```
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**PLIPKDIVS

```
PPPPPPPP    LL          IIIIII    PPPPPPPP    KK      KK  DDDDDDDD     IIIIII    VV      VV    SSSSSSSS
PPPPPPPP    LL          IIIIII    PPPPPPPP    KK      KK  DDDDDDDD     IIIIII    VV      VV    SSSSSSSS
PP      PP  LL            II      PP      PP  KK      KK  DD      DD     II      VV      VV  SS
PP      PP  LL            II      PP      PP  KK      KK  DD      DD     II      VV      VV  SS
PP      PP  LL            II      PP      PP  KK    KK    DD      DD     II      VV      VV  SS
PP      PP  LL            II      PP      PP  KK    KK    DD      DD     II      VV      VV  SS
PPPPPPPP    LL            II      PPPPPPPP    KKKKKK      DD      DD     II      VV      VV    SSSSSS
PPPPPPPP    LL            II      PPPPPPPP    KKKKKK      DD      DD     II      VV      VV    SSSSSS
PP          LL            II      PP          KK    KK    DD      DD     II      VV      VV          SS
PP          LL            II      PP          KK    KK    DD      DD     II      VV      VV          SS
PP          LL            II      PP          KK      KK  DD      DD     II        VV  VV            SS      ....
PP          LL            II      PP          KK      KK  DD      DD     II        VV  VV            SS      ....
PP          LLLLLLLLLL    IIIIII  PP          KK      KK  DDDDDDDD     IIIIII      VV      SSSSSSSS      ....
PP          LLLLLLLLLL    IIIIII  PP          KK      KK  DDDDDDDD     IIIIII      VV      SSSSSSSS      ....

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

```
0000     1          .title  pli$div_pkshort
0000     2          .ident  /1-002/                                        ; Edit WHM1002
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 ;++
0000    28 ;
0000    29 ;
0000    30 ; routine:
0000    31 ;
0000    32 ;       PLI$DIV_PKSHORT
0000    33 ;
0000    34 ;
0000    35 ; facility:
0000    36 ;
0000    37 ;       VAX/VMS PL1 runtime library.
0000    38 ;
0000    39 ; abstract:
0000    40 ;
0000    41 ;       Runtime routine performs fixed decimal (packed decimal) division.
0000    42 ;       The routine is called  when precision and scale  requirements for
0000    43 ;       the quotient imply multiple  precision  division.  The routine is
0000    44 ;       only called when such multiple precision division is required and
0000    45 ;       when the divisor has a precision of less than  30 decimal digits.
0000    46 ;       (Call  pli$div_pk_long  if  multiple precision division is
0000    47 ;       required and the divisor has precision  30 or 31 decimal digits).
0000    48 ;
0000    49 ; author: Peter Baum  20-jun-1980
0000    50 ;
0000    51 ; modifications:
0000    52 ;
0000    53 ;
0000    54 ;
0000    55 ;       1-002   Bill Matthews   29-September-1982
0000    56 ;
0000    57 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
```

```
0000     58 ;
0000     59 ;
0000     60 ;
0000     61 ;
0000     62 ; documentation file: [pl1.doc.codegen]THEORY.MEM
0000     63 ;
0000     64 ; functional description:
0000     65 ;
0000     66 ;       This routine calculates:
0000     67 ;
0000     68 ;       z = x / y
0000     69 ;
0000     70 ;       let a = scale(z) + scale(y) - scale(x)  - 31 + prec(x)
0000     71 ;           b = scale(z) + scale(y) - scale(x) + prec(x)
0000     72 ;           c = 31 - prec(x)
0000     73 ;           d = 31 - prec(y)
0000     74 ;
0000     75 ;       this routine is called if b > 31 and d > 1
0000     76 ;
0000     77 ;       Prior to the call:
0000     78 ;               if c not 0 then shift x left by c.
0000     79 ;               Thus x is a 31 digit packed decimal.
