 64   D4   0E39  1913          CLRL    (r4)
                      0E3B  1914 lss_l$:
              F4C7    31   0E3B  1915          BRW     branch_ret
                      0E3E  1916
                      0E3E  1917 lss_lu_lu:
64    01      D0      0E3E  1918          MOVL    #1,(r4)
63    62      D1      0E41  1919          CMPL    (r2),(r3)
      02      1F      0E44  1920          BLSSU   lss_lu$
      64      D4      0E46  1921          CLRL    (r4)
                      0E48  1922 lss_lu$:
              F4BA    31   0E48  1923          BRW     branch_ret
                      0E4B  1924
                      0E4B  1925 lss_f_f:
64    01      D0      0E4B  1926          MOVL    #1,(r4)
63    62      51      0E4E  1927          CMPF    (r2),(r3)
      02      19      0E51  1928          BLSS    lss_f$
      64      D4      0E53  1929          CLRL    (r4)
                      0E55  1930 lss_f$:
              F4AD    31   0E55  1931          BRW     branch_ret
                      0E58  1932
                      0E58  1933 lss_d_d:
64    01      D0      0E58  1934          MOVL    #1,(r4)
63    62      71      0E5B  1935          CMPD    (r2),(r3)
      02      19      0E5E  1936          BLSS    lss_d$
      64      D4      0E60  1937          CLRL    (r4)
                      0E62  1938 lss_d$:
              F4A0    31   0E62  1939          BRW     branch_ret
                      0E65  1940
                      0E65  1941 lss_g_g:
64    01      D0      0E65  1942          MOVL    #1,(r4)
63    62    51FD      0E68  1943          CMPG    (r2),(r3)
      02      19      0E6C  1944          BLSS    lss_g$
      64      D4      0E6E  1945          CLRL    (r4)
                      0E70  1946 lss_g$:
              F492    31   0E70  1947          BRW     branch_ret
                      0E73  1948
                      0E73  1949 lss_h_h:
64    01      D0      0E73  1950          MOVL    #1,(r4)
63    62    71FD      0E76  1951          CMPH    (r2),(r3)
      02      19      0E7A  1952          BLSS    lss_h$
      64      D4      0E7C  1953          CLRL    (r4)
                      0E7E  1954 lss_h$:
              F484    31   0E7E  1955          BRW     branch_ret
                      0E81  1956
                      0E81  1957
                      0E81  1958
                      0E81  1959 ;*****************************************************************************
                      0E81  1960 ;
                      0E81  1961 ;       B I T W I S E    A N D    and    E Q V   and   N O T
                      0E81  1962 ;                         O R      and    X O R   and   I M P
                      0E81  1963 ;
                      0E81  1964 ;*****************************************************************************
                      0E81  1965
                      0E81  1966
                      0E81  1967 ; Bitwise Operators
                      0E81  1968 ;
                      0E81  1969 bit_and_b_b:
```

```
                53   63   92   0E81  1970                MCOMB   (r3),r3                 ; AND
           64   62   53   8B   0E84  1971                BICB3   r3,(r2),(r4)
                     04   0E88  1972                RET
                          0E89  1973 bit_and_w_w:
                53   63   B2   0E89  1974                MCOMW   (r3),r3
           64   62   53   8B   0E8C  1975                BICB3   r3,(r2),(r4)
                     04   0E90  1976                RET
                          0E91  1977 bit_and_l_l:
                53   63   D2   0E91  1978                MCOML   (r3),r3
           64   62   53   CB   0E94  1979                BICL3   r3,(r2),(r4)
                     04   0E98  1980                RET
                          0E99  1981
                          0E99  1982 bit_eqv_b_b:
                53   63   92   0E99  1983                MCOMB   (r3),r3                 ; EQV
           64   62   53   8D   0E9C  1984                XORB3   r3,(r2),(r4)
                     04   0EA0  1985                RET
                          0EA1  1986
                          0EA1  1987 bit_eqv_w_w:
                53   63   B2   0EA1  1988                MCOMW   (r3),r3
           64   62   53   AD   0EA4  1989                XORW3   r3,(r2),(r4)
                     04   0EA8  1990                RET
                          0EA9  1991
                          0EA9  1992 bit_eqv_l_l:
                53   63   D2   0EA9  1993                MCOML   (r3),r3
           64   62   53   CD   0EAC  1994                XORL3   r3,(r2),(r4)
                     04   0EB0  1995                RET
                          0EB1  1996
                          0EB1  1997 bit_not_b:
                64   62   92   0EB1  1998                MCOMB   (r2),(r4)
                     04   0EB4  1999                RET
                          0EB5  2000 bit_not_w:
                64   62   B2   0EB5  2001                MCOMW   (r2),(r4)
                     04   0EB8  2002                RET
                          0EB9  2003 bit_not_l:
                64   62   D2   0EB9  2004                MCOML   (r2),(r4)               ; ~E in C
                     04   0EBC  2005                RET
                          0EBD  2006
                          0EBD  2007 bit_or_b_b:
           64   62   63   89   0EBD  2008                BISB3   (r3),(r2),(r4)         ; OR
                     04   0EC1  2009                RET
                          0EC2  2010 bit_or_w_w:
           64   62   63   A9   0EC2  2011                BISW3   (r3),(r2),(r4)
                     04   0EC6  2012                RET
                          0EC7  2013 bit_or_l_l:
           64   62   63   C9   0EC7  2014                BISL3   (r3),(r2),(r4)
                     04   0ECB  2015                RET
                          0ECC  2016
                          0ECC  2017 bit_xor_b_b:
           64   62   63   8D   0ECC  2018                XORB3   (r3),(r2),(r4)         ; XOR, NEQV
                     04   0ED0  2019                RET
                          0ED1  2020 bit_xor_w_w:
           64   62   63   AD   0ED1  2021                XORW3   (r3),(r2),(r4)
                     04   0ED5  2022                RET
                          0ED6  2023 bit_xor_l_l:
           64   62   63   CD   0ED6  2024                XORL3   (r3),(r2),(r4)
                     04   0EDA  2025                RET
                          0EDB  2026 bit_imp_b_b:
```

```
                52   62   92   0EDB  2027          MCOMB   (r2), r2
           64   63   52   89   0EDE  2028          BISB3   r2, (r3), (r4)
                     04   0EE2  2029          RET
                          0EE3  2030 bit_imp_w_w:
                52   62   B2   0EE3  2031          MCOMW   (r2), r2
           64   63   52   A9   0EE6  2032          BISW3   r2, (r3), (r4)
                     04   0EEA  2033          RET
                          0EEB  2034 bit_imp_l_l:
                52   62   D2   0EEB  2035          MCOML   (r2), r2
           64   63   52   C9   0EEE  2036          BISL3   r2, (r3), (r4)
                     04   0EF2  2037          RET
                          0EF3  2038
                          0EF3  2039
                          0EF3  2040 ;****************************************************************************
                          0EF3  2041 ;
                          0EF3  2042 ;       L  O  G  I  C  A  L    A  N  D    and    N  O  T    and    O  R
                          0EF3  2043 ;
                          0EF3  2044 ;****************************************************************************
                          0EF3  2045
                          0EF3  2046
                          0EF3  2047 ; Logical Operators
                          0EF3  2048 ;
                          0EF3  2049 and_b_b:
                     62   95   0EF3  2050          TSTB    (r2)                    ; AND
                     2C   13   0EF5  2051          BEQL    set_and_result
                     63   95   0EF7  2052          TSTB    (r3)
                     28   13   0EF9  2053          BEQL    set_and_result
                64   01   D0   0EFB  2054          MOVL    #1,(r4)
                     04   0EFE  2055          RET
                          0EFF  2056 and_w_w:
                     62   B5   0EFF  2057          TSTW    (r2)
                     20   13   0F01  2058          BEQL    set_and_result
                     63   B5   0F03  2059          TSTW    (r3)
                     1C   13   0F05  2060          BEQL    set_and_result
                64   01   D0   0F07  2061          MOVL    #1,(r4)
                     04   0F0A  2062          RET
                          0F0B  2063 and_l_l:
                     62   D5   0F0B  2064          TSTL    (r2)
                     14   13   0F0D  2065          BEQL    set_and_result
                     63   D5   0F0F  2066          TSTL    (r3)
                     10   13   0F11  2067          BEQL    set_and_result
                64   01   D0   0F13  2068          MOVL    #1,(r4)
                     04   0F16  2069          RET
                          0F17  2070 and_d_d:
                     62   73   0F17  2071          TSTD    (r2)
                     08   13   0F19  2072          BEQL    set_and_result
                     63   73   0F1B  2073          TSTD    (r3)
                     04   13   0F1D  2074          BEQL    set_and_result
                64   01   D0   0F1F  2075          MOVL    #1,(r4)
                     04   0F22  2076          RET
                          0F23  2077 set_and_result:
                     64   D4   0F23  2078          CLRL    (r4)
                     04   0F25  2079          RET
                          0F26  2080
                          0F26  2081
                          0F26  2082 not_b:
                     62   95   0F26  2083          TSTB    (r2)
```

```
                     18   13   0F28  2084          BEQL    set_not_result
                     64   D4   0F2A  2085          CLRL    (r4)
                          04   0F2C  2086          RET
                               0F2D  2087 not_w:
                     62   B5   0F2D  2088          TSTW    (r2)
                     11   13   0F2F  2089          BEQL    set_not_result
                     64   D4   0F31  2090          CLRL    (r4)
                          04   0F33  2091          RET
                               0F34  2092 not_l:
                     62   D5   0F34  2093          TSTL    (r2)
                     0A   13   0F36  2094          BEQL    set_not_result
                     64   D4   0F38  2095          CLRL    (r4)
                          04   0F3A  2096          RET
                               0F3B  2097 not_d:
                     62   73   0F3B  2098          TSTD    (r2)
                     03   13   0F3D  2099          BEQL    set_not_result
                     64   D4   0F3F  2100          CLRL    (r4)
                          04   0F41  2101          RET
                               0F42  2102 set_not_result:
                64   01   D0   0F42  2103          MOVL    #1,(r4)
                          04   0F45  2104          RET
                               0F46  2105
                               0F46  2106 or_b_b:
                     62   95   0F46  2107          TSTB    (r2)                    ; OR
                     28   12   0F48  2108          BNEQ    set_or_result
                     63   95   0F4A  2109          TSTB    (r3)
                     24   12   0F4C  2110          BNEQ    set_or_result
                     64   D4   0F4E  2111          CLRL    (r4)
                          04   0F50  2112          RET
                               0F51  2113 or_w_w:
                     62   B5   0F51  2114          TSTW    (r2)
                     1D   12   0F53  2115          BNEQ    set_or_result
                     63   B5   0F55  2116          TSTW    (r3)
                     19   12   0F57  2117          BNEQ    set_or_result
                     64   D4   0F59  2118          CLRL    (r4)
                          04   0F5B  2119          RET
                               0F5C  2120 or_l_l:
                     62   D5   0F5C  2121          TSTL    (r2)
                     12   12   0F5E  2122          BNEQ    set_or_result
                     63   D5   0F60  2123          TSTL    (r3)
                     0E   12   0F62  2124          BNEQ    set_or_result
                     64   D4   0F64  2125          CLRL    (r4)
                          04   0F66  2126          RET
                               0F67  2127
                               0F67  2128 or_d_d:
                     62   73   0F67  2129          TSTD    (r2)
                     07   12   0F69  2130          BNEQ    set_or_result
                     63   73   0F6B  2131          TSTD    (r3)
                     03   12   0F6D  2132          BNEQ    set_or_result
                     64   D4   0F6F  2133          CLRL    (r4)
                          04   0F71  2134          RET
                               0F72  2135
                               0F72  2136 set_or_result:
                64   01   D0   0F72  2137          MOVL    #1,(r4)
                          04   0F75  2138          RET
                               0F76  2139
                               0F76  2140 xor_l_l:
```

```
                62   D5   0F76  2141          TSTL    (r2)
                07   13   0F78  2142          BEQL    xor_l_l_1$
                63   D5   0F7A  2143          TSTL    (r3)
                0A   13   0F7C  2144          BEQL    set_xor_result
                64   D4   0F7E  2145          CLRL    (r4)
                     04   0F80  2146          RET
                          0F81  2147 xor_l_l_1$:
                63   D5   0F81  2148          TSTL    (R3)
                03   12   0F83  2149          BNEQ    set_xor_result
                64   D4   0F85  2150          CLRL    (r4)
                     04   0F87  2151          RET
                          0F88  2152 set_xor_result:
             64 01   D0   0F88  2153          MOVL    #1,(r4)
                     04   0F8B  2154          RET
                          0F8C  2155
                          0F8C  2156 ;*****************************************************************************
                          0F8C  2157 ;
                          0F8C  2158 ;       U N A R Y   M I N U S   and   P L U S
                          0F8C  2159 ;
                          0F8C  2160 ;*****************************************************************************
                          0F8C  2161
                          0F8C  2162
                          0F8C  2163 ; Negating Arithmetic
                          0F8C  2164 ;
                          0F8C  2165 unary_minus_b:
             64 62   8E   0F8C  2166          MNEGB   (r2),(r4)
                01   1D   0F8F  2167          BVS     umb$
                     04   0F91  2168          RET
                          0F92  2169 umb$:
                F31B 31   0F92  2170          BRW     int_overflow
                          0F95  2171
                          0F95  2172 unary_minus_w:
             64 62   AE   0F95  2173          MNEGW   (r2),(r4)
                01   1D   0F98  2174          BVS     umw$
                     04   0F9A  2175          RET
                          0F9B  2176 umw$:
                F312 31   0F9B  2177          BRW     int_overflow
                          0F9E  2178
                          0F9E  2179 unary_minus_l:
             64 62   CE   0F9E  2180          MNEGL   (r2),(r4)
                01   1D   0FA1  2181          BVS     uml$
                     04   0FA3  2182          RET
                          0FA4  2183 uml$:
                F309 31   0FA4  2184          BRW     int_overflow
                          0FA7  2185
                          0FA7  2186 unary_minus_lu:
64 0000FFFF 8F   62   C3  0FA7  2187          SUBL3   (r2),#^XFFFF,(r4)       ; The negative of an unsigned quantity
                     04   0FAF  2188          RET                            ; is computed by subtracting its value
                          0FB0  2189                                         ; from 2**32 in C.
                          0FB0  2190 unary_minus_f:
             64 62   52   0FB0  2191          MNEGF   (r2),(r4)
                     04   0FB3  2192          RET
                          0FB4  2193 unary_minus_d:
             64 62   72   0FB4  2194          MNEGD   (r2),(r4)
                     04   0FB7  2195          RET
                          0FB8  2196 unary_minus_g:
             64 62 52FD   0FB8  2197          MNEGG   (r2),(r4)
```

```
                     04  0FBC  2198          RET
                         0FBD  2199 unary_minus_h:
          64    62 72FD  0FBD  2200          MNEGH   (r2),(r4)
                     04  0FC1  2201          RET
                         0FC2  2202
                         0FC2  2203 unary_minus_fc:
          64    62   52  0FC2  2204          MNEGF   (r2),(r4)
    04 A4   04 A2   52  0FC5  2205          MNEGF   4(r2),4(r4)
                     04  0FCA  2206          RET
                         0FCB  2207 unary_minus_dc:
          64    62   72  0FCB  2208          MNEGD   (r2),(r4)
    08 A4   08 A2   72  0FCE  2209          MNEGD   8(r2),8(r4)
                     04  0FD3  2210          RET
                         0FD4  2211 unary_minus_gc:
          64    62 52FD  0FD4  2212          MNEGG   (r2),(r4)
    08 A4   08 A2 52FD  0FD8  2213          MNEGG   8(r2),8(r4)
                     04  0FDE  2214          RET
                         0FDF  2215 unary_minus_hc:
          64    62 72FD  0FDF  2216          MNEGH   (r2),(r4)
    10 A4   10 A2 72FD  0FE3  2217          MNEGH   16(r2),16(r4)
                     04  0FE9  2218          RET
                         0FEA  2219
                         0FEA  2220
                         0FEA  2221 ; Move a scalar quantity
                         0FEA  2222 ;
                         0FEA  2223 unary_plus_b:
          64    62   90  0FEA  2224          MOVB    (r2),(r4)
                     04  0FED  2225          RET
                         0FEE  2226 unary_plus_w:
          64    62   B0  0FEE  2227          MOVW    (r2),(r4)
                     04  0FF1  2228          RET
                         0FF2  2229 unary_plus_l:
          64    62   D0  0FF2  2230          MOVL    (r2),(r4)
                     04  0FF5  2231          RET
                         0FF6  2232
                         0FF6  2233 unary_plus_f:
          64    62   50  0FF6  2234          MOVF    (r2),(r4)
                     04  0FF9  2235          RET
                         0FFA  2236 unary_plus_d:
          64    62   70  0FFA  2237          MOVD    (r2),(r4)
                     04  0FFD  2238          RET
                         0FFE  2239 unary_plus_g:
          64    62 50FD  0FFE  2240          MOVG    (r2),(r4)
                     04  1002  2241          RET
                         1003  2242 unary_plus_h:
          64    62 70FD  1003  2243          MOVH    (r2),(r4)
                     04  1007  2244          RET
                         1008  2245
                         1008  2246 unary_plus_fc:
          64    62   50  1008  2247          MOVF    (r2),(r4)
    04 A4   04 A2   50  100B  2248          MOVF    4(r2),4(r4)
                     04  1010  2249          RET
                         1011  2250 unary_plus_dc:
          64    62   70  1011  2251          MOVD    (r2),(r4)
    08 A4   08 A2   70  1014  2252          MOVD    8(r2),8(r4)
                     04  1019  2253          RET
                         101A  2254 unary_plus_gc:
```

```
                64   62 50FD   101A  2255          MOVG    (r2),(r4)
          08 A4   08 A2 50FD   101E  2256          MOVG    8(r2),8(r4)
                        04     1024  2257          RET
                              1025  2258 unary_plus_hc:
                64   62 70FD   1025  2259          MOVH    (r2),(r4)
          10 A4   10 A2 70FD   1029  2260          MOVH    16(r2),16(r4)
                        04     102F  2261          RET
                              1030  2262
                              1030  2263
                              1030  2264 ;*****************************************************************************
                              1030  2265 ;
                              1030  2266 ; A B S O L U T E   V A L U E
                              1030  2267 ;
                              1030  2268 ;*****************************************************************************
                              1030  2269
                              1030  2270 abs_b:
                     62   95  1030  2271          TSTB    (r2)
                     04   19  1032  2272          BLSS    abs_b_negate
                64   62   90  1034  2273          MOVB    (r2),(r4)
                        04    1037  2274          RET
                              1038  2275 abs_b_negate:
                64   62   8E  1038  2276          MNEGB   (r2),(r4)
                        04    103B  2277          RET
                              103C  2278
                              103C  2279 abs_w:
                     62   B5  103C  2280          TSTW    (r2)
                     04   19  103E  2281          BLSS    abs_w_negate
                64   62   B0  1040  2282          MOVW    (r2),(r4)
                        04    1043  2283          RET
                              1044  2284 abs_w_negate:
                64   62   AE  1044  2285          MNEGW   (r2),(r4)
                        04    1047  2286          RET
                              1048  2287
                              1048  2288 abs_l:
                     62   D5  1048  2289          TSTL    (r2)
                     04   19  104A  2290          BLSS    abs_l_negate
                64   62   D0  104C  2291          MOVL    (r2),(r4)
                        04    104F  2292          RET
                              1050  2293 abs_l_negate:
                64   62   CE  1050  2294          MNEGL   (r2),(r4)
                        04    1053  2295          RET
                              1054  2296
                              1054  2297 abs_f:
                     62   53  1054  2298          TSTF    (r2)
                     04   19  1056  2299          BLSS    abs_f_negate
                64   62   50  1058  2300          MOVF    (r2),(r4)
                        04    105B  2301          RET
                              105C  2302 abs_f_negate:
                64   62   52  105C  2303          MNEGF   (r2),(r4)
                        04    105F  2304          RET
                              1060  2305
                              1060  2306 abs_d:
                     62   73  1060  2307          TSTD    (r2)
                     04   19  1062  2308          BLSS    abs_d_negate
                64   62   70  1064  2309          MOVD    (r2),(r4)
                        04    1067  2310          RET
                              1068  2311 abs_d_negate:
```

```
64   62    72  1068  2312          MNEGD   (r2),(r4)
           04  106B  2313          RET
               106C  2314
               106C  2315  abs_g:
      62 53FD  106C  2316          TSTG    (r2)
      05   19  106F  2317          BLSS    abs_g_negate
64    62 50FD  1071  2318          MOVG    (r2),(r4)
           04  1075  2319          RET
               1076  2320  abs_g_negate:
64    62 52FD  1076  2321          MNEGG   (r2),(r4)
           04  107A  2322          RET
               107B  2323
               107B  2324  abs_h:
      62 73FD  107B  2325          TSTH    (r2)
      B8   19  107E  2326          BLSS    abs_b_negate
64    62 70FD  1080  2327          MOVH    (r2),(r4)
           04  1084  2328          RET
               1085  2329  abs_h_negate:
64    62 72FD  1085  2330          MNEGH   (r2),(r4)
           04  1089  2331          RET
               108A  2332
               108A  2333
               108A  2334
               108A  2335
               108A  2336  ;****************************************************************************
               108A  2337  ;
               108A  2338  ;       D E R E F E R E N C I N G
               108A  2339  ;
               108A  2340  ;****************************************************************************
               108A  2341
               108A  2342
               108A  2343  ; Dereferencing Pointers
               108A  2344  ;
               108A  2345  indirect_lu:
 00000030'EF   DD  108A  2346          PUSHL   arg1_desc
00000000'EF    01 FB 1090  2347          CALLS   #1,dbg$bliss_indirection
      64   50  D0  1097  2348          MOVL    r0,(r4)
           04  109A  2349          RET
               109B  2350
               109B  2351  ; Indirection of a pointer in language C.
               109B  2352  ; We pass in the entire value descriptor (because the indirection
               109B  2353  ; needs to look at the typeid). The routine returns a new descriptor
               109B  2354  ; that represents the result of the indirection. We put a pointer
               109B  2355  ; to this new descriptor back into the RESULT_ADDR output parameter.
               109B  2356  ;
               109B  2357  indirect_tptr:
 00000034'EF   DD  109B  2358          PUSHL   arg1_valdesc
00000000'EF    01 FB 10A1  2359          CALLS   #1,dbg$c_indirection
    14 BC   50  D0  10A8  2360          MOVL    r0,@result_addr(AP)
           04  10AC  2361          RET
               10AD  2362
               10AD  2363
               10AD  2364  ;****************************************************************************
               10AD  2365  ;
               10AD  2366  ;       B I T   S E L E C T I O N
               10AD  2367  ;
               10AD  2368  ;****************************************************************************
```