0000     80 ;
0000     81 ;
0000     82 ;
0000     83 ;
0000     84 ;       input:
0000     85 ;               0(ap)   # of arguments
0000     86 ;               4(ap)   address of dividend (shifted left by c)
0000     87 ;               8(ap)   address of divisor
0000     88 ;               12(ap)  precision of divisor (high order bytes zeroed)
0000     89 ;               16(ap)  address of quotient
0000     90 ;               20(ap)  precision of quotient (high order bytes zeroed)
0000     91 ;               24(ap)  a as defined above(high order bytes zeroed)
0000     92 ;               28(ap)  d as defined above(high order bytes zeroed)
0000     93 ;
0000     94 ;
0000     95 ;       output:
0000     96 ;               quotient returned at address specified by 16(ap)
0000     97 ;
0000     98 ;
0000     99 ; variable usage:
0000    100 ;
0000    101 ;                           size
0000    102 ;                             in
0000    103 ;           variable   digits          use
0000    104 ;           ---------  ------   -------------------------
0000    105 ;               x(ap)     31     Dividend
0000    106 ;               y(ap)    py(ap)  Divisor
0000    107 ;               py(ap)   ------  Binary number that gives precision of y
0000    108 ;               z(ap)    pz(ap)  Quotient
0000    109 ;               pz(ap)   ------  Binary number that gives precision of z
0000    110 ;               (sp)      31     Initially abs(x); successive remainders as
0000    111 ;                                algorithm progresses.
0000    112 ;               stkz2(sp) d      Temporarily holds the next d
0000    113 ;                                digits of quotient.
0000    114 ;               stkt1(sp) 31     Temporary because packed instructions
```

```
0000    115 ;                                   don't allow overlapped operands
0000    116 ;               stky(sp)  31        Holds abs(y)
0000    117 ;               stksign(sp)         2 bits used to indicate the sign of the
0000    118 ;                                   quotient. 00=+, 10=+, 01=-; via incb
0000    119 ;
0000    120 ;
0000    121 ;
0000    122 ; register usage:
0000    123 ;
0000    124 ;    register   use
0000    125 ;    --------   ------------------------------------------------------------
0000    126 ;       r6      a = additional digits of precision required beyond prec(x)
0000    127 ;       r7      stky(sp) = address of divisor
0000    128 ;       r8      py(ap) = precision of divisor
0000    129 ;       r9      r = number of additional digits of the quotient
0000    130 ;               that are to be found for next step
0000    131 ;       r10     z(ap)
0000    132 ;       r11     d = 31 - prec(y) = max. no of digits obtained each iteration
0000    133 ;
0000    134 ;
0000    135 ;
0000    136 ;
0000    137 ; optimization notes:
0000    138 ;
0000    139 ;       1) Optimized for speed, not space.
0000    140 ;       2) Optimized for y > 0.
0000    141 ;       3) Assumes speed for register to register operations are the same
0000    142 ;          for byte operations and longword operations.
0000    143 ;       4) Many packed instruction sequences were timed. Do not change
0000    144 ;          unless actual tests are made to determine relative speed.
0000    145 ;          Tests were made on 11/780 and Comet.
0000    146 ;
0000    147 ;--
0000    148 ;
0000    149 ; stack offsets for work area
0000    150 ;
0000    151         $offset 0,,<-                   ;
0000    152         <,16>,-                         ;abs(x), 31 digits
0000    153         <stkz2,16>,-                    ;z2 31 digits
0000    154         <stkt1,16>,-                    ;t1 31 digits
0000    155         <stky,16>,-                     ;abs(y)
0000    156         <stksign,1>,-                   ;sign of quotient, 2 bits
0000    157         <stklen,0>,-                    ;length of work area
0000    158         >                               ;
0000    159 ;
0000    160 ; parameter offsets
0000    161 ;
0000    162         $offset 4,,<-                   ;
0000    163         <x>,-                           ;x = dividend by reference
0000    164         <y>,-                           ;y = divisor by reference
0000    165         <py>,-                          ;prec(y) by value
0000    166         <z>,-                           ;z = quotient by reference
0000    167         <pz>,-                          ;prec(z) by value
0000    168         <consta>,-                      ;a by value
0000    169         <constd>,-                      ;d by value
0000    170         >                               ;
0000    171 ;
```

```
                                   0000   172 ;
                                   0000   173 ;
                                   0000   174         rtshare
                                   0000   175 ;
                                   0000   176 ;       constant data area
                                   0000   177 ;
                               0C  0000   178 zero:   .packed +0                      ;local packed decimal constant zero
                                   0001   179 ;
                                   0001   180 ; local symbol definitions
                                   0001   181 ;
                         0000000F  0001   182 bytes_to_sign=15                        ;bytes to sign for fixed decimal 31
                                   0001   183 ;
                                   0001   184 ;
                                   0001   185 ;
                                   0001   186
                             CFFC  0001   187         .entry  pli$div_pkshort,^M<iv,dv,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                   0003   188 ;
                                   0003   189 ;initialize registers and temporaries
                                   0003   190 ;
                 5E    BF AE   9E  0003   191         movab   -stklen(sp),sp          ;make room for temporaries
                 5A    10 AC   D0  0007   192         movl    z(ap),r10               ;sav• address of quotient
                 57    30 AE   9E  000B   193         movab   stky(sp),r7             ;address of divisor
                 58    0C AC   D0  000F   194         movl    py(ap),r8               ;precision of divisor
                 5B    1C AC   D0  0013   195         movl    constd(ap),r11          ;d = 31 - prec(y)
                 56    18 AC   D0  0017   196         movl    consta(ap),r6           ;a = scale(z) + scale(y) - scale(x)
                                   001B   197                                         ; - 31 + prec(x)
                       40 AE   94  001B   198         clrb    stksign(sp)             ;clear sign flag
             6E    04 BC   1F  34  001E   199         movp    #31,@x(ap),(sp)         ;move x, set cond. code
                           26  14  0023   200         bgtr    50$                     ;branch if x>0
                           1E  19  0025   201         blss    40$                     ;branch if x<0
                                   0027   202 ;
                                   0027   203 ;x = 0
                                   0027   204 ;
      08 BC    58    D5 AF   00  37  0027   205         cmpp4   #0,zero,r8,@y(ap)       ;set condition code
                           0A  13  002E   206         beql    30$                     ;branch if divide by 0
 14 AC    00    CB AF    00   00  F8  0030   207         ashp    #0,#0,zero,#0,pz(ap),(r10) ;z = 0
                           6A      0038
                               04  0039   208         ret
      BF AF    00    C2 AF   00  27  003A   209 30$:    divp    #0,zero,#0,zero,pz(ap),(r10) ;cause divide by 0
                 6A    14 AC       0041
                               04  0044   210         ret                             ;
                                   0045   211 ;
                                   0045   212 ;x not 0, determine sign of x
                                   0045   213 ;
                                   0045   214 40$:
                       40 AE   96  0045   215         incb    stksign(sp)             ;set low order bit
                       0F AE   97  0048   216         decb    bytes_to_sign(sp)                       ;x < 0 so make it positive
                                   004B   217 ;
                                   004B   218 ;determine sign of y
                                   004B   219 ;y may be 0 at this point
                                   004B   220 ;code optimized for y>0
                                   004B   221 ;
                                   004B   222 50$:
             67    08 BC   58  34  004B   223         movp    r8,@y(ap),(r7)          ;move y into temporary
                           0C  18  0050   224         bgeq    60$                     ;branch if y > = 0
                       40 AE   96  0052   225         incb    stksign(sp)             ;set neg indicator
 58    A4 AF    00    08 BC   58  23  0055   226         subp6   r8,@y(ap),#0,zero,r8,(r7) ;convert to positive
```

```
                                67          005D
                                            005E    227 60$:
                                            005E    228 ;
                                            005E    229 ;start of divide proper; setup
                                            005E    230 ;
                    67  58  6E   1F  37      005E    231         cmpp4   #31,(sp),r8,(r7)        ;x<y?
                                13  19      0063    232         blss    95$                     ;branch if x<y, i.e. shift of d is o.k.
                                            0065    233 ;
                                            0065    234 ;y < x
                                            0065    235 ;
6A  14 AC  6E   1F  67  58  27              0065    236         divp    r8,(r7),#31,(sp),pz(ap),(r10) ;z=x/y
    1F  6A  14 AC   67  58  25              006D    237         mulp    r8,(r7),pz(ap),(r10),#31,stkt1(sp) ;t1=(x/y)*y
                        20 AE               0074
                                14  11      0076    238         brb     110$                    ;
                                            0078    239 ;
                                            0078    240 ;x < y
                                            0078    241 ;
  14 AC  00  83 AF  00  00  F8              0078    242 95$:    ashp    #0,#0,zero,#0,pz(ap),(r10) ;clear quotient
                            6A              008?
                                11  11      0081    243         brb     115$                    ;
                                            0083    244 ;
                                            0083    245 ;start of multiple precision divide
                                            0083    246 ;
    1F  10 AE   5B  67  58  25              0083    247 100$:   mulp    r8,(r7),R11,stkz2(sp),#31,stkt1(sp) ;t1=y*z2
                        20 AE               008A
        6E  1F  20 AE   1F  22              008C    248 110$:   subp4   #31,stkt1(sp),#31,(sp)  ;x=x-t1
                                41  13      0092    249         beql    150$                    ;branch if remainder = 0
                                            0094    250 ;
                                            0094    251 ;determine r, the number of the next low order digits to obtain
                                            0094    252 ;
                        59  5B  D0          0094    253 115$:   movl    r11,r9                  ;r=d
                        5B  56  D1          0097    254         cmpl    r6,r11                  ;a>d?