```
                                 10AD  2369
                                 10AD  2370
                                 10AD  2371 ; BLISS bit-select operator (X<p,s,e>)
                                 10AD  2372 ;
                                 10AD  2373 bitselect:
          00000040'EF    DD      10AD  2374          PUSHL   result_desc
          00000030'EF    DD      10B3  2375          PUSHL   arg1_desc
                04 AC    DD      10B9  2376          PUSHL   operator_entry(AP)
     00000000'EF   03    FB      10BC  2377          CALLS   #3,dbg$bliss_bitselect
                         04      10C3  2378          RET
                                 10C4  2379
                                 10C4  2380
                                 10C4  2381 ;*****************************************************************************
                                 10C4  2382 ;
                                 10C4  2383 ;       S  E  T
                                 10C4  2384 ;
                                 10C4  2385 ;*****************************************************************************
                                 10C4  2386
                                 10C4  2387
                                 10C4  2388 ; Difference of Two Sets
                                 10C4  2389 ;
                                 10C4  2390 difference_set_set:
                    55   D4      10C4  2391          CLRL    r5
                    56   D4      10C6  2392          CLRL    r6
               55   00000030'FF  B0    10C8  2393          MOVW    @arg1_desc,r5
               56   00000038'FF  B0    10CF  2394          MOVW    @arg2_desc,r6
00000040'FF    00000030'FF  B0    10D6  2395          MOVW    @arg1_desc,@result_desc
                    50   54    D0    10E1  2396          MOVL    r4,r0
                                 10E4  2397 difference_set1$:
                80   82   83    8B    10E4  2398          BICB3   (r3)+,(r2)+,(r0)+
                         55    D7    10E8  2399          DECL    r5
                         56    D7    10EA  2400          DECL    r6
                         55    D5    10EC  2401          TSTL    r5
                         06    12    10EE  2402          BNEQ    difference_set2$
                         56    D5    10F0  2403          TSTL    r6
                         10    12    10F2  2404          BNEQ    difference_set4$
                         21    11    10F4  2405          BRB     difference_set_ret$
                                 10F6  2406 difference_set2$:
                         56    D5    10F6  2407          TSTL    r6
                         EA    12    10F8  2408          BNEQ    difference_set1$
                                 10FA  2409 difference_set3$:
              80   82   FF  8F    8B    10FA  2410          BICB3   #^XFF,(r2)+,(r0)+
                        F8 55    F5    10FF  2411          SOBGTR  r5,difference_set3$
                         13    11    1102  2412          BRB     difference_set_ret$
                                 1104  2413 difference_set4$:
              80   FF 8F    83    8B    1104  2414          BICB3   (r3)+,#^XFF,(r0)+
                       F8 56    F5    1109  2415          SOBGTR  r6,difference_set4$
00000040'FF    00000038'FF  B0    110C  2416          MOVW    @arg2_desc,@result_desc
                                 1117  2417 difference_set_ret$:
                         04    1117  2418          RET
                                 1118  2419
                                 1118  2420
                                 1118  2421 ; Set Equal
                                 1118  2422 ;
                                 1118  2423 eql_set_set:
                         64    D4    1118  2424          CLRL    (r4)
                         55    D4    111A  2425          CLRL    r5
```

```
                     56   D4  111C  2426          CLRL    r6
          55 00000030'FF   B0  111E  2427          MOVW    @arg1_desc,r5
          56 00000038'FF   B0  1125  2428          MOVW    @arg2_desc,r6
                           112C  2429 eql_set1$:
                82   83   91  112C  2430          CMPB    (r3)+,(r2)+
                     2D   12  112F  2431          BNEQ    eql_set_ret$
                     55   D7  1131  2432          DECL    r5
                     56   D7  1133  2433          DECL    r6
                     55   D5  1135  2434          TSTL    r5
                     09   12  1137  2435          BNEQ    eql_set2$
                     56   D5  1139  2436          TSTL    r6
                     16   12  113B  2437          BNEQ    eql_set4$
                64   01   D0  113D  2438          MOVL    #1,(r4)
                     1C   11  1140  2439          BRB     eql_set_ret$
                           1142  2440 eql_set2$:
                     56   D5  1142  2441          TSTL    r6
                     E6   12  1144  2442          BNEQ    eql_set1$
                           1146  2443 eql_set3$:
                82   00   91  1146  2444          CMPB    #^X00,(r2)+
                     13   12  1149  2445          BNEQ    eql_set_ret$
                  F8 55   F5  114B  2446          SOBGTR  r5,eql_set3$
                64   01   D0  114E  2447          MOVL    #1,(r4)
                     0B   11  1151  2448          BRB     eql_set_ret$
                           1153  2449 eql_set4$:
                00   83   91  1153  2450          CMPB    (r3)+,#^X00
                     06   12  1156  2451          BNEQ    eql_set_ret$
                  F8 56   F5  1158  2452          SOBGTR  r6,eql_set4$
                64   01   D0  115B  2453          MOVL    #1,(r4)
                           115E  2454 eql_set_ret$:
                     04   115E  2455          RET
                           115F  2456
                           115F  2457
                           115F  2458 ; Set B is a subset of Set A
                           115F  2459 ;
                           115F  2460 geq_set_set:
                     64   D4  115F  2461          CLRL    (r4)
                     55   D4  1161  2462          CLRL    r5
                     56   D4  1163  2463          CLRL    r6
          56 00000030'FF   B0  1165  2464          MOVW    @arg1_desc,r6
          55 00000038'FF   B0  116C  2465          MOVW    @arg2_desc,r5
                50   52   D0  1173  2466          MOVL    r2,r0
                52   53   D0  1176  2467          MOVL    r3,r2
                53   50   D0  1179  2468          MOVL    r0,r3
                     74   11  117C  2469          BRB     leq_set1$
                           117E  2470
                           117E  2471
                           117E  2472 ; IN, C is an element of Set B
                           117E  2473 ;
                           117E  2474 in_set_set:
64   63   01   62   EF  117E  2475          EXTZV   (r2),#1,(r3),(r4)
                     03   13  1183  2476          BEQL    in_set$
                64   01   D0  1185  2477          MOVL    #1,(r4)
                           1188  2478 in_set$:
                     04   1188  2479          RET
                           1189  2480
                           1189  2481
                           1189  2482 ; Intersection of Two Sets
```

```
                                 1189  2483 ;
                                 1189  2484 intersect_set_set:
                          55 D4  1189  2485         CLRL    r5
                          56 D4  118B  2486         CLRL    r6
              55 00000030'FF B0  118D  2487         MOVW    @arg1_desc,r5
              56 00000038'FF B0  1194  2488         MOVW    @arg2_desc,r6
00000040'FF   00000030'FF B0  119B  2489         MOVW    @arg1_desc,@result_desc
                    50 54 D0  11A6  2490         MOVL    r4,r0
                                 11A9  2491 intersect_set1$:
                    51 83 92  11A9  2492         MCOMB   (r3)+,r1
                 80 82 51 8B  11AC  2493         BICB3   r1,(r2)+,(r0)+
                          55 D7  11B0  2494         DECL    r5
                          56 D7  11B2  2495         DECL    r6
                          55 D5  11B4  2496         TSTL    r5
                          06 12  11B6  2497         BNEQ    intersect_set2$
                          56 D5  11B8  2498         TSTL    r6
                          0F 12  11BA  2499         BNEQ    intersect_set4$
                          1F 11  11BC  2500         BRB     intersect_set_ret$
                                 11BE  2501 intersect_set2$:
                          56 D5  11BE  2502         TSTL    r6
                          E7 12  11C0  2503         BNEQ    intersect_set1$
                                 11C2  2504 intersect_set3$:
                 80 82 00 8B  11C2  2505         BICB3   #^X00,(r2)+,(r0)+
                       F9 55 F5  11C6  2506         SOBGTR  r5,intersect_set3$
                          12 11  11C9  2507         BRB     intersect_set_ret$
                                 11CB  2508 intersect_set4$:
                 80 00 83 8B  11CB  2509         BICB3   (r3)+,#^X00,(r0)+
                       F9 56 F5  11CF  2510         SOBGTR  r6,intersect_set4$
00000040'FF   00000038'FF B0  11D2  2511         MOVW    @arg2_desc,@result_desc
                                 11DD  2512 intersect_set_ret$:
                          04 11DD  2513         RET
                                 11DE  2514
                                 11DE  2515
                                 11DE  2516 ; Set A is a subset of Set B
                                 11DE  2517 ;
                                 11DE  2518 leq_set_set:
                          64 D4  11DE  2519         CLRL    (r4)
                          55 D4  11E0  2520         CLRL    r5
                          56 D4  11E2  2521         CLRL    r6
          55 00000030'FF B0  11E4  2522         MOVW    @arg1_desc,r5
          56 00000038'FF B0  11EB  2523         MOVW    @arg2_desc,r6
                                 11F2  2524 leq_set1$:
              50 82 83 8B  11F2  2525         BICB3   (r3)+,(r2)+,r0
                          50 95  11F6  2526         TSTB    r0
                          31 12  11F8  2527         BNEQ    leq_set_ret$
                          55 D7  11FA  2528         DECL    r5
                          56 D7  11FC  2529         DECL    r6
                          55 D5  11FE  2530         TSTL    r5
                          09 12  1200  2531         BNEQ    leq_set2$
                          56 D5  1202  2532         TSTL    r6
                          1A 12  1204  2533         BNEQ    leq_set4$
                       64 01 D0  1206  2534         MOVL    #1,(r4)
                          20 11  1209  2535         BRB     leq_set_ret$
                                 120B  2536 leq_set2$:
                          56 D5  120B  2537         TSTL    r6
                          E3 12  120D  2538         BNEQ    leq_set1$
                                 120F  2539 leq_set3$:
```

```
                50   82    FF 8F    8B  120F  2540          BICB3   #^XFF,(r2)+,r0
                                50    95  1214  2541          TSTB    r0
                                13    12  1216  2542          BNEQ    leq_set_ret$
                              F4 55   F5  1218  2543          SOBGTR  r5,leq_set3$
                           64   01    D0  121B  2544          MOVL    #1,(r4)
                                0B    11  121E  2545          BRB     leq_set_ret$
                                      1220  2546 leq_set4$:
                50   FF 8F    83    8B  1220  2547          BICB3   (r3)+,#^XFF,r0
                              F8 56   F5  1225  2548          SOBGTR  r6,leq_set4$
                           64   01    D0  1228  2549          MOVL    #1,(r4)
                                      122B  2550 leq_set_ret$:
                                04    122B  2551          RET
                                      122C  2552
                                      122C  2553
                                      122C  2554 ; Set Not Equal
                                      122C  2555 ;
                                      122C  2556 neq_set_set:
                                64    D4  122C  2557          CLRL    (r4)
                                55    D4  122E  2558          CLRL    r5
                                56    D4  1230  2559          CLRL    r6
                55   00000030'FF    B0  1232  2560          MOVW    @arg1_desc,r5
                56   00000038'FF    B0  1239  2561          MOVW    @arg2_desc,r6
                                50    D4  1240  2562          CLRL    r0
                                51    D4  1242  2563          CLRL    r1
                           50   55    D0  1244  2564          MOVL    r5,r0
                           56   55    D1  1247  2565          CMPL    r5,r6
                                03    18  124A  2566          BGEQ    neq_set1$
                           50   56    D0  124C  2567          MOVL    r6,r0
                                      124F  2568 neq_set1$:
                82   83    91    124F  2569          CMPB    (r3)+,(r2)+
                     02    12    1252  2570          BNEQ    neq_set1_1$
                     51    D6    1254  2571          INCL    r1
                                      1256  2572 neq_set1_1$:
                     55    D7    1256  2573          DECL    r5
                     56    D7    1258  2574          DECL    r6
                     55    D5    125A  2575          TSTL    r5
                     06    12    125C  2576          BNEQ    neq_set2$
                     56    D5    125E  2577          TSTL    r6
                     12    12    1260  2578          BNEQ    neq_set4$
                     1A    11    1262  2579          BRB     neq_set$
                                1264  2580 neq_set2$:
                     56    D5    1264  2581          TSTL    r6
                     E7    12    1266  2582          BNEQ    neq_set1$
                                1268  2583 neq_set3$:
                82   00    91    1268  2584          CMPB    #^X00,(r2)+
                     02    12    126B  2585          BNEQ    neq_set3_1$
                     51    D6    126D  2586          INCL    r1
                                1270  2587 neq_set3_1$:
                  F6 55    F5    126F  2588          SOBGTR  r5,neq_set3$
                     0A    11    1272  2589          BRB     neq_set$
                                1274  2590 neq_set4$:
                00   83    91    1274  2591          CMPB    (r3)+,#^X00
                     02    12    1277  2592          BNEQ    neq_set4_1$
                     51    D6    1279  2593          INCL    r1
                                127B  2594 neq_set4_1$:
                  F6 56    F5    127B  2595          SOBGTR  r6,neq_set4$
                                127E  2596 neq_set$:
```

```
                    51    50    D1   127E  2597          CMPL    r0,r1
                          03    13   1281  2598          BEQL    neq_set_ret$
                    64    01    D0   1283  2599          MOVL    #1,(r4)
                                     1286  2600 neq_set_ret$:
                                04   1286  2601          RET
                                     1287  2602
                                     1287  2603
                                     1287  2604 ; Union of Two Sets
                                     1287  2605 ;
                                     1287  2606 union_set_set:
                          55    D4   1287  2607          CLRL    r5
                          56    D4   1289  2608          CLRL    r6
               55   00000030'FF    B0   128B  2609          MOVW    @arg1_desc,r5
               56   00000038'FF    B0   1292  2610          MOVW    @arg2_desc,r6
00000040'FF     00000030'FF    B0   1299  2611          MOVW    @arg1_desc,@result_desc
                    50    54    D0   12A4  2612          MOVL    r4,r0
                                     12A7  2613 union_set1$:
                80  82    83    89   12A7  2614          BISB3   (r3)+,(r2)+,(r0)+
                          55    D7   12AB  2615          DECL    r5
                          56    D7   12AD  2616          DECL    r6
                          55    D5   12AF  2617          TSTL    r5
                          06    12   12B1  2618          BNEQ    union_set2$
                          56    D5   12B3  2619          TSTL    r6
                          0E    12   12B5  2620          BNEQ    union_set4$
                          1D    11   12B7  2621          BRB     union_set_ret$
                                     12B9  2622 union_set2$:
                          56    D5   12B9  2623          TSTL    r6
                          EA    12   12BB  2624          BNEQ    union_set1$
                                     12BD  2625 union_set3$:
                80  82    90   12BD  2626          MOVB    (r2)+,(r0)+
                  FA 55    F5   12C0  2627          SOBGTR  r5,union_set3$
                       11    11   12C3  2628          BRB     union_set_ret$
                                     12C5  2629 union_set4$:
                80  83    90   12C5  2630          MOVB    (r3)+,(r0)+
                  FA 56    F5   12C8  2631          SOBGTR  r6,union_set4$
00000040'FF     00000038'FF    B0   12CB  2632          MOVW    @arg2_desc,@result_desc
                                     12D6  2633 union_set_ret$:
                                04   12D6  2634          RET
                                     12D7  2635
                                     12D7  2636 ; C address-of operator.
                                     12D7  2637 ; We pass in the entire value descriptor to the C_ADDRESS_OF routine,
                                     12D7  2638 ; because it will need to look at the typeid. We also pass in a
                                     12D7  2639 ; pointer to the result value descriptor. This routine will fill
                                     12D7  2640 ; in the result value descriptor.
                                     12D7  2641 ;
                                     12D7  2642 address_l:
          00000044'EF    DD   12D7  2643          PUSHL   result_valdesc
          00000034'EF    DD   12DD  2644          PUSHL   arg1_valdesc
     00000000'EF    02    FB   12E3  2645          CALLS   #2,dbg$c_address_of
                                04   12EA  2646          RET
                                     12EB  2647
                                     12EB  2648
                                     12EB  2649 ; C SIZEOF operator
                                     12EB  2650 ; Pass in the entire value descriptor to the C_SIZEOF routine. (It will
                                     12EB  2651 ; have to look at the typeid.) This routine will return the size in R0.
                                     12EB  2652 ;
                                     12EB  2653 sizeof_l:
```

```
                00000034'EF   DD  12EB  2654          PUSHL   arg1_valdesc
          00000000'EF   01    FB  12F1  2655          CALLS   #1,dbg$c_sizeof
                  64   50     D0  12F8  2656          MOVL    r0,(r4)
                              04  12FB  2657          RET
                                  12FC  2658
                                  12FC  2659 ; Addition of typed pointer and integer in C.
                                  12FC  2660 ;
                                  12FC  2661 add_tptr_l:
                00000044'EF   DD  12FC  2662          PUSHL   result_valdesc
                0000003C'EF   DD  1302  2663          PUSHL   arg2_valdesc
                00000034'EF   DD  1308  2664          PUSHL   arg1_valdesc
          00000000'EF   03    FB  130E  2665          CALLS   #3,dbg$c_add_tptr_l
                              04  1315  2666          RET
                                  1316  2667
                                  1316  2668 ; Subtraction of typed pointer and integer in C.
                                  1316  2669 ;
                                  1316  2670 sub_tptr_l:
                00000044'EF   DD  1316  2671          PUSHL   result_valdesc
                0000003C'EF   DD  131C  2672          PUSHL   arg2_valdesc
                00000034'EF   DD  1322  2673          PUSHL   arg1_valdesc
          00000000'EF   03    FB  1328  2674          CALLS   #3,dbg$c_sub_tptr_l
                              04  132F  2675          RET
                                  1330  2676
                                  1330  2677 ; Subtraction of two typed pointers in C.
                                  1330  2678 ;
                                  1330  2679 sub_tptr_tptr:
                00000044'EF   DD  1330  2680          PUSHL   result_valdesc
                0000003C'EF   DD  1336  2681          PUSHL   arg2_valdesc
                00000034'EF   DD  133C  2682          PUSHL   arg1_valdesc
          00000000'EF   03    FB  1342  2683          CALLS   #3,dbg$c_sub_tptr_tptr
                              04  1349  2684          RET
                                  134A  2685
                                  134A  2686 ; increment and decrement in C (++X  X++  --X  X--)
                                  134A  2687 ;
                                  134A  2688 pre_incr_l:
                                  134A  2689 pre_incr_lu:
             00028723   8F    DD  134A  2690          PUSHL   #dbg$_cpreincr
          00000000'GF   01    FB  1350  2691          CALLS   #1,G^lib$signal
                  64   62     D0  1357  2692          MOVL    (r2),(r4)
                  64   D6     D6  135A  2693          INCL    (r4)
                              04  135C  2694          RET
                                  135D  2695
                                  135D  2696 pre_incr_d:
             00028723   8F    DD  135D  2697          PUSHL   #dbg$_cpreincr
          00000000'GF   01    FB  1363  2698          CALLS   #1,G^lib$signal
                  64   62     70  136A  2699          MOVD    (r2),(r4)
                  64   08     60  136D  2700          ADDD2   #1.,(r4)
                              04  1370  2701          RET
                                  1371  2702
                                  1371  2703 pre_incr_tptr:
             00028723   8F    DD  1371  2704          PUSHL   #dbg$_cpreincr
          00000000'GF   01    FB  1377  2705          CALLS   #1,G^lib$signal
                00000034'EF   DD  137E  2706          PUSHL   arg1_valdesc
          00000000'EF   01    FB  1384  2707          CALLS   #1,dbg$c_pre_incr_tptr
                  64   50     D0  138B  2708          MOVL    r0,(r4)
                              04  138E  2709          RET
                                  138F  2710
```

```
                     138F   2711 post_incr_l:
                     138F   2712 post_incr_lu:
                     138F   2713 post_incr_tptr:
     0002872B 8F     DD     138F   2714          PUSHL   #dbg$_cpostincr
00000000'GF   01     FB     1395   2715          CALLS   #1,G^lib$signal
        64   62     D0     139C   2716          MOVL    (r2),(r4)
                    04     139F   2717          RET
                     13A0   2718
                     13A0   2719 post_incr_d:
     0002872B 8F     DD     13A0   2720          PUSHL   #dbg$_cpostincr
00000000'GF   01     FB     13A6   2721          CALLS   #1,G^lib$signal
        64   62     70     13AD   2722          MOVD    (r2),(r4)
                    04     13B0   2723          RET
                     13B1   2724
                     13B1   2725 pre_decr_l:
                     13B1   2726 pre_decr_lu:
     00028733 8F     DD     13B1   2727          PUSHL   #dbg$_cpredecr
00000000'GF   01     FB     13B7   2728          CALLS   #1,G^lib$signal
        64   62     D0     13BE   2729          MOVL    (r2),(r4)
             64     D7     13C1   2730          DECL    (r4)
                    04     13C3   2731          RET
                     13C4   2732
                     13C4   2733 pre_decr_d:
     00028733 8F     DD     13C4   2734          PUSHL   #dbg$_cpredecr
00000000'GF   01     FB     13CA   2735          CALLS   #1,G^lib$signal
        64   62     70     13D1   2736          MOVD    (r2),(r4)
        64   08     62     13D4   2737          SUBD2   #1.,(r4)
                    04     13D7   2738          RET
                     13D8   2739
                     13D8   2740 pre_decr_tptr:
     00028733 8F     DD     13D8   2741          PUSHL   #dbg$_cpredecr
00000000'GF   01     FB     13DE   2742          CALLS   #1,G^lib$signal
     00000034'EF     DD     13E5   2743          PUSHL   arg1_valdesc
00000000'EF   01     FB     13EB   2744          CALLS   #1,dbg$c_pre_decr_tptr
        64   50     D0     13F2   2745          MOVL    r0,(r4)
                    04     13F5   2746          RET
                     13F6   2747
                     13F6   2748 post_decr_l:
                     13F6   2749 post_decr_lu:
                     13F6   2750 post_decr_tptr:
     0002873B 8F     DD     13F6   2751          PUSHL   #dbg$_cpostdecr
00000000'GF   01     FB     13FC   2752          CALLS   #1,G^lib$signal
        64   62     D0     1403   2753          MOVL    (r2),(r4)
                    04     1406   2754          RET
                     1407   2755
                     1407   2756 post_decr_d:
     0002873B 8F     DD     1407   2757          PUSHL   #dbg$_cpostdecr
00000000'GF   01     FB     140D   2758          CALLS   #1,G^lib$signal
        64   62     70     1414   2759          MOVD    (r2),(r4)
                    04     1417   2760          RET
                     1418   2761
                     1418   2762 ;****************************************************************************
                     1418   2763 ;
                     1418   2764 ;       P A C K E D     D E C I M A L
                     1418   2765 ;
                     1418   2766 ;****************************************************************************
                     1418   2767
```

```
                                1418  2768
                                1418  2769 ; Subtraction
                                1418  2770 ;
                                1418  2771 sub_p_p:
56   00000030'EF    D0          1418  2772          MOVL    arg1_desc, r6                                   ; Get left desc.
57   00000038'EF    D0          141F  2773          MOVL    arg2_desc, r7                                   ; Get right desc.
58   00000040'EF    D0          1426  2774          MOVL    result_desc, r8                                 ; Get result desc.
       50    67    3C          142D  2775          MOVZWL  dsc$w_length(r7), r0
 50    50    FF    8F    78    1430  2776          ASHL    #-1, r0, r0                                     ; Get position of sign.
    04 B740    01    8C    1435  2777          XORB2   #1, @dsc$a_pointer(r7)[r0]                      ; Change sign and add.
               15    11    143A  2778          BRB     addp$
                                143C  2779
                                143C  2780 ; Addition.
                                143C  2781 ;
                                143C  2782 add_p_p:
56   00000030'EF    D0          143C  2783          MOVL    arg1_desc, r6                                   ; Get left desc.
57   00000038'EF    D0          1443  2784          MOVL    arg2_desc, r7                                   ; Get right desc.
58   00000040'EF    D0          144A  2785          MOVL    result_desc, r8                                 ; Get result desc.
      5E    B8    AE    9E    1451  2786 addp$:    MOVAB   -72(sp), sp                                     ; Workspace.
               56    DD          1455  2787          PUSHL   r6                                              ; Strip leading/trailing
00001B58'EF    01    FB          1457  2788          CALLS   #1, dbg$strip_zeroes                            ; zeroes.
               57    DD          145E  2789          PUSHL   r7                                              ; Same again.
00001B58'EF    01    FB          1460  2790          CALLS   #1, dbg$strip_zeroes
55    08 A6    08 A7    83    1467  2791          SUBB3   dsc$b_scale(r7), dsc$b_scale(r6), r5       ; Get change of scale.
               0C    18          146D  2792          BGEQ    1$                                              ; OK if shorter fraction.
         55    55    8E          146F  2793          MNEGB   r5, r5                                          ; Else, reverse operands.
         57    56    CC          1472  2794          XORL2   r6, r7
         56    57    CC          1475  2795          XORL2   r7, r6
         57    56    CC          1478  2796          XORL2   r6, r7
   50    66    55    81    147B  2797 1$:       ADDB3   r5, dsc$w_length(r6), r0                        ; Get total # digits.
   51    1E    50    83    147F  2798          SUBB3   r0, #30, r1                                     ; Is length in range?
               14    18    1483  2799          BGEQ    2$
         55    51    80    1485  2800          ADDB2   r1, r5                                          ; Too long.
            08 A7    51    82    1488  2801          SUBB2   r1, dsc$b_scale(r7)                             ; Adjust scale.
6E   67   00    04 B7    67    51    F8    148C  2802          ASHP    r1, dsc$w_length(r7), @dsc$a_pointer(r7), #0, dsc$w_length(r7), (sp)
               04 B7    6E    67    34    1494  2803          MOVP    dsc$w_length(r7), (sp), @dsc$a_pointer(r7)
6E   1F   00    04 B6    66    55    F8    1499  2804 2$:       ASHP    r5, dsc$w_length(r6), @dsc$a_pointer(r6), #0, #31, (sp)
     68   6E    1F    04 B7    67    21    14A1  2805          ADDP6   dsc$w_length(r7), @dsc$a_pointer(r7), -
                        04 B8          14A8
                                14AA  2806                  #31, (sp), dsc$w_length(r8), @dsc$a_pointer(r8)
            08 A8    08 A7    90    14AA  2807          MOVB    dsc$b_scale(r7), dsc$b_scale(r8)        ; Copy scale.
               09 A8    1F    90    14AF  2808          MOVB    #31, dsc$b_digits(r8)
                               04    14B3  2809          RET
                                14B4  2810
                                14B4  2811 ; Multiplication.
                                14B4  2812 ;
                                14B4  2813 mul_p_p:
                  5E    10    C2    14B4  2814          SUBL    #16, sp                                                 ; Work space
            56    00000030'EF    D0    14B7  2815          MOVL    arg1_desc, r6                                           ; Get descri
            57    00000038'EF    D0    14BE  2816          MOVL    arg2_desc, r7
            58    00000040'EF    D0    14C5  2817          MOVL    result_desc, r8
                                56    DD    14CC  2818          PUSHL   r6                                              ; Strip zero
                  00001B58'EF    01    FB    14CE  2819          CALLS   #1, dbg$strip_zeroes                            ; length.
                                57    DD    14D5  2820          PUSHL   r7
                  00001B58'EF    01    FB    14D7  2821          CALLS   #1, dbg$strip_zeroes
                                59    D4    14DE  2822          CLRL    r9
                                5A    D4    14E0  2823          CLRL    r10
```

```
                59   67   66   A1   14E2  2824          ADDW3   dsc$w_length(r6), dsc$w_length(r7), r9          ; Estimate l
                59   1F   A2   14E6  2825          SUBW2   #31, r9                                          ; Will produ
                     58   15   14E9  2826          BLEQ    mul_p$                                           ; No.
                5A   66   67   A3   14EB  2827          SUBW3   dsc$w_length(r7), dsc$w_length(r6), r10          ; Yes.  Will
                     0C   18   14EF  2828          BGEQ    1$                                               ; Want longe
                5A   5A   AE   14F1  2829          MNEGW   r10, r10                                         ; Switch ope
                51   56   D0   14F4  2830          MOVL    r6, r1
                56   57   D0   14F7  2831          MOVL    r7, r6
                57   51   D0   14FA  2832          MOVL    r1, r7
                5A   59   B1   14FD  2833 1$:       CMPW    r9, r10                                          ; Algorithm
                     27   15   1500  2834          BLEQ    5$                                               ;   (1) r9
                59   5A   A2   1502  2835          SUBW2   r10, r9                                          ;   (2) r10
                     59   B6   1505  2836          INCW    r9                                               ;   If r9 <=
           59   59   FF   8F   78   1507  2837          ASHL    #-1, r9, r9                                      ;       (1)
                5B   67   32   150C  2838          CVTWL   dsc$w_length(r7), r11                            ;       (2)
                67   59   A2   150F  2839          SUBW2   r9, dsc$w_length(r7)                             ;       (3)
              08 A7   59   80   1512  2840          ADDB2   r9, dsc$b_scale(r7)                              ;   If r9 >
                52   59   AE   1516  2841          MNEGW   r9, r2                                           ;       (1)
6E 67 00   04 B7   5B   52   F8   1519  2842          ASHP    r2, r11, @dsc$a_pointer(r7), #0, dsc$w_length(r7), (sp) ;       (2)
           04 B7   6E   67   34   1521  2843          MOVP    dsc$w_length(r7), (sp), @dsc$a_pointer(r7)       ;       (3)
                59   5A   A0   1526  2844          ADDW2   r10, r9                                          ;       (4)
                5B   66   32   1529  2845 5$:       CVTWL   dsc$w_length(r6), r11                            ;       (5)
                66   59   A2   152C  2846          SUBW2   r9, dsc$w_length(r6)
              08 A6   59   80   152F  2847          ADDB2   r9, dsc$b_scale(r6)
                59   59   AE   1533  2848          MNEGW   r9, r9
6E 66 00   04 B6   5B   59   F8   1536  2849          ASHP    r9, r11, @dsc$a_pointer(r6), #0, dsc$w_length(r6), (sp)
           04 B6   6E   66   34   153E  2850          MOVP    dsc$w_length(r6), (sp), @dsc$a_pointer(r6)
                               1543  2851 mul_p$:
        08 A8   08 A7   08 A6   81   1543  2852          ADDB3   dsc$b_scale(r6), dsc$b_scale(r7), dsc$b_scale(r8)  ; Get final
              09 A8   1F   90   154A  2853          MOVB    #31, dsc$b_digits(r8)                            ; Number of
              68   09 A8   99   154E  2854          CVTBW   dsc$b_digits(r8), dsc$w_length(r8)               ; Fill in nu
 1F 04 B7   67   04 B6   66   25   1552  2855          MULP    dsc$w_length(r6), @dsc$a_pointer(r6), -          ; Multiply a
                     04 B8         155A
                               155C  2856                  dsc$w_length(r7), @dsc$a_pointer(r7), -
                               155C  2857                  #31, @dsc$a_pointer(r8)
                          04   155C  2858          RET
                               155D  2859
                               155D  2860 ; Division.
                               155D  2861 ;
                               155D  2862 div_p_p:
                5E   10   C2   155D  2863          SUBL    #16, sp                                          ; Workspace.
          56   00000030'EF   D0   1560  2864          MOVL    arg1_desc, r6                                    ; Get descri
          57   00000038'EF   D0   1567  2865          MOVL    arg2_desc, r7                                    ;    zeroes
          58   00000040'EF   D0   156E  2866          MOVL    result_desc, r8
                     56   DD   1575  2867          PUSHL   r6
          00001B58'EF   01   FB   1577  2868          CALLS   #1, dbg$strip_zeroes
                     57   DD   157E  2869          PUSHL   r7
          00001B58'EF   01   FB   1580  2870          CALLS   #1, dbg$strip_zeroes
              5A   1F   66   A3   1587  2871          SUBW3   dsc$w_length(r6), #31, r10                       ; Dividend m
                08 A6   5A   82   158B  2872          SUBB2   r10, dsc$b_scale(r6)                             ;    Adjust
6E 1F 00   04 B6   66   5A   F8   158F  2873          ASHP    r10, dsc$w_length(r6), @dsc$a_pointer(r6), #0, #31, (sp) ;    Shift t
           04 B6   6E   1F   34   1597  2874          MOVP    #31, (sp), @dsc$a_pointer(r6)
                66   1F   B0   159C  2875          MOVW    #31, dsc$w_length(r6)                            ;    Length
                68   1F   B0   159F  2876          MOVW    #31, dsc$w_length(r8)                            ; Quotient p
              08 A8   68   33   15A2  2877          CVTWB   dsc$w_length(r8), dsc$b_scale(r8)                ; Number of
        08 A8   08 A6   08 A7   83   15A6  2878          SUBB3   dsc$b_scale(r7), dsc$b_scale(r6), dsc$b_scale(r8) ; Estimate q
                7E   67   3C   15AD  2879          MOVZWL  dsc$w_length(r7), -(sp)                          ; Push all i
```

```
                     00    DD   15B0  2880          PUSHL   #0
                  7E 68    3C   15B2  2881          MOVZWL  dsc$w_length(r8), -(sp)
                     04 A8 DD   15B5  2882          PUSHL   dsc$a_pointer(r8)
                  7E 67    3C   15B8  2883          MOVZWL  dsc$w_length(r7), -(sp)
                     04 A7 DD   15BB  2884          PUSHL   dsc$a_pointer(r7)
                     04 A6 DD   15BE  2885          PUSHL   dsc$a_pointer(r6)
          00000000'GF 07    FB   15C1  2886          CALLS   #7, G^PLI$DIV_PK_LONG                                   ; Do the div
                        04    15C8  2887          RET
                              15C9  2888
                              15C9  2889
                              15C9  2890 ; Unary minus.
                              15C9  2891 ;
                              15C9  2892 unary_minus_p:
          56 00000030'EF      D0    15C9  2893          MOVL    arg1_desc, r6                                     ; Get descri
          58 00000040'EF      D0    15D0  2894          MOVL    result_desc, r8
                     50 66    3C    15D7  2895          MOVZWL  dsc$w_length(r6), r0
           50 50 FF 8F        78    15DA  2896          ASHL    #-1, r0, r0                                       ; Determine
              04 B640         01    8C    15DF  2897    XORB2   #1, @dsc$a_pointer(r6)[r0]                        ; Change sig
                        0E    11    15E4  2898          BRB     unary_p$                                          ;   descript
                              15E6  2899
                              15E6  2900 ; Unary plus.
                              15E6  2901 ;
                              15E6  2902 unary_plus_p:
          56 00000030'EF      D0    15E6  2903          MOVL    arg1_desc, r6                                     ; Get descri
          58 00000040'EF      D0    15ED  2904          MOVL    result_desc, r8
                              15F4  2905 unary_p$:
                     56       DD    15F4  2906          PUSHL   r6                                                ; Strip lead
          00001B58'EF         01    FB    15F6  2907    CALLS   #1, dbg$strip_zeroes
       04 B8 04 B6            66    34    15FD  2908    MOVP    dsc$w_length(r6), @dsc$a_pointer(r6), @dsc$a_pointer(r8); Move to re
                     68       66    B0    1603  2909    MOVW    dsc$w_length(r6), dsc$w_length(r8)                ; Adjust len
              08 A8 08 A6     90    1606  2910          MOVB    dsc$b_scale(r6), dsc$b_scale(r8)
                 09 A8 68     33    160B  2911          CVTWB   dsc$w_length(r8), dsc$b_digits(r8)
                        04    160F  2912          RET
                              1610  2913
                              1610  2914 ; EQL, NEQ, GTR, GEQ, LSS, and LEQ.
                              1610  2915 ;
                              1610  2916 eql_p_p:
                56 00000030'EF D0    1610  2917          MOVL    arg1_desc, r6                                     ; Get descri
                57 00000038'EF D0    1617  2918          MOVL    arg2_desc, r7
                           56 DD     161E  2919          PUSHL   r6                                                ; Strip lead
                00001B58'EF   01 FB  1620  2920          CALLS   #1, dbg$strip_zeroes                              ; zeroes.
                           57 DD     1627  2921          PUSHL   r7                                                ; Same again
                00001B58'EF   01 FB  1629  2922          CALLS   #1, dbg$strip_zeroes
                           64 D4     1630  2923          CLRL    (r4)                                              ; Assume NEQ
       04 B7 67 04 B6     66 37     1632  2924          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -           ; Do the com
                                    1639  2925                  dsc$w_length(r7), @dsc$a_pointer(r7)              ;   be sure
                           0A 12     1639  2926          BNEQ    eql_p$
                  08 A7 08 A6 91     163B  2927          CMPB    dsc$b_scale(r6), dsc$b_scale(r7)
                           03 12     1640  2928          BNEQ    eql_p$
                        64 01 D0     1642  2929          MOVL    #1, (r4)                                          ; Success.
                                    1645  2930 eql_p$:
                        ECBD 31     1645  2931          BRW     branch_ret
                                    1648  2932
                                    1648  2933 neq_p_p:
                56 00000030'EF D0    1648  2934          MOVL    arg1_desc, r6                                     ; Get descri
                57 00000038'EF D0    164F  2935          MOVL    arg2_desc, r7
                           56 DD     1656  2936          PUSHL   r6                                                ; Strip lead
```

```
                00001B58'EF   01   FB  1658  2937          CALLS   #1, dbg$strip_zeroes                          ; zeroes.
                              57   DD  165F  2938          PUSHL   r7                                           ; Same again
                00001B58'EF   01   FB  1661  2939          CALLS   #1, dbg$strip_zeroes
                         64   01   D0  1668  2940          MOVL    #1, (r4)                                     ; Assume NEQ
    04 B7   67   04 B6   66   37   166B  2941          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -      ; Do the com
                                    1672  2942                  dsc$w_length(r7), @dsc$a_pointer(r7)         ;   sure to
                              09   12  1672  2943          BNEQ    neq_p$
                   08 A7   08 A6   91  1674  2944          CMPB    dsc$b_scale(r6), dsc$b_scale(r7)
                              02   12  1679  2945          BNEQ    neq_p$
                              64   D4  167B  2946          CLRL    (r4)                                         ; Failure.
                                    167D  2947  neq_p$:
                            EC85   31  167D  2948          BRW     branch_ret
                                    1680  2949
                                    1680  2950  gtr_p_p:
                            004D   30  1680  2951          BSBW    setup_packed_numbers$
                         64   01   D0  1683  2952          MOVL    #1, (r4)                                     ; Assume GTR
    04 B7   67   04 B6   66   37   1686  2953          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -      ; Scales wer
                                    168D  2954                  dsc$w_length(r7), @dsc$a_pointer(r7)
                              02   14  168D  2955          BGTR    gtr_p$
                              64   D4  168F  2956          CLRL    (r4)                                         ; Failure.
                                    1691  2957  gtr_p$:
                            EC71   31  1691  2958          BRW     branch_ret
                                    1694  2959
                                    1694  2960  geq_p_p:
                            0039   30  1694  2961          BSBW    setup_packed_numbers$
                         64   01   D0  1697  2962          MOVL    #1, (r4)                                     ; Assume GEQ
    04 B7   67   04 B6   66   37   169A  2963          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -      ; Scales wer
                                    16A1  2964                  dsc$w_length(r7), @dsc$a_pointer(r7)
                              02   18  16A1  2965          BGEQ    geq_p$
                              64   D4  16A3  2966          CLRL    (r4)                                         ; Failure.
                                    16A5  2967  geq_p$:
                            EC5D   31  16A5  2968          BRW     branch_ret
                                    16A8  2969
                                    16A8  2970  lss_p_p:
                            0025   30  16A8  2971          BSBW    setup_packed_numbers$
                         64   01   D0  16AB  2972          MOVL    #1, (r4)                                     ;
    04 B7   67   04 B6   66   37   16AE  2973          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -      ; Scales wer
                                    16B5  2974                  dsc$w_length(r7), @dsc$a_pointer(r7)
                              02   19  16B5  2975          BLSS    lss_p$
                              64   D4  16B7  2976          CLRL    (r4)                                         ; Failure.
                                    16B9  2977  lss_p$:
                            EC49   31  16B9  2978          BRW     branch_ret
                                    16BC  2979
                                    16BC  2980  leq_p_p:
                            0011   30  16BC  2981          BSBW    setup_packed_numbers$
                         64   01   D0  16BF  2982          MOVL    #1, (r4)                                     ; Assume LEQ
    04 B7   67   04 B6   66   37   16C2  2983          CMPP4   dsc$w_length(r6), @dsc$a_pointer(r6), -      ; Scales wer
                                    16C9  2984                  dsc$w_length(r7), @dsc$a_pointer(r7)
                              02   15  16C9  2985          BLEQ    leq_p$
                              64   D4  16CB  2986          CLRL    (r4)                                         ; Failure.
                                    16CD  2987  leq_p$:
                            EC35   31  16CD  2988          BRW     branch_ret
                                    16D0  2989
                                    16D0  2990  setup_packed_numbers$:
                              58   D4  16D0  2991          CLRL    r8                                   ; Use as a adjustment scale
                              59   D4  16D2  2992          CLRL    r9                                   ; Use as a adjustment digits
            56   00000030'EF   D0  16D4  2993          MOVL    arg1_desc, r6                        ; Get descriptors
```

```
                57   00000038'EF   D0   16DB  2994          MOVL    arg2_desc, r7
                               56   DD   16E2  2995          PUSHL   r6                      ; Strip leading/trailing
                00001B58'EF    01   FB   16E4  2996          CALLS   #1, dbg$strip_zeroes    ;  zeroes
                               57   DD   16EB  2997          PUSHL   r7                      ; Same again
                00001B58'EF    01   FB   16ED  2998          CALLS   #1, dbg$strip_zeroes
             58    08 A7    08 A6   83   16F4  2999          SUBB3   dsc$b_scale(r6),dsc$b_scale(r7),r8; See if they are scaled
                                          16FA  3000                                          ;   the same
                               58   95   16FA  3001          TSTB    r8
                               76   13   16FC  3002          BEQL    setup_packed_rsb$       ; They are, simply return
                               05   14   16FE  3003          BGTR    scale_arg2$             ; Adjust the second operand
                                         1700  3004                                          ;  (first has larger scaling)
                          58   58   8E   1700  3005          MNEGB   r8, r8
                               0E   11   1703  3006          BRB     scale_arg1$             ; Adjust the first operand
                                         1705  3007                                          ;  (second has larger scaling)
                                         1705  3008 scale_arg2$:
                56   00000038'EF   D0   1705  3009          MOVL    arg2_desc, r6           ; Use r6 to point to the
                                         170C  3010                                          ;  operand needs adjustment
                                         170C  3011                                          ;  (the one has smaller scaling)
                57   00000030'EF   D0   170C  3012          MOVL    arg1_desc, r7           ; r7 points to the operand
                                         1713  3013                                          ;  has larger scaling
                                         1713  3014 scale_arg1$:
                59   58    09 A6   81   1713  3015          ADDB3   dsc$b_digits(r6), r8, r9 ; Update the total number of
                                         1718  3016                                          ;  digits
                          1F   59   91   1718  3017          CMPB    r9, #31                 ; Check to see if this causes
                                         171B  3018                                          ;  overflow
                               25   14   171B  3019          BGTR    truncate_the_other$     ; If we try to adjust the
                                         171D  3020                                          ;  smaller scale will cause
                                         171D  3021                                          ;  overflow, so We truncate the
                                         171D  3022                                          ;  other scale instead
    59   00    04 B6   66   58   F8   171D  3023          ASHP    r8, dsc$w_length(r6), @dsc$a_pointer(r6), -
                00000048'EF            1724
                                         1729  3024                  #0, r9, operand
  00   00000048'EF    59   00   F8   1729  3025          ASHP    #0, r9, operand, #0, r9, @dsc$a_pointer(r6)
                     04 B6   59        1732
                        66   59   B0   1735  3026          MOVW    r9, dsc$w_length(r6)
                     09 A6   59   90   1738  3027          MOVB    r9, dsc$b_digits(r6)
                     08 A6   58   82   173C  3028          SUBB2   r8, dsc$b_scale(r6)
                               32   11   1740  3029          BRB     setup_packed_rsb$
                                         1742  3030 truncate_the_other$:
                               58   97   1742  3031          DECB    r8                      ; Shift count
                          5A   67   3C   1744  3032          MOVZWL  dsc$w_length(r7), r10   ; Get the length
                               5A   B7   1747  3033          DECW    r10                     ; Length
 5A   00    04 B7    67    FF  8F   F8   1749  3034          ASHP    #-1, dsc$w_length(r7), @dsc$a_pointer(r7), -
                00000048'EF            1751
                                         1756  3035                  #0, r10, operand
  00   00000048'EF    5A   00   F8   1756  3036          ASHP    #0, r10, operand, #0, r10, @dsc$a_pointer(r7)
                     04 B7   5A        175F
                        08 A7   97   1762  3037          DECB    dsc$b_scale(r7)
                               5A   B5   1765  3038          TSTW    r10                     ; If we got done to zero
                                         1767  3039                                          ;  digit, then we are all done
                               0B   13   1767  3040          BEQL    setup_packed_rsb$
                          67   5A   B0   1769  3041          MOVW    r10, dsc$w_length(r7)
                     09 A7    5A   90   176C  3042          MOVB    r10, dsc$b_digits(r7)
                               58   95   1770  3043          TSTB    r8
                               9F   12   1772  3044          BNEQ    scale_arg1$
                                         1774  3045 setup_packed_rsb$:
                56   00000030'EF   D0   1774  3046          MOVL    arg1_desc, r6
```

```
                57   00000038'EF   D0  1778  3047          MOVL    arg2_desc, r7
                                   05  1782  3048          RSB
                                       1783  3049
                                       1783  3050
                                       1783  3051 ; PLI bit-string operations.
                                       1783  3052 ;
                                       1783  3053 concat_tf_tf:
                          0179     30  1783  3054          BSBW    setup_pli_intrface$
                       51   54     D0  1786  3055          MOVL    r4, r1
                  0000000C'EF     DF  1789  3056          PUSHAL  dope2
                            53     DD  178F  3057          PUSHL   r3
                  00000008'EF     DF  1791  3058          PUSHAL  dope1
                            52     DD  1797  3059          PUSHL   r2
              00000000'GF   04     FB  1799  3060          CALLS   #4, G^PLI$CATBIT
                            04     17A0  3061          RET
                                   17A1  3062
                                   17A1  3063 eql_tf_tf:
                          015B     30  17A1  3064          BSBW    setup_pli_intrface$
                       64   01     D0  17A4  3065          MOVL    #1, (r4)
                  0000000C'EF     DF  17A7  3066          PUSHAL  dope2
                            53     DD  17AD  3067          PUSHL   r3
                  00000008'EF     DF  17AF  3068          PUSHAL  dope1
                            52     DD  17B5  3069          PUSHL   r2
              00000000'GF   04     FB  17B7  3070          CALLS   #4, G^PLI$CMPBIT
                            50     D5  17BE  3071          TSTL    r0
                            02     13  17C0  3072          BEQL    eql_tf$
                            64     D4  17C2  3073          CLRL    (r4)
                                   17C4  3074 eql_tf$:
                       EB3E     31  17C4  3075          BRW     branch_ret
                                   17C7  3076
                                   17C7  3077 geq_tf_tf:
                          0135     30  17C7  3078          BSBW    setup_pli_intrface$
                       64   01     D0  17CA  3079          MOVL    #1,(r4)
                  0000000C'EF     DF  17CD  3080          PUSHAL  dope2
                            53     DD  17D3  3081          PUSHL   r3
                  00000008'EF     DF  17D5  3082          PUSHAL  dope1
                            52     DD  17DB  3083          PUSHL   r2
              00000000'GF   04     FB  17DD  3084          CALLS   #4, G^PLI$CMPBIT
                            50     D5  17E4  3085          TSTL    r0
                            08     13  17E6  3086          BEQL    geq_tf$
          FFFFFFFF   8F     50     D1  17E8  3087          CMPL    r0, #^XFFFFFFFF
                            02     12  17EF  3088          BNEQ    geq_tf$
                            64     D4  17F1  3089          CLRL    (r4)
                                   17F3  3090 geq_tf$:
                       EB0F     31  17F3  3091          BRW     branch_ret
                                   17F6  3092
                                   17F6  3093 gtr_tf_tf:
                          0106     30  17F6  3094          BSBW    setup_pli_intrface$
                       64   01     D0  17F9  3095          MOVL    #1,(r4)
                  0000000C'EF     DF  17FC  3096          PUSHAL  dope2
                            53     DD  1802  3097          PUSHL   r3
                  00000008'EF     DF  1804  3098          PUSHAL  dope1
                            52     DD  180A  3099          PUSHL   r2
              00000000'GF   04     FB  180C  3100          CALLS   #4, G^PLI$CMPBIT
                            50     D5  1813  3101          TSTL    r0
                            0B     13  1815  3102          BEQL    not_gtr$
          FFFFFFFF   8F     50     D1  1817  3103          CMPL    r0, #^XFFFFFFFF
```

```
                    02   13  181E  3104          BEQL    not_gtr$
                    02   11  1820  3105          BRB     gtr_tf$
                             1822  3106 not_gtr$:
                    64   D4  1822  3107          CLRL    (r4)
                             1824  3108 gtr_tf$:
                  EADE   31  1824  3109          BRW     branch_ret
                             1827  3110
                             1827  3111 leq_tf_tf:
                  00D5   30  1827  3112          BSBW    setup_pli_intrface$
                64   01   D0  182A  3113          MOVL    #1,(r4)
          0000000C'EF      DF  182D  3114          PUSHAL  dope2
                    53   DD  1833  3115          PUSHL   r3
          00000008'EF      DF  1835  3116          PUSHAL  dope1
                    52   DD  183B  3117          PUSHL   r2
      00000000'GF   04   FB  183D  3118          CALLS   #4, G^PLI$CMPBIT
                    50   D5  1844  3119          TSTL    r0
                    0B   13  1846  3120          BEQL    leq_tf$
      FFFFFFFF 8F   50   D1  1848  3121          CMPL    r0, #^XFFFFFFFF
                    02   13  184F  3122          BEQL    leq_tf$
                    64   D4  1851  3123          CLRL    (r4)
                             1853  3124 leq_tf$:
                  EAAF   31  1853  3125          BRW     branch_ret
                             1856  3126
                             1856  3127 lss_tf_tf:
                  00A6   30  1856  3128          BSBW    setup_pli_intrface$
                64   01   D0  1859  3129          MOVL    #1,(r4)
          0000000C'EF      DF  185C  3130          PUSHAL  dope2
                    53   DD  1862  3131          PUSHL   r3
          00000008'EF      DF  1864  3132          PUSHAL  dope1
                    52   DD  186A  3133          PUSHL   r2
      00000000'GF   04   FB  186C  3134          CALLS   #4, G^PLI$CMPBIT
                    50   D5  1873  3135          TSTL    r0
                    0B   13  1875  3136          BEQL    not_lss$
      FFFFFFFF 8F   50   D1  1877  3137          CMPL    r0, #^XFFFFFFFF
                    02   12  187E  3138          BNEQ    not_lss$
                    02   11  1880  3139          BRB     lss_tf$
                             1882  3140 not_lss$:
                    64   D4  1882  3141          CLRL    (r4)
                             1884  3142 lss_tf$:
                  EA7E   31  1884  3143          BRW     branch_ret
                             1887  3144
                             1887  3145 neq_tf_tf:
                  0075   30  1887  3146          BSBW    setup_pli_intrface$
                64   01   D0  188A  3147          MOVL    #1,(r4)
          0000000C'EF      DF  188D  3148          PUSHAL  dope2
                    53   DD  1893  3149          PUSHL   r3
          00000008'EF      DF  1895  3150          PUSHAL  dope1
                    52   DD  189B  3151          PUSHL   r2
      00000000'GF   04   FB  189D  3152          CALLS   #4, G^PLI$CMPBIT
                    50   D5  18A4  3153          TSTL    r0
                    02   12  18A6  3154          BNEQ    neq_tf$
                    64   D4  18A8  3155          CLRL    (r4)
                             18AA  3156 neq_tf$:
                  EA58   31  18AA  3157          BRW     branch_ret
                             18AD  3158
                             18AD  3159 bit_not_tf_tf:
                  004F   30  18AD  3160          BSBW    setup_pli_intrface$
```

```
                51    54    D0  18B0  3161          MOVL    r4, r1
          00000008'EF       DF  18B3  3162          PUSHAL  dope1
                52          DD  18B9  3163          PUSHL   r2
    00000000'GF   02          FB  18BB  3164          CALLS   #2, G^PLI$NOTBIT
                              04  18C2  3165          RET
                                  18C3  3166
                                  18C3  3167 bit_and_tf_tf:
                0039          30  18C3  3168          BSBW    setup_pli_intrface$
                51    54      D0  18C6  3169          MOVL    r4, r1
          0000000C'EF         DF  18C9  3170          PUSHAL  dope2
                53          DD  18CF  3171          PUSHL   r3
          00000008'EF         DF  18D1  3172          PUSHAL  dope1
                52          DD  18D7  3173          PUSHL   r2
    00000000'GF   04          FB  18D9  3174          CALLS   #4, G^PLI$ANDBIT
                              04  18E0  3175          RET
                                  18E1  3176
                                  18E1  3177 bit_or_tf_tf:
                001B          30  18E1  3178          BSBW    setup_pli_intrface$
                51    54      D0  18E4  3179          MOVL    r4, r1
          0000000C'EF         DF  18E7  3180          PUSHAL  dope2
                53          DD  18ED  3181          PUSHL   r3
          00000008'EF         DF  18EF  3182          PUSHAL  dope1
                52          DD  18F5  3183          PUSHL   r2
    00000000'GF   04          FB  18F7  3184          CALLS   #4, G^PLI$ORBIT
                              04  18FE  3185          RET
                                  18FF  3186
                                  18FF  3187
                                  18FF  3188 ; This subroutine sets up the call interface required for the PL/I RTL
                                  18FF  3189 ; bit-string routines.  Those routines require the following for both
                                  18FF  3190 ; operands:
                                  18FF  3191 ;
                                  18FF  3192 ;           A dope vector of the form:
                                  18FF  3193 ;
                                  18FF  3194 ;           +----------+---------------+
                                  18FF  3195 ;           |   Length | PLI data type |
                                  18FF  3196 ;           +----------+---------------+
                                  18FF  3197 ;
                                  18FF  3198 ;           DSC$K_DTYPE_V maps into DBG$K_PLI_ABIT,
                                  18FF  3199 ;           DSC$K_DTYPE_VU maps into DBG$K_PLI_UBIT.
                                  18FF  3200 ;
                                  18FF  3201 ;           Length is in bits.
                                  18FF  3202 ;
                                  18FF  3203 setup_pli_intrface$:
52    00000030'EF         D0  18FF  3204          MOVL    arg1_desc, r2           ; r2 points left vms descriptor
0000000A'EF   62          B0  1906  3205          MOVW    dsc$w_length(r2), dope1+2; get length
      01    02 A2         91  190D  3206          CMPB    dsc$b_dtype(r2), #dsc$k_dtype_v
                0D        13  1911  3207          BEQL    setup_pli_abit1$
00000008'EF   0C          B0  1913  3208          MOVW    #dbg$k_pli_ubit, dope1  ; map into pli data type
      52    04 A2         DE  191A  3209          MOVAL   4(r2), r2               ; Get the address of the 2nd longword
                              191E  3210                                          ;  of the vms descriptor
                0B        11  191E  3211          BRB     setup_pli_cont1$
                              1920  3212 setup_pli_abit1$:
00000008'EF   0E          B0  1920  3213          MOVW    #dbg$k_pli_abit, dope1  ; map into pli data type
      52    04 A2         D0  1927  3214          MOVL    4(r2), r2               ; Get the address of the left value
                              192B  3215 setup_pli_cont1$:
      53    10 AC         D0  192B  3216          MOVL    right_arg(AP), r3       ; Get the address of the right arg
                53          D5  192F  3217          TSTL    r3                      ; Test to see if it is present
```

```
                 2C   13  1931  3218          BEQL    setup_pli_result$       ; There is no right arg
53   00000038'EF    D0  1933  3219          MOVL    arg2_desc, r3           ; r3 points right vms descriptor
0000000E'EF    63   B0  193A  3220          MOVW    dsc$w_length(r3), dope2+2; get length
      01   02 A3   91  1941  3221          CMPB    dsc$b_dtype(r3), #dsc$k_dtype_v
                 0D   13  1945  3222          BEQL    setup_pli_abit2$
00000000'EF    0C   B0  1947  3223          MOVW    #dbg$k_pli_ubit, dope2  ; map into pli data type
      53   04 A3   DE  194E  3224          MOVAL   4(r3), r3               ; Get the address of the 2nd longword
                       1952  3225                                          ;  of the vms descriptor
                 0B   11  1952  3226          BRB     setup_pli_result$
                       1954  3227  setup_pli_abit2$:
0000000C'EF    0E   B0  1954  3228          MOVW    #dbg$k_pli_abit, dope2  ; map into pli data type
      53   04 A3   D0  195B  3229          MOVL    4(r3), r3               ; Get the address of the right value
                       195F  3230  setup_pli_result$:
54   00000040'EF    D0  195F  3231          MOVL    result_desc, r4         ; r4 points to result vms descriptor
      54   04 A4   D0  1966  3232          MOVL    4(r4), r4               ; r4 points to the address of the result
                       196A  3233  setup_rsb$:
                 05  196A  3234          RSB
                       196B  3235
                       196B  3236
                       196B  3237  ; Record File Address data type, 6 bytes long.
                       196B  3238  ;
                       196B  3239  eql_rfa_rfa:
                 64   D4  196B  3240          CLRL    (r4)
            63   62   D1  196D  3241          CMPL    (r2),(r3)
                 0A   12  1970  3242          BNEQ    eql_rfa$
   04 A3    04   A2   B1  1972  3243          CMPW    4(r2),4(r3)
                 03   12  1977  3244          BNEQ    eql_rfa$
            64   01   D0  197A  3245          MOVL    #1,(r4)
                        C  3246  eql_rfa$:
               E986   31  197C  3247          BRW     branch_ret
                       197F  3248
                       197F  3249  neq_rfa_rfa:
                 64   D4  197F  3250          CLRL    (r4)
            63   62   D1  1981  3251          CMPL    (r2),(r3)
                 0A   13  1984  3252          BEQL    neq_rfa$
   04 A3    04   A2   B1  1986  3253          CMPW    4(r2),4(r3)
                 03   13  198B  3254          BEQL    neq_rfa$
            64   01   D0  198D  3255          MOVL    #1,(r4)
                       1990  3256  neq_rfa$:
               E972   31  1990  3257          BRW     branch_ret
                       1993  3258
                       1993  3259  unary_plus_q:
            64   62   7D  1993  3260          MOVQ    (r2),(r4)
               E96C   31  1996  3261          BRW     branch_ret
                       1999  3262
                       1999  3263  unary_minus_q:
            64   62   D2  1999  3264          MCOML   (r2),(r4)
   04 A4    04   A2   D2  199C  3265          MCOML   4(r2),4(r4)
            64   01   C0  19A1  3266          ADDL2   #1,(r4)
      04 A4   00   D8  19A4  3267          ADWC    #0,4(r4)
               E95A   31  19A8  3268          BRW     branch_ret
                       19AB  3269
                       19AB  3270  unary_plus_o:
            64   62   7D  19AB  3271          MOVQ    (r2),(r4)
   08 A4    08   A2   7D  19AE  3272          MOVQ    8(r2),8(r4)
               E94F   31  19B3  3273          BRW     branch_ret
                       19B6  3274
```

```
                        19B6  3275 unary_minus_o:
          64    62  D2  19B6  3276         MCOML   (r2),(r4)
    04 A4    04 A2  D2  19B9  3277         MCOML   4(r2),4(r4)
    08 A4    08 A2  D2  19BE  3278         MCOML   8(r2),8(r4)
    0C A4    0C A2  D2  19C3  3279         MCOML   12(r2),12(r4)
          64    01  C0  19C8  3280         ADDL2   #1,(r4)
       04 A4    00  D8  19CB  3281         ADWC    #0,4(r4)
       08 A4    00  D8  19CF  3282         ADWC    #0,8(r4)
       0C A4    00  D8  19D3  3283         ADWC    #0,12(r4)
             E92B   31  19D7  3284         BRW     branch_ret
                        19DA  3285
                        19DA  3286 succ_enum:                                           ;BB001
     00000044'EF    DD  19DA  3287         PUSHL   result_valdesc                      ;BB001
     00000034'EF    DD  19E0  3288         PUSHL   arg1_valdesc                        ;BB001
00000000'EF    02   FB  19E6  3289         CALLS   #2, dbg$succ_enum                   ;BB001
                    04  19ED  3290         RET                                         ;BB001
                        19EE  3291
                        19EE  3292 pred_enum:                                           ;BB001
     00000044'EF    DD  19EE  3293         PUSHL   result_valdesc                      ;BB001
     00000034'EF    DD  19F4  3294         PUSHL   arg1_valdesc                        ;BB001
00000000'EF    02   FB  19FA  3295         CALLS   #2, dbg$pred_enum                   ;BB001
                    04  1A01  3296         RET                                         ;BB001
                        1A02  3297
                        1A02  3298 ; The following handle operations on fixed binary.
                        1A02  3299 ;
                        1A02  3300
                        1A02  3301 unary_plus_fixed:
     00000044'EF    DD  1A02  3302         PUSHL   result_valdesc
     00000034'EF    DD  1A08  3303         PUSHL   arg1_valdesc
00000000'EF    02   FB  1A0E  3304         CALLS   #2, dbg$unary_plus_fixed
                    04  1A15  3305         RET
                        1A16  3306
                        1A16  3307 unary_minus_fixed:
     00000044'EF    DD  1A16  3308         PUSHL   result_valdesc
     00000034'EF    DD  1A1C  3309         PUSHL   arg1_valdesc
00000000'EF    02   FB  1A22  3310         CALLS   #2, dbg$unary_minus_fixed
                    04  1A29  3311         RET
                        1A2A  3312
                        1A2A  3313 abs_fixed:
     00000044'EF    DD  1A2A  3314         PUSHL   result_valdesc
     00000034'EF    DD  1A30  3315         PUSHL   arg1_valdesc
00000000'EF    02   FB  1A36  3316         CALLS   #2, dbg$abs_fixed
                    04  1A3D  3317         RET
                        1A3E  3318
                        1A3E  3319 add_fixed_fixed:
     00000044'EF    DD  1A3E  3320         PUSHL   result_valdesc
     0000003C'EF    DD  1A44  3321         PUSHL   arg2_valdesc
     00000034'EF    DD  1A4A  3322         PUSHL   arg1_valdesc
00000000'EF    03   FB  1A50  3323         CALLS   #3, dbg$add_fixed_fixed
                    04  1A57  3324         RET
                        1A58  3325
                        1A58  3326 sub_fixed_fixed:
     00000044'EF    DD  1A58  3327         PUSHL   result_valdesc
     0000003C'EF    DD  1A5E  3328         PUSHL   arg2_valdesc
     00000034'EF    DD  1A64  3329         PUSHL   arg1_valdesc
00000000'EF    03   FB  1A6A  3330         CALLS   #3, dbg$sub_fixed_fixed
                    04  1A71  3331         RET
```

```
                        1A72  3332
                        1A72  3333 mul_fixed_fixed:
     00000044'EF    DD  1A72  3334          PUSHL   result_valdesc
     0000003C'EF    DD  1A78  3335          PUSHL   arg2_valdesc
     00000034'EF    DD  1A7E  3336          PUSHL   arg1_valdesc
00000000'EF    03  FB  1A84  3337          CALLS   #3, dbg$mul_fixed_fixed
                    04  1A8B  3338          RET
                        1A8C  3339
                        1A8C  3340 div_fixed_fixed:
     00000044'EF    DD  1A8C  3341          PUSHL   result_valdesc
     0000003C'EF    DD  1A92  3342          PUSHL   arg2_valdesc
     00000034'EF    DD  1A98  3343          PUSHL   arg1_valdesc
00000000'EF    03  FB  1A9E  3344          CALLS   #3, dbg$div_fixed_fixed
                    04  1AA5  3345          RET
                        1AA6  3346
                        1AA6  3347 eql_fixed_fixed:
     00000044'EF    DD  1AA6  3348          PUSHL   result_valdesc
     0000003C'EF    DD  1AAC  3349          PUSHL   arg2_valdesc
     00000034'EF    DD  1AB2  3350          PUSHL   arg1_valdesc
00000000'EF    03  FB  1AB8  3351          CALLS   #3, dbg$eql_fixed_fixed
                    04  1ABF  3352          RET
                        1AC0  3353
                        1AC0  3354 neq_fixed_fixed:
     00000044'EF    DD  1AC0  3355          PUSHL   result_valdesc
     0000003C'EF    DD  1AC6  3356          PUSHL   arg2_valdesc
     00000034'EF    DD  1ACC  3357          PUSHL   arg1_valdesc
00000000'EF    03  FB  1AD2  3358          CALLS   #3, dbg$neq_fixed_fixed
                    04  1AD9  3359          RET
                        1ADA  3360
                        1ADA  3361 lss_fixed_fixed:
     00000044'EF    DD  1ADA  3362          PUSHL   result_valdesc
     0000003C'EF    DD  1AE0  3363          PUSHL   arg2_valdesc
     00000034'EF    DD  1AE6  3364          PUSHL   arg1_valdesc
00000000'EF    03  FB  1AEC  3365          CALLS   #3, dbg$lss_fixed_fixed
                    04  1AF3  3366          RET
                        1AF4  3367
                        1AF4  3368 gtr_fixed_fixed:
     00000044'EF    DD  1AF4  3369          PUSHL   result_valdesc
     0000003C'EF    DD  1AFA  3370          PUSHL   arg2_valdesc
     00000034'EF    DD  1B00  3371          PUSHL   arg1_valdesc
00000000'EF    03  FB  1B06  3372          CALLS   #3, dbg$gtr_fixed_fixed
                    04  1B0D  3373          RET
                        1B0E  3374
                        1B0E  3375 leq_fixed_fixed:
     00000044'EF    DD  1B0E  3376          PUSHL   result_valdesc
     0000003C'EF    DD  1B14  3377          PUSHL   arg2_valdesc
     00000034'EF    DD  1B1A  3378          PUSHL   arg1_valdesc
00000000'EF    03  FB  1B20  3379          CALLS   #3, dbg$leq_fixed_fixed
                    04  1B27  3380          RET
                        1B28  3381
                        1B28  3382 geq_fixed_fixed:
     00000044'EF    DD  1B28  3383          PUSHL   result_valdesc
     0000003C'EF    DD  1B2E  3384          PUSHL   arg2_valdesc
     00000034'EF    DD  1B34  3385          PUSHL   arg1_valdesc
00000000'EF    03  FB  1B3A  3386          CALLS   #3, dbg$geq_fixed_fixed
                    04  1B41  3387          RET
                        1B42  3388
```

```
                          1B42  3389 ; No corresponding routine index to perform the given operation, signal
                          1B42  3390 ; debug internal coding error.
                          1B42  3391 ;
                          1B42  3392 unknown_rout_index:
        00000000'EF   DF  1B42  3393          PUSHAL  errmsg
                 01   DD  1B48  3394          PUSHL   #1
           00028362   8F  DD  1B4A  3395          PUSHL   #dbg$_interr
00000000'GF      03   FB  1B50  3396          CALLS   #3,G^LIB$SIGNAL
                      04  1B57  3397          RET
```

```
                     1B58  3399          .SBTTL  dbg$strip_zeroes
                     1B58  3400
                     1B58  3401 ;++
                     1B58  3402 ;GLOBAL ROUTINE dbg$strip_zeroes (VMS_DESC) =
                     1B58  3403 ;
                     1B58  3404 ; FUNCTION
                     1B58  3405 ;       This routine strips leading and trailing zeroes from a packed
                     1B58  3406 ;       decimal string.
                     1B58  3407 ;
                     1B58  3408 ; INPUTS
                     1B58  3409 ;       VMS_DESC - A pointer to a VMS descriptor.
                     1B58  3410 ;
                     1B58  3411 ; OUTPUTS
                     1B58  3412 ;       VMS_DESC is returned.
                     1B58  3413 ;--
                     1B58  3414
                     1B58  3415 ; Register usage:
                     1B58  3416 ;
                     1B58  3417 ;       r0,r1   temporaries
                     1B58  3418 ;       r6      pointer to VMS descriptor
                     1B58  3419 ;       r7      number of decimal digits
                     1B58  3420 ;       r8      pointer to digit string
                     1B58  3421 ;       r9      index
                     1B58  3422
             03FC    1B58  3423 .ENTRY  dbg$strip_zeroes, ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                     1B5A  3424
          5E   10    C2    1B5A  3425          SUBL    #16,SP                          ; Local storage space
       56   04 AC    D0    1B5D  3426          MOVL    4(AP),R6                        ; Address of descriptor
          57   66    32    1B61  3427          CVTWL   dsc$w_length(R6),R7             ; Number of decimal digits
          58   04 A6 D0    1B64  3428          MOVL    dsc$a_pointer(R6),R8            ; Address of digit string
       6E   68    57    34    1B68  3429 1$:    MOVP    R7,(R8),(SP)                    ; Copy decimal value to stack
                        3C    13    1B6C  3430          BEQL    5$                      ; Special check for zero
          59   57    FF 8F    78    1B6E  3431 2$:    ASHL    #-1,R7,R9               ; Get offset to sign byte
                        3E    13    1B73  3432          BEQL    6$                      ; Leave at least one digit
       6E49    F0 8F    93    1B75  3433          BITB    #^XF0,(SP)[R9]          ; Test high nibble of sign byte
                        12    12    1B7A  3434          BNEQ    3$                      ; Branch if no trailing zeros
             50   57    D0    1B7C  3435          MOVL    R7,R0                   ; Get original number of digits
                        57    D7    1B7F  3436          DECL    R7                      ; Count length down by one
                  08 A6 96    1B81  3437          INCB    dsc$b_scale(R6)         ; Remember scaling has changed
68   57   00   6E   50    FF 8F    F8    1B84  3438          ASHP    #-1,R0,(SP),#0,R7,(R8)  ; Scale number by one digit
                        DA    11    1B8C  3439          BRB     1$                      ; Go repeat test for zeros
          50   59    01    C1    1B8E  3440 3$:    ADDL3   #1,R9,R0                ; Get number of bytes written
          6E   50    00    3B    1B92  3441          SKPC    #0,R0,(SP)              ; Find first non-zero byte
                        50    D7    1B96  3442          DECL    R0                      ; Get <# of bytes> minus 1
          57   50    01    78    1B98  3443          ASHL    #1,R0,R7                ; Convert bytes to nibbles
            61    F0 8F    93    1B9C  3444          BITB    #^XF0,(R1)              ; Check high nibble of byte
                        02    13    1BA0  3445          BEQL    4$                      ; Ignore high nibble if zero
                        57    D6    1BA2  3446          INCL    R7                      ; Otherwise increase length
          68   61    57    34    1BA4  3447 4$:    MOVP    R7,(R1),(R8)            ; Copy over trailing digits
                        09    11    1BA8  3448          BRB     6$
             57   01    D0    1BAA  3449 5$:    MOVL    #1,R7                   ; Special handling if result
                  08 A6 94    1BAD  3450          CLRB    dsc$b_scale(R6)         ;  is zero - scale factor is
             68   0C    D0    1BB0  3451          MOVL    #^X0C,(R8)              ;  zero and sign is positive
             66   57    F7    1BB3  3452 6$:    CVTLW   R7,dsc$w_length(R6)     ; Set length in descriptor
          09 A6   57    F6    1BB6  3453          CVTLB   R7,dsc$b_digits(R6)     ; Insure digit count is set up
             50   56    D0    1BBA  3454          MOVL    R6,R0
                        04    1BBD  3455          RET                             ; All normalized - return
```

```
                     1BBE  3457          .SBTTL  trap_msg_handler
                     1BBE  3458 ;++
                     1BBE  3459 ;ROUTINE TRAP_MSG_HANDLER(SIGARG, MECHARG, ENBLARG): NOVALUE =
                     1BBE  3460 ;
                     1BBE  3461 ; FUNCTION
                     1BBE  3462 ;       This is the handler routine for DBG$PERFORM_OPERATOR routine.  It traps
                     1BBE  3463 ;       overflow (V), decimal overflow (DV), floating underflow (FU), and
                     1BBE  3464 ;       give an informational message to the user program.  It also traps
                     1BBE  3465 ;       the error condition such as divided by zero, or illegal operand type.
                     1BBE  3466 ;
                     1BBE  3467 ; INPUTS
                     1BBE  3468 ;       SIGARG  - The signal argument vector.
                     1BBE  3469 ;
                     1BBE  3470 ;       MECHARG - The mechanism argument vector.
                     1BBE  3471 ;
                     1BBE  3472 ;       ENBLARG - The enable argument vector.
                     1BBE  3473 ;
                     1BBE  3474 ; OUTPUTS
                     1BBE  3475 ;       None.
                     1BBE  3476 ;--
                     1BBE  3477
                     1BBE  3478
                     1BBE  3479 ; Register usage
                     1BBE  3480 ;
                     1BBE  3481 ;       r2      Pointer to operator token entry
                     1BBE  3482 ;       r3      Temporary
                     1BBE  3483
                000C 1BBE  3484 .ENTRY  trap_msg_handler,^M<r2,r3>
                     1BC0  3485
          50   04 AC   D0  1BC0  3486          MOVL    chf$l_sigarglst(AP),r0  ; Pointer to Signal Arguments
          51   08 AC   D0  1BC4  3487          MOVL    chf$l_mcharglst(AP),r1  ; Pointer to Mechanism Arguments
     52   00000000'EF   D0  1BC8  3488          MOVL    save_operator_entry,r2  ; Get the pointer to operator token
                     1BCF  3489                                                  ;   entry from saved area
          52   0C A2   DE  1BCF  3490          MOVAL   token_name(r2),r2       ; Pointer to counted ascii string
                     1BD3  3491
0000048C  8F   04 A0   D1  1BD3  3492 1$:       CMPL    chf$l_sig_name(r0),#ss$_fltovf
                   11   12  1BDB  3493          BNEQ    2$
                   52   DD  1BDD  3494          PUSHL   r2
                   01   DD  1BDF  3495          PUSHL   #1
          00028A02  8F   DD  1BE1  3496          PUSHL   #dbg$_fltovf
     00000000'GF    03   FB  1BE7  3497          CALLS   #3,G^LIB$SIGNAL
                     1BEE  3498
0000049C  8F   04 A0   D1  1BEE  3499 2$:       CMPL    chf$l_sig_name(r0),#ss$_fltund
                   16   12  1BF6  3500          BNEQ    3$
     53   00000004'EF   D0  1BF8  3501          MOVL    save_result,r3
                   63   D4  1BFF  3502          CLRL    (r3)
                   52   DD  1C01  3503          PUSHL   r2
                   01   DD  1C03  3504          PUSHL   #1
          0002869B  8F   DD  1C05  3505          PUSHL   #dbg$_ifltund
                 01D0   31  1C0B  3506          BRW     give_info
                     1C0E  3507
000286A3  8F   04 A0   D1  1C0E  3508 3$:       CMPL    chf$l_sig_name(r0),#dbg$_iintovf
                   0D   12  1C16  3509          BNEQ    4$
                   52   DD  1C18  3510          PUSHL   r2
                   01   DD  1C1A  3511          PUSHL   #1
          000286A3  8F   DD  1C1C  3512          PUSHL   #dbg$_iintovf
                 01B9   31  1C22  3513          BRW     give_info
```

```
                                1C25  3514
00028240 8F    04 A0    D1      1C25  3515 4$:     CMPL    chf$l_sig_name(r0),#dbg$_divbyzero
                        0D      12  1C2D  3516          BNEQ    5$
                 00028240 8F    DD  1C2F  3517          PUSHL   #dbg$_divbyzero
              00000000'GF       01  FB  1C35  3518          CALLS   #1,G^LIB$SIGNAL
                                1C3C  3519
00028EC0 8F    04 A0    D1      1C3C  3520 5$:     CMPL    chf$l_sig_name(r0),#dbg$_sfcntneg
                        0D      12  1C44  3521          BNEQ    6$
                 00028EC0 8F    DD  1C46  3522          PUSHL   #dbg$_sfcntneg
              00000000'GF       01  FB  1C4C  3523          CALLS   #1,G^LIB$SIGNAL
                                1C53  3524
000004A4 8F    04 A0    D1      1C53  3525 6$:     CMPL    chf$l_sig_name(r0),#ss$_decovf
                        11      12  1C5B  3526          BNEQ    7$
                        52      DD  1C5D  3527          PUSHL   r2
                        01      DD  1C5F  3528          PUSHL   #1
                 00028A3A 8F    DD  1C61  3529          PUSHL   #dbg$_decovf
              00000000'GF       03  FB  1C67  3530          CALLS   #3,G^LIB$SIGNAL
                                1C6E  3531
000004BC 8F    04 A0    D1      1C6E  3532 7$:     CMPL    chf$l_sig_name(r0),#ss$_fltdiv_f
                        0D      12  1C76  3533          BNEQ    8$
                 00028240 8F    DD  1C78  3534          PUSHL   #dbg$_divbyzero
              00000000'GF       01  FB  1C7E  3535          CALLS   #1,G^LIB$SIGNAL
                                1C85  3536
000004B4 8F    04 A0    D1      1C85  3537 8$:     CMPL    chf$l_sig_name(r0),#ss$_fltovf_f
                        11      12  1C8D  3538          BNEQ    9$
                        52      DD  1C8F  3539          PUSHL   r2
                        01      DD  1C91  3540          PUSHL   #1
                 00028A02 8F    DD  1C93  3541          PUSHL   #dbg$_fltovf
              00000000'GF       03  FB  1C99  3542          CALLS   #3,G^LIB$SIGNAL
                                1CA0  3543
000004C4 8F    04 A0    D1      1CA0  3544 9$:     CMPL    chf$l_sig_name(r0),#ss$_fltund_f
                        16      12  1CA8  3545          BNEQ    10$
                53   00000004'EF  D0  1CAA  3546          MOVL    save_result,r3
                        63      D4  1CB1  3547          CLRL    (r3)
                        52      DD  1CB3  3548          PUSHL   r2
                        01      DD  1CB5  3549          PUSHL   #1
                 0002869B 8F    DD  1CB7  3550          PUSHL   #dbg$_ifltund
                       011E     31  1CBD  3551          BRW     give_info
                                1CC0  3552
00000484 8F    04 A0    D1      1CC0  3553 10$:    CMPL    chf$l_sig_name(r0),#ss$_intdiv
                        0D      12  1CC8  3554          BNEQ    11$
                 00028240 8F    DD  1CCA  3555          PUSHL   #dbg$_divbyzero
              00000000'GF       01  FB  1CD0  3556          CALLS   #1,G^LIB$SIGNAL
                                1CD7  3557
                                1CD7  3558 11$:
00000494 8F    04 A0    D1      1CD7  3559          CMPL    chf$l_sig_name(r0),#ss$_fltdiv
                        0D      12  1CDF  3560          BNEQ    12$
                 00028240 8F    DD  1CE1  3561          PUSHL   #dbg$_divbyzero
              00000000'GF       01  FB  1CE7  3562          CALLS   #1,G^LIB$SIGNAL
                                1CEE  3563
                                1CEE  3564 12$:
001682C4 8F    04 A0    D1      1CEE  3565          CMPL    chf$l_sig_name(r0),#mth$_floovemat
                        11      12  1CF6  3566          BNEQ    13$
                        52      DD  1CF8  3567          PUSHL   r2
                        01      DD  1CFA  3568          PUSHL   #1
                 00028A02 8F    DD  1CFC  3569          PUSHL   #dbg$_fltovf
              00000000'GF       03  FB  1D02  3570          CALLS   #3,G^LIB$SIGNAL
```

```
                                1D09  3571
                                1D09  3572 13$:
001682CC 8F    04 A0    D1      1D09  3573          CMPL    chf$l_sig_name(r0),#mth$_floundmat
                   16    12      1D11  3574          BNEQ    14$
   53    00000004'EF    D0      1D13  3575          MOVL    save_result,r3
                   63    D4      1D1A  3576          CLRL    (r3)
                   52    DD      1D1C  3577          PUSHL   r2
                   01    DD      1D1E  3578          PUSHL   #1
          0002869B 8F    DD      1D20  3579          PUSHL   #dbg$_ifltund
                 00B5    31      1D26  3580          BRW     give_info
                                1D29  3581
                                1D29  3582 14$:
00168294 8F    04 A0    D1      1D29  3583          CMPL    chf$l_sig_name(r0),#mth$_undexp
                   0D    12      1D31  3584          BNEQ    15$
                   52    DD      1D33  3585          PUSHL   r2
                   01    DD      1D35  3586          PUSHL   #1
          00028ED8 8F    DD      1D37  3587          PUSHL   #dbg$_undexpn
                 009E    31      1D3D  3588          BRW     give_info
                                1D40  3589
                                1D40  3590 15$:
00000454 8F    04 A0    D1      1D40  3591          CMPL    chf$l_sig_name(r0),#ss$_roprand
                   11    12      1D48  3592          BNEQ    16$
                   52    DD      1D4A  3593          PUSHL   r2
                   01    DD      1D4C  3594          PUSHL   #1
          00028A0A'EF    DD      1D4E  3595          PUSHL   dbg$_roprandf
   00000000'GF    03    FB      1D54  3596          CALLS   #3,G^LIB$SIGNAL
                                1D5B  3597
                                1D5B  3598 16$:
000286AB 8F    04 A0    D1      1D5B  3599          CMPL    chf$l_sig_name(r0),#dbg$_istrtru
                   0D    12      1D63  3600          BNEQ    17$
                   52    DD      1D65  3601          PUSHL   r2
                   01    DD      1D67  3602          PUSHL   #1
          000286AB 8F    DD      1D69  3603          PUSHL   #dbg$_istrtru
                 006C    31      1D6F  3604          BRW     give_info
                                1D72  3605
                                1D72  3606 17$:
0000043C 8F    04 A0    D1      1D72  3607          CMPL    chf$l_sig_name(r0),#ss$_opcdec
                   11    12      1D7A  3608          BNEQ    18$
                   52    DD      1D7C  3609          PUSHL   r2
                   01    DD      1D7E  3610          PUSHL   #1
          00028EE0 8F    DD      1D80  3611          PUSHL   #dbg$_opcdec
   00000000'GF    03    FB      1D86  3612          CALLS   #3,G^LIB$SIGNAL
                                1D8D  3613
                                1D8D  3614 18$:
0000047C 8F    04 A0    D1      1D8D  3615          CMPL    chf$l_sig_name(r0),#ss$_intovf
                   11    12      1D95  3616          BNEQ    19$
                   52    DD      1D97  3617          PUSHL   r2
                   01    DD      1D99  3618          PUSHL   #1
          00028A5A 8F    DD      1D9B  3619          PUSHL   #dbg$_intovf
   00000000'GF    03    FB      1DA1  3620          CALLS   #3,G^LIB$SIGNAL
                                1DA8  3621
                                1DA8  3622 19$:
00028EF0 8F    04 A0    D1      1DA8  3623          CMPL    chf$l_sig_name(r0),#dbg$_cvtneguns
                   0D    12      1DB0  3624          BNEQ    20$
                   52    DD      1DB2  3625          PUSHL   r2
                   01    DD      1DB4  3626          PUSHL   #1
          00028EF0 8F    DD      1DB6  3627          PUSHL   #dbg$_cvtneguns
```

```
                    001F   31   1DBC  3628          BRW     give_info
                                1DBF  3629
                                1DBF  3630 20$:
00028ED8 8F    04   A0   D1   1DBF  3631          CMPL    chf$l_sig_name(r0),#dbg$_undexpn
                      0D   12   1DC7  3632          BNEQ    unknown_signal
                      52   DD   1DC9  3633          PUSHL   r2
                      01   DD   1DCB  3634          PUSHL   #1
          00028ED8 8F      DD   1DCD  3635          PUSHL   #dbg$_undexpn
                    0008   31   1DD3  3636          BRW     give_info
                                1DD6  3637
                                1DD6  3638 unknown_signal:
   50   00000918 8F      D0   1DD6  3639          MOVL    #ss$_resignal,r0
                      04   1DDD  3640          RET
                                1DDE  3641
                                1DDE  3642 give_info:
   00000000'GF      03   FB   1DDE  3643          CALLS   #3,G^LIB$SIGNAL
                                1DE5  3644          $UNWIND_S
                      04   1DF0  3645          RET
                                1DF1  3646
```

```
                     1DF1  3648          .SBTTL  dbg$cvt_trfa_to_value
                     1DF1  3649 ;++
                     1DF1  3650 ;GLOBAL ROUTINE DBG$CVT_TRFA_TO_VALUE(DST_ARG, SRC_ARG, TOKENCODE): =
                     1DF1  3651 ;
                     1DF1  3652 ; FUNCTION
                     1DF1  3653 ;       This routine accumulates the value in DST_ARG from SRC_ARG
                     1DF1  3654 ;       depending on given radix.
                     1DF1  3655 ;
                     1DF1  3656 ;       This routine is called from the caller one byte at a time, ie.,
                     1DF1  3657 ;       RFA value initialized to zero.
                     1DF1  3658 ;       DO i FROM 1 to number of characters
                     1DF1  3659 ;          RFA value = RFA value * radix + converted character value(i)
                     1DF1  3660 ;
                     1DF1  3661 ; INPUTS
                     1DF1  3662 ;       DST_ARG - The address of the destination value (6 bytes).
                     1DF1  3663 ;
                     1DF1  3664 ;       SRC_ARG - The address of the source value (6 bytes).
                     1DF1  3665 ;
                     1DF1  3666 ;       TOKENCODE-Token code can be one of TOKEN$K_INTEGER,
                     1DF1  3667 ;              TOKEN$K_HEX_INTEGER, TOKEN$K_BIN_INTEGER,
                     1DF1  3668 ;              TOKEN$K_OCT_INTEGER.
                     1DF1  3669 ;
                     1DF1  3670 ; OUTPUTS
                     1DF1  3671 ;       The result of the conversion is left in the destination value
                     1DF1  3672 ;       address.  Overflow status is returned.
                     1DF1  3673 ;--
                     1DF1  3674
                     1DF1  3675 ; Parameter Offsets
                     1DF1  3676 ;
           00000004  1DF1  3677     dst_arg = 4
           00000008  1DF1  3678     src_arg = 8
           0000000C  1DF1  3679     tokencode = 12
                     1DF1  3680
                     1DF1  3681 ; Register usuage
                     1DF1  3682 ;       r2      Address of the dst_arg
                     1DF1  3683 ;       r3      Address of the src_arg
                     1DF1  3684 ;       r4,r5   quadword tempory result
                     1DF1  3685 ;       r6,r7   quadword tempory result
                     1DF1  3686 ;
                     1DF1  3687
                0FFC 1DF1  3688 .ENTRY  dbg$cvt_trfa_to_value,^M<r2,r3,r4,r5,r6,r7>
52   04 AC    D0  1DF3  3689          MOVL    dst_arg(AP),r2          ; destination address of 6 bytes
53   08 AC    D0  1DF7  3690          MOVL    src_arg(AP),r3          ; source address of 6 bytes
   54   62    D0  1DFB  3691          MOVL    (r2),r4                 ; move the destination value
55   04 A2    3C  1DFE  3692          MOVZWL  4(r2),r5                ;       into quadword (in r4,r5)
                     1E02  3693
04   0C AC    D1  1E02  3694          CMPL    tokencode(AP),#token$k_integer
        15    13  1E06  3695          BEQL    shift_decimal$
05   0C AC    D1  1E08  3696          CMPL    tokencode(AP),#token$k_hex_integer
        20    13  1E0C  3697          BEQL    shift_hexdecimal$
0A   0C AC    D1  1E0E  3698          CMPL    tokencode(AP),#token$k_bin_integer
        21    13  1E12  3699          BEQL    shift_binary$
0B   0C AC    D1  1E14  3700          CMPL    tokencode(AP),#token$k_oct_integer
        22    13  1E18  3701          BEQL    shift_octal$
      0044    31  1E1A  3702          BRW     report_error$
                     1E1D  3703
                     1E1D  3704 shift_decimal$:                         ; Multiply the value by 10
```

```
                56   54    01   79  1E1D  3705          ASHQ    #1,r4,r6                        ;   multiply r4 by 2
                54   54    03   79  1E21  3706          ASHQ    #3,r4,r4                        ;   multiply r4 by 8
                     54    56   C0  1E25  3707          ADDL2   r6,r4                           ;   add result (= * 10)
                     55    57   D8  1E28  3708          ADWC    r7,r5                           ; Cannot be a overflow of 8
                                    1E2B  3709                                                  ;   bytes, it'll overflow 6
                                    1E2B  3710                                                  ;   bytes.
                         0015   31  1E2B  3711          BRW     set_result$
                                    1E2E  3712
                                    1E2E  3713 shift_hexdecimal$:                               ; Multiply the value by 16
                54   54    04   79  1E2E  3714          ASHQ    #4,r4,r4
                         000E   31  1E32  3715          BRW     set_result$
                                    1E35  3716
                                    1E35  3717 shift_binary$:                                   ; Multiply the value by 2
                54   54    01   79  1E35  3718          ASHQ    #1,r4,r4
                         0007   31  1E39  3719          BRW     set_result$
                                    1E3C  3720
                                    1E3C  3721 shift_octal$:                                    ; Multiply the value by 8
                54   54    03   79  1E3C  3722          ASHQ    #3,r4,r4
                         0000   31  1E40  3723          BRW     set_result$
                                    1E43  3724
                                    1E43  3725 set_result$:
                     62    54   D0  1E43  3726          MOVL    r4,(r2)                         ; Move it into destination
                  04 A2    55   B0  1E46  3727          MOVW    r5,4(r2)                        ;   (RFA)
           55  0000FFFF   8F   CA  1E4A  3728          BICL    #^XFFFF,r5                      ; Test the high 2 bytes for
                     24   12  1E51  3729          BNEQ    rfa_overflow$                   ;   overflow
                     62   63   A0  1E53  3730          ADDW    (r3),(r2)                       ; Add the character value
                02 A2    02   A3  D8  1E56  3731          ADWC    2(r3),2(r2)                     ;   into RFA (6 bytes addition)
                     1A   1F  1E5B  3732          BCS     rfa_overflow$                   ; Check for overflow
                     50   01   D0  1E5D  3733          MOVL    #1,r0
                               04  1E60  3734          RET
                                    1E61  3735
                                    1E61  3736 report_error$:
                00000035'EF    DF  1E61  3737          PUSHAL  errmsg1
                         01    DD  1E67  3738          PUSHL   #1
                00028362 8F    DD  1E69  3739          PUSHL   #dbg$_interr
           00000000'GF   03    FB  1E6F  3740          CALLS   #3,G^LIB$SIGNAL
                               04  1E76  3741          RET
                                    1E77  3742
                                    1E77  3743 rfa_overflow$:
                         50    D4  1E77  3744          CLRL    r0
                               04  1E79  3745          RET
                                    1E7A  3746
```

```
                     1E7A  3748          .SBTTL  dbg$cvt_tquadword_to_value
                     1E7A  3749 ;++
                     1E7A  3750 ;GLOBAL ROUTINE DBG$CVT_TQUADWORD_TO_VALUE(DST_ARG, SRC_ARG, TOKENCODE): =
                     1E7A  3751 ;
                     1E7A  3752 ; FUNCTION
                     1E7A  3753 ;       This routine accumulates the value in DST_ARG from SRC_ARG
                     1E7A  3754 ;       depending on given radix.
                     1E7A  3755 ;
                     1E7A  3756 ;       This routine is called from the caller one byte at a time, ie.,
                     1E7A  3757 ;       QUADWORD value initialized to zero.
                     1E7A  3758 ;       DO i FROM 1 to number of characters
                     1E7A  3759 ;          QUADWORD value = QUADWORD value * radix + converted character value(i)
                     1E7A  3760 ;
                     1E7A  3761 ;       NOTE: This routine assuming the dtype is Q, not QU, if dtype is QU,
                     1E7A  3762 ;       then DBG$CVT_TUQUADWORD_TO_VALUE should be called instead.
                     1E7A  3763 ;
                     1E7A  3764 ; INPUTS
                     1E7A  3765 ;       DST_ARG - The address of the destination value (8 bytes).
                     1E7A  3766 ;
                     1E7A  3767 ;       SRC_ARG - The address of the source value (8 bytes).
                     1E7A  3768 ;
                     1E7A  3769 ;       TOKENCODE-Token code can be one of TOKEN$K_INTEGER,
                     1E7A  3770 ;               TOKEN$K_HEX_INTEGER, TOKEN$K_BIN_INTEGER,
                     1E7A  3771 ;               TOKEN$K_OCT_INTEGER.
                     1E7A  3772 ;
                     1E7A  3773 ; OUTPUTS
                     1E7A  3774 ;       The result of the conversion is left in the destination value
                     1E7A  3775 ;       address.  Overflow status is returned.
                     1E7A  3776 ;--
                     1E7A  3777
                     1E7A  3778 ; Parameter Offsets
                     1E7A  3779 ;
          00000004   1E7A  3780     dst_arg = 4
          00000008   1E7A  3781     src_arg = 8
          0000000C   1E7A  3782     tokencode = 12
                     1E7A  3783
                     1E7A  3784 ; Register usuage
                     1E7A  3785 ;       r2      Address of the dst_arg
                     1E7A  3786 ;       r3      Address of the src_arg
                     1E7A  3787 ;       r4,r5   quadword tempory result
                     1E7A  3788 ;       r6,r7   quadword tempory result
                     1E7A  3789 ;
                     1E7A  3790
             00FC    1E7A  3791 .ENTRY  dbg$cvt_tquadword_to_value,^M<r2,r3,r4,r5,r6,r7>
52   04 AC    D0     1E7C  3792         MOVL    dst_arg(AP),r2          ; destination address of 8 bytes
53   08 AC    D0     1E80  3793         MOVL    src_arg(AP),r3          ; source address of 8 bytes
   54   62    7D     1E84  3794         MOVQ    (r2),r4                 ; move the destination value
                     1E87  3795                                         ;       into quadword (in r4,r5)
                     1E87  3796
04   0C AC    D1     1E87  3797         CMPL    tokencode(AP),#token$k_integer
        15    13     1E8B  3798         BEQL    shift_decimal1$
05   0C AC    D1     1E8D  3799         CMPL    tokencode(AP),#token$k_hex_integer
        26    13     1E91  3800         BEQL    shift_hexdecimal1$
0A   0C AC    D1     1E93  3801         CMPL    tokencode(AP),#token$k_bin_integer
        29    13     1E97  3802         BEQL    shift_binary1$
0B   0C AC    D1     1E99  3803         CMPL    tokencode(AP),#token$k_oct_integer
        2C    13     1E9D  3804         BEQL    shift_octal1$
```

```
                0043   31   1E9F  3805          BRW     report_error1$
                       1EA2  3806
                       1EA2  3807 shift_decimal1$:                        ; Multiply the value by 10
          56    54   01   79  1EA2  3808          ASHQ    #1,r4,r6          ;   multiply r4 by 2
                  53   1D  1EA6  3809          BVS     quad_overflow$
          54    54   03   79  1EA8  3810          ASHQ    #3,r4,r4          ;   multiply r4 by 8
                  4D   1D  1EAC  3811          BVS     quad_overflow$
                54   56   C0  1EAE  3812          ADDL2   r6,r4             ;   add result (= * 10)
                55   57   D8  1EB1  3813          ADWC    r7,r5             ; Add with carry
                  45   1D  1EB4  3814          BVS     quad_overflow$    ; Branch on signed overflow set
                001B   31  1EB6  3815          BRW     set_result1$
                       1EB9  3816
                       1EB9  3817 shift_hexdecimal1$:                    ; Multiply the value by 16
          54    54   04   79  1EB9  3818          ASHQ    #4,r4,r4
                  3C   1D  1EBD  3819          BVS     quad_overflow$
                0012   31  1EBF  3820          BRW     set_result1$
                       1EC2  3821
                       1EC2  3822 shift_binary1$:                        ; Multiply the value by 2
          54    54   01   79  1EC2  3823          ASHQ    #1,r4,r4
                  33   1D  1EC6  3824          BVS     quad_overflow$
                0009   31  1EC8  3825          BRW     set_result1$
                       1ECB  3826
                       1ECB  3827 shift_octal1$:                         ; Multiply the value by 8
          54    54   03   79  1ECB  3828          ASHQ    #3,r4,r4
                  2A   1D  1ECF  3829          BVS     quad_overflow$
                0000   31  1ED1  3830          BRW     set_result1$
                       1ED4  3831
                       1ED4  3832 set_result1$:
                62   54   7D  1ED4  3833          MOVQ    r4,(r2)           ; Move it into destination
                       1ED7  3834                                        ;   (QUADWORD)
                62   63   C0  1ED7  3835          ADDL2   (r3),(r2)         ; Add the character value
          04 A2   04   A3   D8  1EDA  3836          ADWC    4(r3),4(r2)       ;   into QUADWORD (8 bytes addition)
                  1A   1D  1EDF  3837          BVS     quad_overflow$    ; Check for overflow
                50   01   D0  1EE1  3838          MOVL    #1,r0
                       04  1EE4  3839          RET
                       1EE5  3840
                       1EE5  3841 report_error1$:
      00000067'EF      DF  1EE5  3842          PUSHAL  errmsg2
                01     DD  1EEB  3843          PUSHL   #1
          00028362     8F  1EED  3844          PUSHL   #dbg$_interr
00000000'GF    03     FB  1EF3  3845          CALLS   #3,G^LIB$SIGNAL
                       04  1EFA  3846          RET
                       1EFB  3847
                       1EFB  3848 quad_overflow$:
                50   D4  1EFB  3849          CLRL    r0
                       04  1EFD  3850          RET
```

```
                        1EFE  3852          .SBTTL  dbg$cvt_tuquadword_to_value
                        1EFE  3853 ;++
                        1EFE  3854 ;GLOBAL ROUTINE DBG$CVT_TUQUADWORD_TO_VALUE(DST_ARG, SRC_ARG, TOKENCODE): =
                        1EFE  3855 ;
                        1EFE  3856 ; FUNCTION
                        1EFE  3857 ;       This routine accumulates the value in DST_ARG from SRC_ARG
                        1EFE  3858 ;       depending on given radix.
                        1EFE  3859 ;
                        1EFE  3860 ;       This routine is called from the caller one byte at a time, ie.,
                        1EFE  3861 ;       QUADWORD value initialized to zero.
                        1EFE  3862 ;       DO i FROM 1 to number of characters
                        1EFE  3863 ;          QUADWORD value = QUADWORD value * radix + converted character value(i)
                        1EFE  3864 ;
                        1EFE  3865 ;       NOTE: This routine assuming the dtype is QU, not Q, if dtype is Q,
                        1EFE  3866 ;       then DBG$CVT_TQUADWORD_TO_VALUE should be called instead.
                        1EFE  3867 ;
                        1EFE  3868 ; INPUTS
                        1EFE  3869 ;       DST_ARG - The address of the destination value (8 bytes).
                        1EFE  3870 ;
                        1EFE  3871 ;       SRC_ARG - The address of the source value (8 bytes).
                        1EFE  3872 ;
                        1EFE  3873 ;       TOKENCODE-Token code can be one of TOKEN$K_INTEGER,
                        1EFE  3874 ;               TOKEN$K_HEX_INTEGER, TOKEN$K_BIN_INTEGER,
                        1EFE  3875 ;               TOKEN$K_OCT_INTEGER.
                        1EFE  3876 ;
                        1EFE  3877 ; OUTPUTS
                        1EFE  3878 ;       The result of the conversion is left in the destination value
                        1EFE  3879 ;       address.  Overflow status is returned.
                        1EFE  3880 ;--
                        1EFE  3881
                        1EFE  3882 ; Parameter Offsets
                        1EFE  3883 ;
           00000004     1EFE  3884     dst_arg = 4
           00000008     1EFE  3885     src_arg = 8
           0000000C     1EFE  3886     tokencode = 12
                        1EFE  3887
                        1EFE  3888 ; Register usuage
                        1EFE  3889 ;       r2      Address of the dst_arg
                        1EFE  3890 ;       r3      Address of the src_arg
                        1EFE  3891 ;       r4,r5   quadword tempory result
                        1EFE  3892 ;       r6,r7   quadword tempory result
                        1EFE  3893 ;
                        1EFE  3894
             00FC       1EFE  3895 .ENTRY  dbg$cvt_tuquadword_to_value,^M<r2,r3,r4,r5,r6,r7>
52     04 AC    D0      1F00  3896        MOVL    dst_arg(AP),r2          ; destination address of 8 bytes
53     08 AC    D0      1F04  3897        MOVL    src_arg(AP),r3          ; source address of 8 bytes
   54     62    7D      1F08  3898        MOVQ    (r2),r4                 ; move the destination value
                        1F0B  3899                                        ;       into quadword (in r4,r5)
                        1F0B  3900
04     0C AC    D1      1F0B  3901        CMPL    tokencode(AP),#token$k_integer
          15    13      1F0F  3902        BEQL    shift_decimalu1$
05     0C AC    D1      1F11  3903        CMPL    tokencode(AP),#token$k_hex_integer
          22    13      1F15  3904        BEQL    shift_hexdecimalu1$
0A     0C AC    D1      1F17  3905        CMPL    tokencode(AP),#token$k_bin_integer
          23    13      1F1B  3906        BEQL    shift_binaryu1$
0B     0C AC    D1      1F1D  3907        CMPL    tokencode(AP),#token$k_oct_integer
          24    13      1F21  3908        BEQL    shift_octalu1$
```

```
                0039   31  1F23  3909          BRW     report_erroru1$
                       1F26  3910
                       1F26  3911 shift_decimalu1$:                    ; Multiply the value by 10
56   54   01    79     1F26  3912          ASHQ    #1,r4,r6            ;   multiply r4 by 2
54   54   03    79     1F2A  3913          ASHQ    #3,r4,r4            ;   multiply r4 by 8
     54   56    C0     1F2E  3914          ADDL2   r6,r4               ;   add result (= * 10)
     55   57    D8     1F31  3915          ADWC    r7,r5               ; Add with carry
          3F    1F     1F34  3916          BCS     uquad_overflow$     ; Branch on unsigned overflow set
              0015   31  1F36  3917          BRW     set_resultu1$
                       1F39  3918
                       1F39  3919 shift_hexdecimalu1$:                 ; Multiply the value by 16
54   54   04    79     1F39  3920          ASHQ    #4,r4,r4
              000E   31  1F3D  3921          BRW     set_resultu1$
                       1F40  3922
                       1F40  3923 shift_binaryu1$:                     ; Multiply the value by 2
54   54   01    79     1F40  3924          ASHQ    #1,r4,r4
              0007   31  1F44  3925          BRW     set_resultu1$
                       1F47  3926
                       1F47  3927 shift_octalu1$:                      ; Multiply the value by 8
54   54   03    79     1F47  3928          ASHQ    #3,r4,r4
              0000   31  1F4B  3929          BRW     set_resultu1$
                       1F4E  3930
                       1F4E  3931 set_resultu1$:
     62   54    7D     1F4E  3932          MOVQ    r4,(r2)             ; Move it into destination
                       1F51  3933                                     ;   (QUADWORD)
     62   63    C0     1F51  3934          ADDL2   (r3),(r2)           ; Add the character value
04 A2  04 A3    D8     1F54  3935          ADWC    4(r3),4(r2)         ;   into QUADWORD (8 bytes addition)
          1A    1F     1F59  3936          BCS     uquad_overflow$     ; Check for overflow
     50   01    D0     1F5B  3937          MOVL    #1,r0
                04     1F5E  3938          RET
                       1F5F  3939
                       1F5F  3940 report_erroru1$:
0000009E'EF     DF     1F5F  3941          PUSHAL  errmsg3
          01    DD     1F65  3942          PUSHL   #1
00028362  8F    DD     1F67  3943          PUSHL   #dbg$_interr
00000000'GF     03     FB    1F6D  3944          CALLS   #3,G^LIB$SIGNAL
                04     1F74  3945          RET
                       1F75  3946
                       1F75  3947 uquad_overflow$:
               50   D4     1F75  3948          CLRL    r0
                04     1F77  3949          RET
```

```
                                        .SBTTL  dbg$cvt_toctaword_to_value
                                ;++
                                ;GLOBAL ROUTINE DBG$CVT_TOCTAWORD_TO_VALUE(DST_ARG, SRC_ARG, TOKENCODE): =
                                ;
                                ; FUNCTION
                                ;       This routine accumulates the value in DST_ARG from SRC_ARG
                                ;       depending on given radix.
                                ;
                                ;       This routine is called from the caller one byte at a time, ie.,
                                ;       OCTAWORD value initialized to zero.
                                ;       DO i FROM 1 to number of characters
                                ;          OCTAWORD value = OCTAWORD value * radix + converted character value(i)
                                ;
                                ;       This routine is used for signed octaword.
                                ;
                                ; INPUTS
                                ;       DST_ARG - The address of the destination value (16 bytes).
                                ;
                                ;       SRC_ARG - The address of the source value (16 bytes).
                                ;
                                ;       TOKENCODE-Token code can be one of TOKEN$K_INTEGER,
                                ;               TOKEN$K_HEX_INTEGER, TOKEN$K_BIN_INTEGER,
                                ;               TOKEN$K_OCT_INTEGER.
                                ;
                                ; OUTPUTS
                                ;       The result of the conversion is left in the destination value
                                ;       address.  Overflow status is returned.
                                ;++
                                ; Parameter Offsets
                                ;
                        00000004    dst_arg = 4
                        00000008    src_arg = 8
                        0000000C    tokencode = 12
                                ; Register usuage
                                ;       r2      Address of the dst_arg
                                ;       r3      Address of the src_arg
                                ;
                          0FFC  .ENTRY  dbg$cvt_toctaword_to_value,^M<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                    52    04 AC   D0    MOVL    dst_arg(AP),r2          ; destination address of 16 bytes
                    53    08 AC   D0    MOVL    src_arg(AP),r3          ; source address of 16 bytes
          0000005C'EF     62   7D       MOVQ    (r2),work_octa          ; move the destination value
                                                                        ;       into work_octa
      00000064'EF   08 A2   7D          MOVQ    8(r2),work_octa+8
                       54 7CFD          CLRQ    r4
                       58 7CFD          CLRQ    r8
                 04    0C AC   D1       CMPL    tokencode(AP),#token$k_integer
                       15   13          BEQL    decimal$
                 05    0C AC   D1       CMPL    tokencode(AP),#token$k_hex_integer
                       12   13          BEQL    hexdecimal$
                 0A    0C AC   D1       CMPL    tokencode(AP),#token$k_bin_integer
                       0F   13          BEQL    binary$
                 0B    0C AC   D1       CMPL    tokencode(AP),#token$k_oct_integer
                       0C   13          BEQL    octal$
                      0164  31          BRW     report_error2$
```

```
                                 1FB2  4008 decimal$:
                          0009   31  1FB2  4009          BRW     shift_decimal2$
                                 1FB5  4010 hexdecimal$:
                          0088   31  1FB5  4011          BRW     shift_hexdecimal2$
                                 1FB8  4012 binary$:
                          00C2   31  1FB8  4013          BRW     shift_binary2$
                                 1FBB  4014 octal$:
                          00FC   31  1FBB  4015          BRW     shift_octal2$
                                 1FBE  4016
                                 1FBE  4017 shift_decimal2$:                     ; Multiply the value by 10
                                 1FBE  4018
    54  1F  01  0000005C'EF  F0   1FBE  4019          INSV    work_octa,#1,#31,r4     ;   multiply work_octa by 2
                                 1FC7  4020                                      ; Move integer to bit-field
                                 1FC7  4021                                      ;   move from work_octa into
                                 1FC7  4022                                      ;   pos 1, length 31 of r4
                                 1FC7  4023                                      ;   r4 is zero out first,
                                 1FC7  4024                                      ;   so bit 0 of r4 is zero.
    55  0000005C'EF  20  1F  EF   1FC7  4025          EXTZV   #31,#32,work_octa,r5    ; Move bit-field into integer
                                 1FD0  4026                                      ;   move from work_octa
                                 1FD0  4027                                      ;   pos 31, length 32 to r5
    56  0000005C'EF  20  3F  EF   1FD0  4028          EXTZV   #63,#32,work_octa,r6    ; Move next 32 bits into r6
0000005C'EF  20  0000005F 8F  EF   1FD9  4029          EXTZV   #95,#32,work_octa,r7    ; Move next 32 bits into r7
                          57       1FE5
0000005C'EF  01  0000007F 8F  EF   1FE6  4030          EXTZV   #127,#1,work_octa,r0    ; Check to see if the leftover
                          50       1FF2
                          48   12  1FF3  4031          BNEQ    decimal_ov$             ;   bits are zero, if not,
                                 1FF5  4032                                      ;   we have overflow.
    58  1D  03  0000005C'EF  F0   1FF5  4033          INSV    work_octa,#3,#29,r8     ;   multiply work_octa by 8
    59  0000005C'EF  20  1D  EF   1FFE  4034          EXTZV   #29,#32,work_octa,r9
    5A  0000005C'EF  20  3D  EF   2007  4035          EXTZV   #61,#32,work_octa,r10
0000005C'EF  20  0000005D 8F  EF   2010  4036          EXTZV   #93,#32,work_octa,r11
                          5B       201C
0000005C'EF  03  0000007D 8F  EF   201D  4037          EXTZV   #125,#3,work_octa,r0
                          50       2029
                          11   12  202A  4038          BNEQ    decimal_ov$
                     5B   54   C0  202C  4039          ADDL2   r4,r8                   ; Add two *2 + *8 = *10
                     59   55   D8  202F  4040          ADWC    r5,r9
                     5A   56   D8  2032  4041          ADWC    r6,r10
                     5B   57   D8  2035  4042          ADWC    r7,r11                  ; Add with carry
                          03   1D  2038  4043          BVS     decimal_ov$             ; Branch on signed overflow set
                      00BA   31  203A  4044          BRW     set_result2$
                                 203D  4045 decimal_ov$:
                      00EC   31  203D  4046          BRW     octa_overflow$
                                 2040  4047
                                 2040  4048 shift_hexdecimal2$:                   ; Multiply the value by 16
    58  1C  04  0000005C'EF  F0   2040  4049          INSV    work_octa,#4,#28,r8     ;
    59  0000005C'EF  20  1C  EF   2049  4050          EXTZV   #28,#32,work_octa,r9
    5A  0000005C'EF  20  3C  EF   2052  4051          EXTZV   #60,#32,work_octa,r10
0000005C'EF  20  0000005C 8F  EF   205B  4052          EXTZV   #92,#32,work_octa,r11
                          5B       2067
0000005C'EF  04  0000007C 8F  EF   2068  4053          EXTZV   #124,#4,work_octa,r0
                          50       2074
                          03   12  2075  4054          BNEQ    hexdecimal_ov$
                      007D   31  2077  4055          BRW     set_result2$
                                 207A  4056 hexdecimal_ov$:
                      00AF   31  207A  4057          BRW     octa_overflow$
                                 207D  4058
```

```
                                    207D  4059 shift_binary2$:                                 ; Multiply the value by 2
    58   1F   01   0000005C'EF   F0  207D  4060          INSV    work_octa,#1,#31,r8           ;
    59   0000005C'EF   20   1F   EF  2086  4061          EXTZV   #31,#32,work_octa,r9
    5A   0000005C'EF   20   3F   EF  208F  4062          EXTZV   #63,#32,work_octa,r10
0000005C'EF   20   0000005F 8F   EF  2098  4063          EXTZV   #95,#32,work_octa,r11
                           5B        20A4
0000005C'EF   01   0000007F 8F   EF  20A5  4064          EXTZV   #127,#1,work_octa,r0
                           50        20B1
                           03   12   20B2  4065          BNEQ    binary_ov$
                      0040 31   20B4  4066          BRW     set_result2$
                                    20B7  4067 binary_ov$:
                      0072 31   20B7  4068          BRW     octa_overflow$
                                    20BA  4069
                                    20BA  4070 shift_octal2$:                                  ; Multiply the value by 8
    58   1D   03   0000005C'EF   F0  20BA  4071          INSV    work_octa,#3,#29,r8           ;
    59   0000005C'EF   20   1D   EF  20C3  4072          EXTZV   #29,#32,work_octa,r9
    5A   0000005C'EF   20   3D   EF  20CC  4073          EXTZV   #61,#32,work_octa,r10
0000005C'EF   20   0000005D 8F   EF  20D5  4074          EXTZV   #93,#32,work_octa,r11
                           5B        20E1
0000005C'EF   03   0000007D 8F   EF  20E2  4075          EXTZV   #125,#3,work_octa,r0
                           50        20EE
                           03   12   20EF  4076          BNEQ    octal_ov$
                      0003 31   20F1  4077          BRW     set_result2$
                                    20F4  4078 octal_ov$:
                      0035 31   20F4  4079          BRW     octa_overflow$
                                    20F7  4080
                                    20F7  4081 set_result2$:
              62   58   7D   20F7  4082          MOVQ    r8,(r2)                               ; Move it into destination
                                    20FA  4083                                                ;   (OCTAWORD)
        08 A2   5A   7D   20FA  4084          MOVQ    r10,8(r2)
              62   63   C0   20FE  4085          ADDL2   (r3),(r2)                             ; Add the character value
     04 A2   04 A3   D8   2101  4086          ADWC    4(r3),4(r2)                           ;   into OCTAWORD (16 bytes addition
     08 A2   08 A3   D8   2106  4087          ADWC    8(r3),8(r2)
     0C A2   0C A3   D8   210B  4088          ADWC    12(r3),12(r2)
                     1A   1D   2110  4089          BVS     octa_overflow$                        ; Check for overflow
              50   01   D0   2112  4090          MOVL    #1,r0
                           04   2115  4091          RET
                                    2116  4092
                                    2116  4093 report_error2$:
          000000D6'EF      DF   2116  4094          PUSHAL  errmsg4
                           01   DD   211C  4095          PUSHL   #1
                00028362   8F   DD   211E  4096          PUSHL   #dbg$_interr
00000000'GF                03   FB   2124  4097          CALLS   #3,G^LIB$SIGNAL
                           04   212B  4098          RET
                                    212C  4099
                                    212C  4100 octa_overflow$:
                     50    D4   212C  4101          CLRL    r0
                           04   212E  4102          RET
                                    212F  4103
                                    212F  4104          .END
```

```
ABS_B                           00001030 R      04
ABS_B_NEGATE                    00001038 R      04
ABS_D                           00001060 R      04
ABS_D_NEGATE                    00001068 R      04
ABS_F                           00001054 R      04
ABS_FIXED                       00001A2A R      04
ABS_F_NEGATE                    0000105C R      04
ABS_G                           0000106C R      04
ABS_G_NEGATE                    00001076 R      04
ABS_H                           0000107B R      04
ABS_H_NEGATE                    00001085 R      04
ABS_L                           00001048 R      04
ABS_L_NEGATE                    00001050 R      04
ABS_W                           0000103C R      04
ABS_W_NEGATE                    00001044 R      04
ADDP$                           00001451 R      04
ADDRESS_L                       000012D7 R      04
ADD_B$                          0000031A R      04
ADD_BU$                         00000324 R      04
ADD_BU_BU                       0000031D R      04
ADD_B_B                         00000313 R      04
ADD_DC_DC                       00000371 R      04
ADD_D_D                         00000354 R      04
ADD_FC_FC                       00000365 R      04
ADD_FIXED_FIXED                 00001A3E R      04
ADD_F_F                         0000034F R      04
ADD_GC_GC                       0000037D R      04
ADD_G_G                         00000359 R      04
ADD_HC_HC                       0000038B R      04
ADD_H_H                         0000035F R      04
ADD_L$                          00000342 R      04
ADD_LU$                         0000034C R      04
ADD_LU_LU                       00000345 R      04
ADD_L_L                         0000033B R      04
ADD_P_P                         0000143C R      04
ADD_TPTR_L                      000012FC R      04
ADD_W$                          0000032E R      04
ADD_WU$                         00000338 R      04
ADD_WU_WU                       00000331 R      04
ADD_W_W                         00000327 R      04
AND_B_B                         00000EF3 R      04
AND_D_D                         00000F17 R      04
AND_L_L                         00000F0B R      04
AND_W_W                         00000EFF R      04
ARGT_DESC                       00000030 R      03
ARG1_VALDESC                    00000034 R      03
ARG2_DESC                       00000038 R      03
ARG2_VALDESC                    0000003C R      03
BASIC$                          0000030F R      04
BGN                             00000075 R      04
BINARY$                         00001FB8 R      04
BINARY_OV$                      000020B7 R      04
BITPOS                        = 00000002
BITSELECT                       000010AD R      04
BIT_AND_B_B                     00000E81 R      04
BIT_AND_L_L                     00000E91 R      04
BIT_AND_TF_TF                   000018C3 R      04
BIT_AND_W_W                     00000E89 R      04
BIT_EQV_B_B                     00000E99 R      04
BIT_EQV_L_L                     00000EA9 R      04
BIT_EQV_W_W                     00000EA1 R      04
BIT_IMP_B_B                     00000EDB R      04
BIT_IMP_L_L                     00000EEB R      04
BIT_IMP_W_W                     00000EE3 R      04
BIT_NOT_B                       00000EB1 R      04
BIT_NOT_L                       00000EB9 R      04
BIT_NOT_TF_TF                   000018AD R      04
BIT_NOT_W                       00000EB5 R      04
BIT_OR_B_B                      00000EBD R      04
BIT_OR_L_L                      00000EC7 R      04
BIT_OR_TF_TF                    000018E1 R      04
BIT_OR_W_W                      00000EC2 R      04
BIT_XOR_B_B                     00000ECC R      04
BIT_XOR_L_L                     00000ED6 R      04
BIT_XOR_W_W                     00000ED1 R      04
BRANCH_RET                      00000305 R      04
CHF$L_MCHARGLST               = 00000008
CHF$L_SIGARGLST               = 00000004
CHF$L_SIG_NAME                = 00000004
CONCAT_OK$                      00000A50 R      04
CONCAT_T$                       00000A30 R      04
CONCAT_TF_TF                    00001783 R      04
CONCAT_T_T                      00000A11 R      04
DBG$ABS_FIXED                   ******** X      00
DBG$ADD_FIXED_FIXED             ******** X      00
DBG$BLISS_BITSELECT             ******** X      00
DBG$BLISS_INDIRECTION           ******** X      00
DBG$B_BPT_INS                   00000060
DBG$B_PREV_PRO1                 0000005E
DBG$B_PREV_PRO2                 0000005F
DBG$B_USER_OPCO                 00000040
DBG$CVT_TOCTAWORD_TO_VALUE      00001F78 RG     04
DBG$CVT_TQUADWORD_TO_VALUE      00001E7A RG     04
DBG$CVT_TRFA_TO_VALUE           00001DF1 RG     04
DBG$CVT_TUQUADWORD_TO_VALUE     00001EFE RG     04
DBG$C_ADDRESS_OF                ******** X      00
DBG$C_ADD_TPTR_L                ******** X      00
DBG$C_INDIRECTION               ******** X      00
DBG$C_PRE_DECR_TPTR             ******** X      04
DBG$C_PRE_INCR_TPTR             ******** X      04
DBG$C_RUNFR_LEN                 00000065
DBG$C_SIZEOF                    ******** X      00
DBG$C_SUB_TPTR_L                ******** X      00
DBG$C_SUB_TPTR_TPTR             ******** X      00
DBG$DIV_FIXED_FIXED             ******** X      00
DBG$EQL_FIXED_FIXED             ******** X      00
DBG$GB_LANGUAGE                 ******** X      00
DBG$GEQ_FIXED_FIXED             ******** X      00
DBG$GTR_FIXED_FIXED             ******** X      00
DBG$K_BASIC                   = 00000004
DBG$K_PLI_ABIT                = 0000000E
DBG$K_PLI_UBIT                = 0000000C
DBG$K_RUNFR_LEN                 00000065
DBG$LEQ_FIXED_FIXED             ******** X      00
```

```
DBG$LSS_FIXED_FIXED             ********   X   00
DBG$L_BPT_PC                    0000004A
DBG$L_CALL_ADDR                 00000052
DBG$L_FRAME_PTR                 0000004E
DBG$L_NEXT_LINK                 00000000
DBG$L_SAVE_FLD                  00000061
DBG$L_USER_AP                   00000034
DBG$L_USER_FP                   00000038
DBG$L_USER_PC                   00000040
DBG$L_USER_PSL                  00000044
DBG$L_USER_R0                   00000004
DBG$L_USER_R1                   00000008
DBG$L_USER_R10                  0000002C
DBG$L_USER_R11                  00000030
DBG$L_USER_R2                   0000000C
DBG$L_USER_R3                   00000010
DBG$L_USER_R4                   00000014
DBG$L_USER_R5                   00000018
DBG$L_USER_R6                   0000001C
DBG$L_USER_R7                   00000020
DBG$L_USER_R8                   00000024
DBG$L_USER_R9                   00000028
DBG$L_USER_REGS                 00000004
DBG$L_USER_SP                   0000003C
DBG$L_WATCHPT                   00000056
DBG$L_WATCHPTEN                 0000005A
DBG$MOL_FIXED_FIXED             ********   X   00
DBG$NEQ_FIXED_FIXED             ********   X   00
DBG$PERFORM_OPERATOR            00000000 RG    04
DBG$PRED_ENUM                   ********   X   00
DBG$STRIP_ZEROES                00001B58 RG    04
DBG$SUB_FIXED_FIXED             ********   X   00
DBG$SUCC_ENUM                   ********   X   00
DBG$UNARY_MINUS_FIXED           ********   X   00
DBG$UNARY_PLUS_FIXED            ********   X   00
DBG$W_RUN_STAT                  00000048
DBG$_CPOSTDECR                = 0002873B
DBG$_CPOSTINCR                = 0002872B
DBG$_CPREDECR                 = 00028733
DBG$_CPREINCR                 = 00028723
DBG$_CVTNEGUNS                = 00028EF0
DBG$_DECOVF                   = 00028A3A
DBG$_DIVBYZERO                = 00028240
DBG$_FLTOVF                   = 00028A02
DBG$_IFLTUND                  = 0002869B
DBG$_IINTOVF                  = 000286A3
DBG$_INTERR                   = 00028362
DBG$_INTOVF                   = 00028A5A
DBG$_ISTRTRU                  = 000286AB
DBG$_OPCDEC                   = 00028EE0
DBG$_ROPRANDF                 = 00028A0A
DBG$_SFCNTNEG                 = 00028EC0
DBG$_STRNGPAD                 = 0002866B
DBG$_UNDEXPN                  = 00028ED8
DECIMAL$                        00001FB2 R     04
DECIMAL_OV$                     0000203D R     04
DIFFERENCE_SET1$                000010E4 R     04
DIFFERENCE_SET2$                000010F6 R     04
DIFFERENCE_SET3$                000010FA R     04
DIFFERENCE_SET4$                00001104 R     04
DIFFERENCE_SET_RET$             00001117 R     04
DIFFERENCE_SET_SET              000010C4 R     04
DIV_B$                          00000426 R     04
DIV_BU$                         00000439 R     04
DIV_BU_BU                       00000429 R     04
DIV_BY_ZERO                     000002CD R     04
DIV_B_B                         0000041F R     04
DIV_D$                          00000548 R     04
DIV_DC_DC                       000004DA R     04
DIV_D_D                         00000490 R     04
DIV_F$                          000004D7 R     04
DIV_FC_FC                       000004A1 R     04
DIV_FIXED_FIXED                 00001A8C R     04
DIV_F_F                         0000048B R     04
DIV_G$                          000005C7 R     04
DIV_GC_GC                       0000054B R     04
DIV_G_G                         00000495 R     04
DIV_H$                          00000646 R     04
DIV_HC_HC                       000005CA R     04
DIV_H_H                         0000049B R     04
DIV_L$                          00000460 R     04
DIV_LU1$                        00000481 R     04
DIV_LU2$                        00000484 R     04
DIV_LU_LU                       00000463 R     04
DIV_L_L                         00000459 R     04
DIV_P_P                         0000155D R     04
DIV_W$                          00000443 R     04
DIV_WU$                         00000456 R     04
DIV_WU_WU                       00000446 R     04
DIV_W_W                         0000043C R     04
DOPE1                           00000008 R     03
DOPE2                           0000000C R     03
DSC$A_POINTER                 = 00000004
DSC$B_DIGITS                  = 00000009
DSC$B_DTYPE                   = 00000002
DSC$B_SCALE                   = 00000008
DSC$K_DTYPE_V                 = 00000001
DSC$W_LENGTH                  = 00000000
DST_ARG                       = 00000004
EQL_B$                          00000B81 R     04
EQL_B_B                         00000B77 R     04
EQL_D$                          00000BB5 R     04
EQL_DC$                         00000BF9 R     04
EQL_DC_DC                       00000BE8 R     04
EQL_D_D                         00000BAB R     04
EQL_F$                          00000BA8 R     04
EQL_FC$                         00000BE5 R     04
EQL_FC_FC                       00000BD4 R     04
EQL_FIXED_FIXED                 00001AA6 R     04
EQL_F_F                         00000B9E R     04
EQL_G$                          00000BC3 R     04
EQL_GC$                         00000C0F R     04
EQL_GC_GC                       00000BFC R     04
EQL_G_G                         00000BB8 R     04
```

```
EQL_H$                  00000BD1 R      04
EQL_HC$                 00000C25 R      04
EQL_HC_HC               00000C12 R      04
EQL_H_H                 00000BC6 R      04
EQL_L$                  00000B9B R      04
EQL_L_L                 00000B91 R      04
EQL_P$                  00001645 R      04
EQL_P_P                 00001610 R      04
EQL_RFA$                0000197C R      04
EQL_RFA_RFA             0000196B R      04
EQL_SET1$               0000112C R      04
EQL_SET2$               00001142 R      04
EQL_SET3$               00001146 R      04
EQL_SET4$               00001153 R      04
EQL_SET_RET$            0000115E R      04
EQL_SET_SET             00001118 R      04
EQL_T$                  00000A7F R      04
EQL_TF$                 000017C4 R      04
EQL_TF_TF               000017A1 R      04
EQL_T_T                 00000A63 R      04
EQL_VT_VT               00000A51 R      04
EQL_W$                  00000B8E R      04
EQL_W_W                 00000B84 R      04
ERRMSG                  00000000 R      02
ERRMSG1                 00000035 R      02
ERRMSG2                 00000067 R      02
ERRMSG3                 0000009E R      02
ERRMSG4                 000000D6 R      02
ERRMSG5                 0000010D R      02
EXT_PREC_DIV            00000474 R      04
GEQ_B$                  00000CE3 R      04
GEQ_B_B                 00000CD9 R      04
GEQ_D$                  00000D24 R      04
GEQ_D_D                 00000D1A R      04
GEQ_F$                  00000D17 R      04
GEQ_FIXED_FIXED         00001B28 R      04
GEQ_F_F                 00000D0D R      04
GEQ_G$                  00000D32 R      04
GEQ_G_G                 00000D27 R      04
GEQ_H$                  00000D40 R      04
GEQ_H_H                 00000D35 R      04
GEQ_L$                  00000CFD R      04
GEQ_LU$                 00000D0A R      04
GEQ_LU_LU               00000D00 R      04
GEQ_L_L                 00000CF3 R      04
GEQ_P$                  000016A5 R      04
GEQ_P_P                 00001694 R      04
GEQ_SET_SET             0000115F R      04
GEQ_T$                  00000AB0 R      04
GEQ_TF$                 000017F3 R      04
GEQ_TF_TF               000017C7 R      04
GEQ_T_T                 00000A94 R      04
GEQ_VT_VT               00000A82 R      04
GEQ_W$                  00000CF0 R      04
GEQ_W_W                 00000CE6 R      04
GET_RESULT_ADDR$        0000004B R      04
GIVE_INFO               00001DDE R      04
GTR_B$                  00000D4D R      04
GTR_B_B                 00000D43 R      04
GTR_D$                  00000D8E R      04
GTR_D_D                 00000D84 R      04
GTR_F$                  00000D81 R      04
GTR_FIXED_FIXED         00001AF4 R      04
GTR_F_F                 00000D77 R      04
GTR_G$                  00000D9C R      04
GTR_G_G                 00000D91 R      04
GTR_H$                  00000DAA R      04
GTR_H_H                 00000D9F R      04
GTR_L$                  00000D67 R      04
GTR_LU$                 00000D74 R      04
GTR_LU_LU               00000D6A R      04
GTR_L_L                 00000D5D R      04
GTR_P$                  00001691 R      04
GTR_P_P                 00001680 R      04
GTR_T$                  00000AE1 R      04
GTR_TF$                 00001824 R      04
GTR_TF_TF               000017F6 R      04
GTR_T_T                 00000AC5 R      04
GTR_VT_VT               00000AB3 R      04
GTR_W$                  00000D5A R      04
GTR_W_W                 00000D50 R      04
HEXDECIMAL$             00001FB5 R      04
HEXDECIMAL_OV$          0000207A R      04
INDIRECT_LO             0000108A R      04
INDIRECT_TPTR           0000109B R      04
INTERSECT_SET1$         000011A9 R      04
INTERSECT_SET2$         000011BE R      04
INTERSECT_SET3$         000011C2 R      04
INTERSECT_SET4$         000011CB R      04
INTERSECT_SET_RET$      000011DD R      04
INTERSECT_SET_SET       00001189 R      04
INT_OVERFLOW            000002B0 R      04
IN_SET$                 00001188 R      04
IN_SET_SET              0000117E R      04
LEFT_ARG              =  0000000C
LEQ_B$                  00000DB7 R      04
LEQ_B_B                 00000DAD R      04
LEQ_D$                  00000DF8 R      04
LEQ_D_D                 00000DEE R      04
LEQ_F$                  00000DEB R      04
LEQ_FIXED_FIXED         00001B0E R      04
LEQ_F_F                 00000DE1 R      04
LEQ_G$                  00000E06 R      04
LEQ_G_G                 00000DFB R      04
LEQ_H$                  00000E14 R      04
LEQ_H_H                 00000E09 R      04
LEQ_L$                  00000DD1 R      04
LEQ_LU$                 00000DDE R      04
LEQ_LU_LU               00000DD4 R      04
LEQ_L_L                 00000DC7 R      04
LEQ_P$                  000016CD R      04
LEQ_P_P                 000016BC R      04
LEQ_SET1$               000011F2 R      04
LEQ_SET2$               0000120B R      04
```

```
LEQ_SET3$                       0000120F  R     04
LEQ_SET4$                       00001220  R     04
LEQ_SET_RET$                    0000122B  R     04
LEQ_SET_SET                     000011DE  R     04
LEQ_T$                          00000B12  R     04
LEQ_TF$                         00001853  R     04
LEQ_TF_TF                       00001827  R     04
LEQ_T_T                         00000AF6  R     04
LEQ_VT_VT                       00000AE4  R     04
LEQ_W$                          00000DC4  R     04
LEQ_W_W                         00000DBA  R     04
LIB$SIGNAL                      ********    X   04
LSS_B$                          00000E21  R     04
LSS_B_B                         00000E17  R     04
LSS_D$                          00000E62  R     04
LSS_D_D                         00000E58  R     04
LSS_F$                          00000E55  R     04
LSS_FIXED_FIXED                 00001ADA  R     04
LSS_F_F                         00000E4B  R     04
LSS_G$                          00000E70  R     04
LSS_G_G                         00000E65  R     04
LSS_H$                          00000E7E  R     04
LSS_H_H                         00000E73  R     04
LSS_L$                          00000E3B  R     04
LSS_LU$                         00000E48  R     04
LSS_LU_LU                       00000E3E  R     04
LSS_L_L                         00000E31  R     04
LSS_P$                          000016B9  R     04
LSS_P_P                         000016A8  R     04
LSS_T$                          00000B43  R     04
LSS_TF$                         00001884  R     04
LSS_TF_TF                       00001856  R     04
LSS_T_T                         00000B27  R     04
LSS_VT_VT                       00000B15  R     04
LSS_W$                          00000E2E  R     04
LSS_W_W                         00000E24  R     04
MOD_LO$                         000007DA  R     04
MOD_LU_LU                       000007CB  R     04
MOD_LU_RET$                     000007E2  R     04
MOD_L_L                         000007B1  R     04
MOD_L_RET$                      000007CA  R     04
MTH$_FLOOVEMAT                =  001682C4
MTH$_FLOUNDMAT                =  001682CC
MTH$_UNDEXP                   =  00168294
MUL_B$                          00000650  R     04
MUL_BU$                         00000669  R     04
MUL_BU_BU                       00000653  R     04
MUL_B_B                         00000649  R     04
MUL_DC_DC                       000006EB  R     04
MUL_D_D                         000006BC  R     04
MUL_FC_FC                       000006CD  R     04
MUL_FIXED_FIXED                 00001A72  R     04
MUL_F_F                         000006B7  R     04
MUL_GC_GC                       00000729  R     04
MUL_G_G                         000006C1  R     04
MUL_HC_HC                       0000076D  R     04
MUL_H_H                         000006C7  R     04
MUL_L$                          00000696  R     04
MUL_LU$                         000006B4  R     04
MUL_LU_LU                       00000699  R     04
MUL_L_L                         0000068F  R     04
MUL_P$                          00001543  R     04
MUL_P_P                         000014B4  R     04
MUL_W$                          00000673  R     04
MUL_WU$                         0000068C  R     04
MUL_WU_WU                       00000676  R     04
MUL_W_W                         0000066C  R     04
NEQ_B$                          00000C32  R     04
NEQ_B_B                         00000C28  R     04
NEQ_D$                          00000C66  R     04
NEQ_DC$                         00000CAA  R     04
NEQ_DC_DC                       00000C99  R     04
NEQ_D_D                         00000C5C  R     04
NEQ_F$                          00000C59  R     04
NEQ_FC$                         00000C96  R     04
NEQ_FC_FC                       00000C85  R     04
NEQ_FIXED_FIXED                 00001AC0  R     04
NEQ_F_F                         00000C4F  R     04
NEQ_G$                          00000C74  R     04
NEQ_GC$                         00000CC0  R     04
NEQ_GC_GC                       00000CAD  R     04
NEQ_G_G                         00000C69  R     04
NEQ_H$                          00000C82  R     04
NEQ_HC$                         00000CD6  R     04
NEQ_HC_HC                       00000CC3  R     04
NEQ_H_H                         00000C77  R     04
NEQ_L$                          00000C4C  R     04
NEQ_L_L                         00000C42  R     04
NEQ_P$                          0000167D  R     04
NEQ_P_P                         00001648  R     04
NEQ_RFA$                        00001990  R     04
NEQ_RFA_RFA                     0000197F  R     04
NEQ_SET$                        0000127E  R     04
NEQ_SET1$                       0000124F  R     04
NEQ_SET1_1$                     00001256  R     04
NEQ_SET2$                       00001264  R     04
NEQ_SET3$                       00001268  R     04
NEQ_SET3_1$                     0000126F  R     04
NEQ_SET4$                       00001274  R     04
NEQ_SET4_1$                     0000127B  R     04
NEQ_SET_RET$                    00001286  R     04
NEQ_SET_SET                     0000122C  R     04
NEQ_T$                          00000B74  R     04
NEQ_TF$                         000018AA  R     04
NEQ_TF_TF                       00001887  R     04
NEQ_T_T                         00000B58  R     04
NEQ_VT_VT                       00000B46  R     04
NEQ_W$                          00000C3F  R     04
NEQ_W_W                         00000C35  R     04
NOT_B                           00000F26  R     04
NOT_D                           00000F3B  R     04
NOT_GTR$                        00001822  R     04
NOT_L                           00000F34  R     04
NOT_LSS$                        00001882  R     04
```

```
NOT_W                           00000F2D  R      04
OCTAL$                          00001FBB  R      04
OCTAL_OV$                       000020F4  R      04
OCTA_OVERFLOW$                  0000212C  R      04
OPERAND                         00000048  R      03
OPERATOR_ENTRY                =  00000004
ORT$K_MAX_ROUT                =  0000011C
ORT$K_MIN_ROUT                =  00000001
OR_B_B                          00000F46  R      04
OR_D_D                          00000F67  R      04
OR_L_L                          00000F5C  R      04
OR_W_W                          00000F51  R      04
OTS$POWCC                       ********    X    04
OTS$POWCDCD_R3                  ********    X    04
OTS$POWCDJ_R3                   ********    X    04
OTS$POWCGCG_R3                  ********    X    04
OTS$POWCGJ_R3                   ********    X    04
OTS$POWCJ                       ********    X    04
OTS$POWDD                       ********    X    04
OTS$POWDJ                       ********    X    04
OTS$POWDR                       ********    X    04
OTS$POWGG                       ********    X    04
OTS$POWGJ                       ********    X    04
OTS$POWHH_R3                    ********    X    04
OTS$POWHJ_R3                    ********    X    04
OTS$POWII                       ********    X    04
OTS$POWJJ                       ********    X    04
OTS$POWRD                       ********    X    04
OTS$POWRJ                       ********    X    04
OTS$POWRR                       ********    X    04
PLI$ANDBIT                      ********    X    00
PLI$CATBIT                      ********    X    00
PLI$CMPBIT                      ********    X    00
PLI$DIV_PK_LONG                 ********    X    00
PLI$NOTBIT                      ********    X    00
PLI$ORBIT                       ********    X    00
POSITIVE_MULD                   000006AC  R      04
POSITIVE_MULR                   000006A5  R      04
POST_DECR_D                     00001407  R      04
POST_DECR_L                     000013F6  R      04
POST_DECR_LU                    000013F6  R      04
POST_DECR_TPTR                  000013F6  R      04
POST_INCR_D                     000013A0  R      04
POST_INCR_L                     0000138F  R      04
POST_INCR_LU                    0000138F  R      04
POST_INCR_TPTR                  0000138F  R      04
POWER_DC_DC                     0000099E  R      04
POWER_DC_L                      000008C6  R      04
POWER_D_D                       0000092F  R      04
POWER_D_F                       0000090D  R      04
POWER_D_L                       00000876  R      04
POWER_FC_FC                     00000968  R      04
POWER_FC_L                      000008AD  R      04
POWER_F_D                       0000091E  R      04
POWER_F_F                       000008FC  R      04
POWER_F_L                       00000865  R      04
POWER_GC_GC                     000009D4  R      04
POWER_GC_L                      000008DF  R      04
POWER_G_G                       00000940  R      04
POWER_G_L                       00000887  R      04
POWER_H_H                       00000954  R      04
POWER_H_L                       0000089A  R      04
POWER_L_L                       00000856  R      04
POWER_W_W                       00000845  R      04
PRED_ENUM                       000019EE  R      04
PRE_DECR_D                      000013C4  R      04
PRE_DECR_L                      000013B1  R      04
PRE_DECR_LU                     000013B1  R      04
PRE_DECR_TPTR                   000013D8  R      04
PRE_INCR_D                      0000135D  R      04
PRE_INCR_L                      0000134A  R      04
PRE_INCR_LU                     0000134A  R      04
PRE_INCR_TPTR                   00001371  R      04
QUAD_OVERFLOW$                  00001EFB  R      04
REM_LU$                         000007FD  R      04
REM_LU_LU                       000007EE  R      04
REM_LU_RET$                     00000805  R      04
REM_L_L                         000007E3  R      04
REPORT_ERROR$                   00001E61  R      04
REPORT_ERROR1$                  00001EE5  R      04
REPORT_ERROR2$                  00002116  R      04
REPORT_ERRORU1$                 00001F5F  R      04
RESULT_ADDR                   =  00000014
RESULT_DESC                     00000040  R      03
RESULT_VALDESC                  00000044  R      03
RFA_OVERFLOW$                   00001E77  R      04
RIGHT_ARG                     =  00000010
ROUT_INDEX                    =  00000008
SAVE_OPERATOR_ENTRY             00000000  R      03
SAVE_RESULT                     00000004  R      03
SCALE_ARG1$                     00001713  R      04
SCALE_ARG2$                     00001705  R      04
SCRATCH1                        00000010  R      03
SCRATCH2                        00000020  R      03
SETUP_PACKED_NUMBERS$           000016D0  R      04
SETUP_PACKED_RSB$               00001774  R      04
SETUP_PLI_ABIT1$                00001920  R      04
SETUP_PLI_ABIT2$                00001954  R      04
SETUP_PLI_CONT1$                0000192B  R      04
SETUP_PLI_INTRFACE$             000018FF  R      04
SETUP_PLI_RESULT$               0000195F  R      04
SETUP_RSB$                      0000196A  R      04
SET_AND_RESULT                  00000F23  R      04
SET_NOT_RESULT                  00000F42  R      04
SET_OR_RESULT                   00000F72  R      04
SET_QUOTIENT                    00000487  R      04
SET_RESULT$                     00001E43  R      04
SET_RESULT1$                    00001ED4  R      04
SET_RESULT2$                    000020F7  R      04
SET_RESULTU1$                   00001F4E  R      04
SET_SHIFT_RESULT                0000083E  R      04
SET_XOR_RESULT                  00000F88  R      04
SHIFT_BINARY$                   00001E35  R      04
SHIFT_BINARY1$                  00001EC2  R      04
```

```
SHIFT_BINARY2$                  0000207D R      04
SHIFT_BINARYU1$                 00001F40 R      04
SHIFT_COUNT_NEGATIVE            000002DB R      04
SHIFT_DECIMAL$                  00001E1D R      04
SHIFT_DECIMAL1$                 00001EA2 R      04
SHIFT_DECIMAL2$                 00001FBE R      04
SHIFT_DECIMALU1$                00001F26 R      04
SHIFT_HEXDECIMAL$               00001E2E R      04
SHIFT_HEXDECIMAL1$              00001EB9 R      04
SHIFT_HEXDECIMAL2$              00002040 R      04
SHIFT_HEXDECIMALU1$             00001F39 R      04
SHIFT_LEFT_L$                   0000080B R      04
SHIFT_LEFT_L_L                  00000806 R      04
SHIFT_OCTA[$                    00001E3C R      04
SHIFT_OCTAL1$                   00001ECB R      04
SHIFT_OCTAL2$                   000020BA R      04
SHIFT_OCTALU1$                  00001F47 R      04
SHIFT_RT_L$                     0000081A R      04
SHIFT_RT_LU$                    00000842 R      04
SHIFT_RT_LU_LU                  0000081D R      04
SHIFT_RT_L_[                    0000080E R      04
SIZEOF_L                        000012EB R      04
SRC_ARG                       = 00000008
SS$_DECOVF                    = 000004A4
SS$_FLTDIV                    = 00000494
SS$_FLTDIV_F                  = 000004BC
SS$_FLTOVF                    = 00000048C
SS$_FLTOVF_F                  = 000004B4
SS$_FLTUND                    = 00000049C
SS$_FLTUND_F                  = 000004C4
SS$_INTDIV                    = 00000484
SS$_INTOVF                    = 0000047C
SS$_NORMAL                    = 00000001
SS$_OPCDEC                    = 0000043C
SS$_RESIGNAL                  = 00000918
SS$_ROPRAND                   = 00000454
STR$COMPARE                     ********    X   04
STR$CONCAT                      ********    X   04
STRING_TRUNCATE                 000002E9 R      04
SUB_B$                          000003A0 R      04
SUB_BU$                         000003AA R      04
SUB_BU_BU                       000003A3 R      04
SUB_B_B                         00000399 R      04
SUB_DC_DC                       000003F7 R      04
SUB_D_D                         000003DA R      04
SUB_FC_FC                       000003EB R      04
SUB_FIXED_FIXED                 00001A58 R      04
SUB_F_F                         000003D5 R      04
SUB_GC_GC                       00000403 R      04
SUB_G_G                         000003DF R      04
SUB_HC_HC                       00000411 R      04
SUB_H_H                         000003E5 R      04
SUB_L$                          000003C8 R      04
SUB_LU$                         000003D2 R      04
SUB_LU_LU                       000003CB R      04
SUB_L_L                         000003C1 R      04
SUB_P_P                         00001418 R      04
SUB_TPTR_L                      00001316 R      04
SUB_TPTR_TPTR                   00001330 R      04
SUB_W$                          000003B4 R      04
SUB_WU$                         000003BE R      04
SUB_WU_WU                       000003B7 R      04
SUB_W_W                         000003AD R      04
SUCC_ENUM                       000019DA R      04
SYS$UNWIND                      ********  GX    04
TEST_IMAGINARYD                 00000CA0 R      04
TEST_IMAGINARYF                 00000C8C R      04
TEST_IMAGINARYG                 00000CB5 R      04
TEST_IMAGINARYH                 00000CCB R      04
TOKEN$K_BIN_INTEGER           = 0000000A
TOKEN$K_HEX_INTEGER           = 00000005
TOKEN$K_INTEGER               = 00000004
TOKEN$K_OCT_INTEGER           = 0000000B
TOKENCODE                     = 0000000C
TOKEN_NAME                    = 0000000C
TRAP_MSG_HANDLER                00001BBE RG     04
TRUNCATE_THE_OTHERS$            00001742 R      04
UMB$                            00000F92 R      04
UML$                            00000FA4 R      04
UMW$                            00000F9B R      04
UNARY_MINUS_B                   00000F8C R      04
UNARY_MINUS_D                   00000FB4 R      04
UNARY_MINUS_DC                  00000FCB R      04
UNARY_MINUS_F                   00000FB0 R      04
UNARY_MINUS_FC                  00000FC2 R      04
UNARY_MINUS_FIXED               00001A16 R      04
UNARY_MINUS_G                   00000FB8 R      04
UNARY_MINUS_GC                  00000FD4 R      04
UNARY_MINUS_H                   00000FBD R      04
UNARY_MINUS_HC                  00000FDF R      04
UNARY_MINUS_L                   00000F9E R      04
UNARY_MINUS_LU                  00000FA7 R      04
UNARY_MINUS_O                   000019B6 R      04
UNARY_MINUS_P                   000015C9 R      04
UNARY_MINUS_Q                   00001999 R      04
UNARY_MINUS_W                   00000F95 R      04
UNARY_P$                        000015F4 R      04
UNARY_PLUS_B                    00000FEA R      04
UNARY_PLUS_D                    00000FFA R      04
UNARY_PLUS_DC                   00001011 R      04
UNARY_PLUS_F                    00000FF6 R      04
UNARY_PLUS_FC                   00001008 R      04
UNARY_PLUS_FIXED                00001A02 R      04
UNARY_PLUS_G                    00000FFE R      04
UNARY_PLUS_GC                   0000101A R      04
UNARY_PLUS_H                    00001003 R      04
UNARY_PLUS_HC                   00001025 R      04
UNARY_PLUS_L                    00000FF2 R      04
UNARY_PLUS_O                    000019AB R      04
UNARY_PLUS_P                    000015E6 R      04
UNARY_PLUS_Q                    00001993 R      04
UNARY_PLUS_W                    00000FEE R      04
UNION_SET1$                     000012A7 R      04
UNION_SET2$                     000012B9 R      04
```

```
UNION_SET3$                     000012BD R      04
UNION_SET4$                     000012C5 R      04
UNION_SET_RET$                  000012D6 R      04
UNION_SET_SET                   00001287 R      04
UNKNOWN_ROUT_INDEX              00001B42 R      04
UNKNOWN_SIGNAL                  00001DD6 R      04
UQUAD_OVERFLOW$                 00001F75 R      04
WORK_OCTA                       0000005C R      03
XOR_L_L                         00000F76 R      04
XOR_L_L_1$                      00000F81 R      04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000065 (101.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DBG$PLIT	00000145 (325.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE
DBG$OWN	0000006C (108.)	03 (3.)	PIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
DBG$CODE	0000212F (8495.)	04 (4.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	10	00:00:00.12	00:00:00.39
Command processing	83	00:00:00.81	00:00:03.75
Pass 1	536	00:00:22.40	00:01:11.19
Symbol table sort	7	00:00:02.65	00:00:12.27
Pass 2	535	00:00:09.06	00:00:36.38
Symbol table output	1	00:00:00.52	00:00:01.27
Psect synopsis output	1	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1168	00:00:35.59	00:02:05.29

The working set limit was 1500 pages.
137763 bytes (270 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1722 non-local and 36 local symbols.
4104 source lines were read in Pass 1, producing 64 object records in Pass 2.
17 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DEBUG.OBJ]DBGMSG.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	13

1264 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DBGPERMOP/OBJ=OBJ$:DBGPERMOP MSRC$:DBGPERMOP/UPDATE=(ENH$:DBGPERMOP)+LIB$:DBGMSG/LIB

0091 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGPERMOP
LIS

DBGSCREEN
LIS

DBGPRINT
LIS