                                03  14      009A    255         bgtr    130$                    ;branch if larger
                        59  56  D0          009C    256         movl    r6,r9                   ;r=a
20 AE   1F  00  6E   1F  59  F8             009F    257 130$:   ashp    r9,#31,(sp),#0,#31,stkt1(sp) ;shift x left by r
            6E  20 AE   1F  34              00A7    258         movp    #31,stkt1(sp),(sp)      ;copy back into x
  14 AC  00  6A  14 AC  59  F8              00AC    259         ashp    r9,pz(ap),(r10),#0,pz(ap),stkt1(sp) ;shift z left by r
                    20 AE                   00B4
          6A  20 AE  14 AC  34              00B6    260         movp    pz(ap),stkt1(sp),(r10)  ;copy back into z
10 AE   5B  6E  1F  67  58  27              00BC    261         divp    r8,(r7),#31,(sp),r11,stkz2(sp) ;z2(d)=x/y
      6A  14 AC  10 AE  5B  20              00C4    262         addp4   r11,stkz2(sp),pz(ap),(r10) ;z=z+z2
                        56  59  C2          00CB    263         subl2   r9,r6                   ;a=a-r
                            B3  12          00CE    264         bneq    100$                    ;branch if more
                    16 40 AE    E8          00D0    265         blbs    stksign(sp),155$        ;branch if quotient <0
                                04          00D4    266         ret                             ;
                                            00D5    267 ;
                                            00D5    268 ;remainder = 0
                                            00D5    269 ;
                                            00D5    270 150$:                                   ;remainder = 0
  14 AC  00  6A  14 AC  56  F8              00D5    271         ashp    r6,pz(ap),(r10),#0,pz(ap),stkt1(sp) ;account for scale
                    20 AE                   00DD
                0D 40 AE    E8              00DF    272         blbs    stksign(sp),160$        ;branch if quotient < 0
          6A  20 AE  14 AC  34              00E3    273         movp    pz(ap),stkt1(sp),(r10)  ;copy back into quotient
                                04          00E9    274         ret                             ;
                                            00EA    275 ;
                                            00EA    276 ;quotient < 0
                                            00EA    277 ;
```

```
            20 AE    6A    14 AC    34  00EA   278 155$:   movp    pz(ap),(r10),stkt1(sp)  ;copy quotient into temp
                                       00F0   279 160$:                                   ;enter if t1 holds quotient
  FF07 CF   00    20 AE    14 AC    23  00F0   280         subp6   pz(ap),stkt1(sp),#0,zero,pz(ap),(r10) ;make z negative
                   6A    14 AC             00F9
                                       04  00FC   281         ret                             ;
                                           00FD   282
                                           00FD   283         .end
```

```
BYTES_TO_SIGN  = 0000000F
CONSTA           00000018
CONSTD           0000001C
DIR...         = 00000001
PLI$DIV_PKSHORT  00000001 RG    02
PY               0000000C
PZ               00000014
STKLEN           00000041
STKSIGN          00000040
STKT1            00000020
STKY             00000030
STKZ2            00000010
X                00000004
Y                00000008
Z                00000010
ZERO             00000000 R     02
```

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	00000041 (65.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI$CODE	000000FD (253.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.32
Command processing	77	00:00:00.51	00:00:01.69
Pass 1	67	00:00:01.18	00:00:02.27
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	51	00:00:00.63	00:00:01.16
Symbol table output	2	00:00:00.02	00:00:00.02
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	207	00:00:02.45	00:00:05.49

```
The working set limit was 750 pages.
6483 bytes (13 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 16 non-local and 12 local symbols.
283 source lines were read in Pass 1, producing 12 object records in Pass 2.
3 pages of virtual memory were used to define 3 macros.
```

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

Macro library name                      Macros defined
------------------                      --------------
_$255$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1              3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   0
TOTALS (all libraries)                              3

44 GETS were required to define 3 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIPKDIVS/OBJ=OBJ$:PLIPKDIVS MSRC$:PLIPKDIVS/UPDATE=(ENH$:PLIPKDIVS)+LIB$:PLIRTM
```

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIMSGTXT
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECOPT
LIS

PLIOPEN
LIS

PLIREAD
LIS

PLIPROTEC
LIS

PLIREWRIT
LIS

PLIGETLIS
LIS

PLIPUTEDI
LIS

PLIPKDIVL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIVS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS