

FILEID**PLIPKDIVL

L 8

```
0000 1 .title pli$div_pk_long
0000 2 .ident /1-002/ ; Edit WHM1002
0000 3 :*****
0000 4 :*****
0000 5 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 6 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 7 :* ALL RIGHTS RESERVED.
0000 8 :*
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :*
0000 27 :++
0000 28 :*
0000 29 :*
0000 30 :routine:
0000 31 :    PLI$DIV_PK_LONG
0000 32 :*
0000 33 :*
0000 34 :*
0000 35 :facility:
0000 36 :    VAX/VMS PL1 runtime library.
0000 37 :*
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 30 or 31 decimal digits.
0000 46 :    (Call pli$div_pkshort if multiple precision division is
0000 47 :    required and the divisor has precision less than 30 decimal digits).
0000 48 :*
0000 49 :author: Peter Baum 30-jun-1980
0000 50 :*
0000 51 :modifications:
0000 52 :*
0000 53 :*
0000 54 :    1-002 Bill Matthews 29-September-1982
0000 55 :*
0000 56 :    Invoke macros $defdat and rtshare instead of $defopr and share.
0000 57 :*
```

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 :
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 < 2
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 :
0000 93 :
0000 94 : output:
0000 95 : quotient returned at address specified by 16(ap)
0000 96 :
0000 97 :
0000 98 : optimization notes:
0000 99 :
0000 100 : 1) Optimized for speed, not space.
0000 101 : 2) Optimized for y > 0.
0000 102 : 3) Assumes speed for register to register operations are the same
0000 103 : for byte operations and longword operations.
0000 104 : 4) Many packed instruction sequences were timed. Do not change
0000 105 : unless actual tests are made to determine relative speed.
0000 106 : Tests were made on 11/780 and Comet.
0000 107 :
0000 108 :
0000 109 :
0000 110 : possible optimizations:
0000 111 :
0000 112 : 1) currently we always calculate the next 15 digits each
0000 113 : iteration and then truncate the last iteration as part of
0000 114 : the final step. We might be able to go through making

0000 115 : calculations with fewer digits on this last pass.
 0000 116 :
 0000 117 :
 0000 118 :
 0000 119 :
 0000 120 :
 0000 121 :
 0000 122 :
 0000 123 :
 0000 124 :
 0000 125 :
 0000 126 :
 0000 127 :
 0000 128 :
 0000 129 :
 0000 130 :
 0000 131 :
 0000 132 :
 0000 133 :
 0000 134 :
 0000 135 :
 0000 136 :
 0000 137 :
 0000 138 : variable use:
 0000 139 :
 0000 140 :
 0000 141 :
 0000 142 :
 0000 143 :
 0000 144 :
 0000 145 :
 0000 146 :
 0000 147 :
 0000 148 :
 0000 149 :
 0000 150 :
 0000 151 :
 0000 152 :
 0000 153 :
 0000 154 :
 0000 155 :
 0000 156 :
 0000 157 :
 0000 158 :
 0000 159 :
 0000 160 :
 0000 161 :
 0000 162 :
 0000 163 :
 0000 164 :
 0000 165 :
 0000 166 : register usage:
 0000 167 :
 0000 168 :
 0000 169 :
 0000 170 :
 0000 171 :
 size
 in
 variable digits use

 y1 15 High order digits of divisor.
 y2 16 Low order digits of divisor.
 x 31 Initially dividend, thereafter
 remainders of successive divide
 operations.
 z px(ap) Quotient.
 z2 31 Temporarily holds trial low
 order digits of quotient.
 t1 31 High order digits of the remainder.
 t2 31 Holds the 15 low order digits of the
 46 digit remainder. 31 digits for
 possible later changes.
 t3 16 Holds the low order digits of the
 remainder.
 t4 31 Temporary used because packed instructions can
 not overlap their operands.

```

0000 172 : r8      py(ap) = precision of y
0000 173 : r9      r = number of additional digits of the quotient
0000 174 :          that are to be found for next step
0000 175 : r10     z(ap)
0000 176 : r11     pz(ap) = precision of quotient
0000 177 :
0000 178 :
0000 179 :-- 
0000 180 :
0000 181 : stack offsets for work area
0000 182 :
0000 183 : $offset 0,,<-
0000 184 : <,16>,-           ;x, 31 digits
0000 185 : <stky1,8>,-       ;y1 15 digits
0000 186 : <stky2,9>,-       ;y2 16 digits
0000 187 : <stkz2,16>,-      ;z2 31 digits
0000 188 : <stkt1,16>,-      ;t1 31 digits
0000 189 : <stkt2,16>,-      ;t2 31 digits (15 digits used)
0000 190 : <stky,16>,-        ;y 31 digits
0000 191 : <stkt3,9>,-        ;t3 16 digits
0000 192 : <stkt4,16>,-      ;t4 31 digits
0000 193 : <stksign,1>,-     ;sign of quotient, 2 bits
0000 194 : <stklen,0>,-      ;length of work area
0000 195 : >
0000 196 :
0000 197 : parameter offsets
0000 198 :
0000 199 : $offset 4,,<-
0000 200 : <x>,-           ;x = dividend by reference
0000 201 : <y>,-           ;y = divisor by reference
0000 202 : <py>,-          ;prec(y) by value
0000 203 : <z>,-           ;z = quotient by reference
0000 204 : <pz>,-          ;prec(z) by value
0000 205 : <consta>,-      ;a as defined above
0000 206 : >
0000 207 :
0000 208 : constant data area
0000 209 : (plidata.mar not used because they cause longword displacements to be used)
0000 210 :
0000 211 : rtshare
0000 212 one: .packed +1
0000 213 zero: .packed +0
0000 214 :**warning** the following two data definitions must be contiguous
0000 215 nines: .byte 9           ;10**16 - 1 (must be followed by 10**15-1)
0000 216 nine15: .packed +9999999999999999 ;10**15 - 1
0000 217 bignine:.packed +00000000000000009999999999999999 ;10**16 - 1
0000 218 ten15: .packed +10000000000000000 ;10**15
0000 219 neg9: .packed -9999999999999999 ;-(10**16 - 1)
0000 220 :
0000 221 : local symbol definitions
0000 222 :
0000 223 bytes_to_sign=15          ;bytes to sign for fixed decimal 31
0000 224 :
0000 225 : run time routine pli$div_pk_long
0000 226 :
0000 227 : .entry pli$div_pk_long,^M<iv,dv,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>

```

```

      5E  85 AE  9E  002F  228    movab  -stklen(sp),sp      ;make room for temporaries
      56  18 AC  D0  0033  229    movl   consta(ap),r6      ;get value of a
      57  51 AE  9E  0037  230    movab  stky(sp),r7      ;address of copy of divisor
      58  0C AC  D0  003B  231    movl   py(ap),r8      ;precision of divisor
      5A  10 AC  D0  003F  232    movl   z(ap),r10     ;save address of quotient
      5B  14 AC  D0  0043  233    movl   pz(ap),r11     ;precision of quotient
      6E  04 BC  1F  34  004A  234    clrb   stksign(sp)  ;clear sign flag
      24  14  004F  235    movp   #31,@x(ap),(sp)  ;move x, set cond. code
      1C  19  0051  236    bgtr  50$      ;branch if x > 0
      0053  237    blss   40$      ;branch if x < 0

      0053  238    ;x = 0
      0053  239    ;x = 0

      08 BC  58  AA AF  00  37  0053  240    ;:
      09  13  005A  241    cmpp4 #0,zero,r8,@y(ap)  ;divisor zero?
      6A  5B  00  A0 AF  00  00  F8  005C  242    beql  30$      ;branch if divide by 0
      04  0064  243    ashp  #0,#0,zero,#0,r11,(r10) ;z = 0
      5B  95 AF  00  98 AF  00  27  0065  244    ret
      6A  006D  245  30$:    divp  #0,zero,#0,zero,r11,(r10) ;cause divide by zero
      04  006E  246    ret
      006F  247    ;:
      006F  248    ;x not 0
      006F  249    ;:
      006F  250  40$:    ;:
      7A AE  96  006F  251    incb   stksign(sp)  ;set low order bit
      OF AE  97  0072  252    decb   bytes_to_sign(sp) ;x < 0 so make it positive
      0075  253    ;:
      0075  254    ;determine sign of y
      0075  255    ;y may be 0 at this point
      0075  256    ;optimized for y>0
      0075  257    ;:
      0075  258  50$:    ;:
      67  08 BC  58  34  0075  259    movp   r8,@y(ap),(r7)  ;move y into temporary
      0D  18  007A  260    bgeq  60$      ;branch if so
      7A AE  96  007C  261    incb   stksign(sp)  ;set neg indicator
      58  FF7A CF  00  08 BC  58  23  007F  262    subp6 r8,@y(ap),#0,zero,r8,(r7) ;convert to positive
      67  0088  263    ;:
      0089  264    ;get y1 and y2
      0089  265    ;:
      0089  266  60$:    ;:
      OF  00  67  58  F0  8F  F8  0089  267    ashp  #-16,r8,(r7),#0,#15,stky1(sp) ;high order 15 digits of y
      1F  00  10 AE  OF  10  F8  0090  268    ashp  #16,#15,stky1(sp),#0,#31,stkt1(sp) ;y with 16 low order zeroed
      10  67  58  31 AE  1F  23  009B  269    subp6 #31,stkt1(sp),r8,(r7),#16,stky2(sp) ;16 low order digits y
      18 AE  00A2  270    ;:
      00A4  271    ;prec(y) is large enough for y1 to possibly not be 0
      00A4  272    ;:
      10 AE  OF  FF58 CF  00  37  00A4  273    cmpp4 #0,zero,#15,stky1(sp) ;y1=0?
      03  13  00AC  274    beql  80$      ;branch if y1 is zero
      0098  31  00AE  275    brw   200$      ;branch if y1 is not zero
      00B1  276    ;:
      00B1  277    ;y2(16) holds all of y
      00B1  278    ;:
      00B1  279  80$:    ;

```

18 AE 10 6E 1F 37 00B1 280 cmpp4 #31,(sp),#16,stky2(sp) ;x<y?
 6A 5B 6E 1F 18 AE 10 13 19 00B7 281 blss 95\$;branch if x<y; shift x by 15 is ok
 1F 6A 5B 18 AE 10 25 00B9 282 divp #16,stky2(sp),#31,(sp),r11,(r10);z=x/y2
 31 AE 00C1 283 mulp #16,stky2(sp),r11,(r10),#31,stkt1(sp);t1=(x/y2)*y2
 5B 00 FF2F CF 00 00 15 11 00CA 284 brb 110\$
 6A 00CC 285 95\$: ashp #0,#0,zero,#0,r11,(r10);clear quotient
 1F 21 AE 0F 18 AE 12 11 00D5 286 brb 115\$
 31 AE 00D7 287 100\$: mulp #16,stky2(sp),#15,stkz2(sp),#31,stkt1(sp);t1=y2*z2
 6E 1F 31 AE 1F 22 00E1 288 110\$: subp4 #31,stkt1(sp),#31,(sp);x=x-t1
 3E 13 00E7 289 beql 150\$;branch if remainder = 0
 COE9 290 :
 00E9 291 :determine r, the number of the next low order digits to obtain
 00E9 292 :
 59 0F D0 00E9 293 115\$: movl #15,r9 ;r=15
 59 56 D1 00EC 294 cmpl r6,r9 ;a>15?
 03 18 00EF 295 bgeq 130\$;branch if larger
 59 56 D0 00F1 296 movl r6,r9 ;r=a
 31 AE 1F 00 6E 1F 59 F8 00F4 297 130\$: ashp r9,#31,(sp),#0,#31,stkt1(sp);shift x left by r
 31 AE 5B 00 6A 31 AE 1F 34 00FC 298 movp #31,stkt1(sp),(sp);copy back into x
 OF 6E 1F 18 AE 10 27 010E 299 ashp r9,r11,(r10),#0,r11,stkt1(sp);shift z left by r
 21 AE 0115 300 movp r11,stkt1(sp),(r10);copy back into z
 6A 5B 21 AE 0F 20 0117 301 divp #16,stky2(sp),#31,(sp),#15,stkz2(sp);z2(15)=x/y2
 56 59 C2 011D 302 addp4 #15,stkz2(sp),r11,(r10);z=z+z2
 B5 12 0120 303 subl2 r9,r6 ;a=a-r
 13 7A AE E8 0122 304 bneq 100\$;branch if more
 04 0126 305 140\$: blbs stksign(sp),155\$;branch if quotient <0
 0127 306 ret :
 0127 307 :
 0127 308 :remainder = 0
 0127 309 :
 0127 310 150\$:
 31 AE 5B 00 6A 5B 56 F8 0127 311 ashp r6,r11,(r10),#0,r11,stkt1(sp);remainder = 0
 0B 7A AE E8 012F 312 blbs stksign(sp),160\$;branch if quotient is negative
 6A 31 AE 5B 34 0133 313 movp r11,stkt1(sp),(r10);copy back into quotient
 04 0138 314 ret :
 31 AE 6A 5B 34 0139 315 155\$: movp r11,(r10),stkt1(sp);copy quotient into temporary
 013E 316 160\$: enter if t1 holds quotient
 5B FEBB CF 00 31 AE 5B 23 013E 317 subp6 r11,stkt1(sp),#0,zero,r11,(r10);make z negative
 6A 0147 318 ret :
 04 0148 319 :
 0149 320 :New division algorithm
 0149 321 :
 0149 322 200\$: :
 0149 323 :
 0149 324 :insure that we have x<y
 0149 325 :
 67 58 6E 1F 37 0149 326 cmpp4 #31,(sp),r8,(r7);x<y?
 31 AE 5B 6E 1F 67 58 19 19 014E 327 blss 220\$;branch if x < y
 1F 67 58 6A 5B 25 0150 328 divp r8,(r7),#31,(sp),r11,(r10);z=x/y
 6E 1F 31 AE 1F 22 015F 329 mulp r11,(r10),r8,(r7),#31,stkt1(sp);t1=y*z
 CO 13 0165 330 subp4 #31,stkt1(sp),#31,(sp);x=x-t1
 331 beql 150\$;branch if remainder is zero

5B 00 FE92 CF 00 09 11 0167 332 brb 230\$
 6A 00 F8 0169 333 220\$: ashp #0,#0,zero,#0,r11,(r10) ;clear quotient
 0171
 0172 334 :
 0172 335 :Assumes z is defined
 0172 336 : 0 < x < y
 0172 337 : and r6=a gives the number of additional digits required.
 0172 338 :Determine r = # of digits for next part of quotient.
 0172 339 :
 59 56 D0 0172 340 230\$: movl r6,r9 ;r=a
 0F 56 D1 0175 341 cmpl r6,#15 ;a>15?
 03 15 0178 342 bleq 240\$;branch if larger
 59 0F D0 017A 343 movl #15,r9 ;r=15
 56 59 C2 017D 344 240\$: subl2 r9,r6 ;update r6 = a
 0180
 0180 345 :
 0180 346 :calculate z2 = min(B-1,[x/y1])
 0180 347 :
 1F 6E 1F 10 AE OF 27 0180 348 250\$: divp #15,stky1(sp),#31,(sp),#31,stkt1(sp);t1=x/y1
 31 AE 0187
 FE71 CF 10 31 AE 1F 37 0189 350 cmpp4 #31,stkt1(sp),#16,nines ;t1>10**16-1?
 09 15 0191 351 bleq 310\$;branch if not
 21 AE FE6B CF OF 34 0193 352 movp #15,nine15,stkz2(sp) ;move in base-1
 0A 11 019A 353 brb 320\$;skip ashp, you have 15 digits
 019C
 019C 354 :
 019C 355 :calculate y2*z2
 019C 356 : t3 = high order 16 digits of y2*z2
 019C 357 : t2 = low order 15 digits of y2*z2
 019C 358 :
 019C 359 310\$:
 OF 00 31 AE 1F FF 8F F8 019C 360 ashp #-1,#31,stkt1(sp),#0,#15,stkz2(sp) ;we only get 15 digits
 21 AE 01A4
 1F 18 AE 10 21 AE OF 25 01A6 361 320\$: mulp #15,stkz2(sp),#16,stky2(sp),#31,stkt1(sp) ;t1=y2*z2
 31 AE 01AE
 10 00 31 AE 1F F1 8F F8 01B0 362 ashp #-15,#31,stkt1(sp),#0,#16,stkt3(sp) ;t3(16)=t1 shifted right 15
 61 AE 01B8
 1F 00 61 AE 10 OF F8 01BA 363 ashp #15,#16,stkt3(sp),#0,#31,stkt4(sp) ;t4(31) = t3 shifted left 15
 OF 31 AE 1F 6A AE 1F 23 01C3 364 subp6 #31,stkt4(sp),#31,stkt1(sp),#15,stkt2(sp) ;t2(15)=t1-t4
 41 AE 01CB
 4C 13 01CD 365 beql 330\$;branch if no borrow required
 01CF
 01CF 366 :
 01CF 367 :borrow is -1, t2 not 0
 01CF 368 :calculate R(H) =
 01CF 369 : t1(31) = 31 high order digits of x(46) - y*z2
 01CF 370 : t2(15) = 15 low order digits of x(46) - y*z2
 01CF 371 :
 1E 10 AE OF 21 AE OF 25 01CF 372 mulp #15,stkz2(sp),#15,stky1(sp),#30,stkt4(sp) ;t4(30)=y1*z2
 6A AE 01D7
 1F 00 6A AE 1E 01 F8 01D9 373 ashp #1,#30,stkt4(sp),#0,#31,stkt1(sp);31 high order of 46
 31 AE 01E0
 1F 6E 1F 31 AE 10 20 01E2 374 addp4 #16,stkt3(sp),#31,stkt1(sp) ;t1(31)=t1(31)+t3(16)
 6A AE 01E9 375 subp6 #31,stkt1(sp),#31,(sp),#31,stkt4(sp) ;t4=x-t1
 1F 6A AE 1F FE07 CF 01 15 01F2 376 bleq 325\$;branch if R(H) negative (t4 <= 0)
 31 AE 01F4 377 subp6 #1,one,#31,stkt4(sp),#31,stkt1(sp) ;t1=t4-1 (borrow 1)

```

OF FE14 CF 10 41 AE OF 23 01FF 378      subp6 #15,stkt2(sp),#16,ten15,#15,stkt4(sp) ;t4=10**15-t2
      6A AE 6A AE 0F 34 0208 379      movp #15,stkt4(sp),stkt2(sp) ;copy back into t2
      63 11 0210 380      brb 370$: ;;
      31 AE 6A AE 1F 34 0212 381      movp #31,stkt4(sp),stkt1(sp) ;make t1 hold high order R(H)
      00AF 31 0218 382      brw 500$: ;R(H) < 0 (can not be zero)
      021B 383:
      021B 384: no borrow, t2 = 0
      021B 385: calculate R(H) = t1(31) = 31 high order digits of x(46) - y*z2
      021B 386: t2(15) = 0 = 15 low order digits of x(46) - y*z2
      021B 387:
      021B 388 330$:
      021B 389      mulp #15,stkz2(sp),#15,stky1(sp),#30,stkt1(sp) ;t1(30)=z2(15)*y1(15)
      31 AE 0F 25 021B 390      ashp #1,#30,stkt1(sp),#0,#31,stkt4(sp) ;31 high order 46
      00 31 AE 1E 01 F8 0223 391      addp4 #16,stkt3(sp),#31,stkt4(sp) ;t4=t3(16)+t4
      6A AE 1F 61 AE 10 20 0225 392      subp6 #31,stkt4(sp),#31,(sp),#31,stkt1(sp) ;t1=x-t4
      1F 6E 1F 6A AE 1F 23 0235 393      bneq 370$                                ;branch if remainder not zero
      31 AE 023C 394:
      023C 395:*****
      0240 396:*
      0240 397:/* R(H) = 0 , i.e.
      0240 398:/*remainder is zero
      0240 399:/*z2(15) holds last non-zero digits of the quotient
      0240 400:/*quotient has not been shifted yet to receive z2
      0240 401:*
      0240 402:/* this is a local routine with no exit other than ret
      0240 403:*****
      0240 404:
      0240 405 340$:
      6A AE 5B 00 6A 5B 59 F8 0240 406      ashp r9,r11,(r10),#0,r11,stkt4(sp) ;t4=z shifted left by r9=r
      6A 6A AE 5B 34 0248 407      movp r11,stkt4(sp),(r10) ;copy back quotient
      59 0F C2 024D 408      subl2 #15,r9 ;shift needed to leave r9 digits
      OF 00 21 AE 0F 59 F8 0250 409      ashp r9,#15,stkz2(sp),#0,#15,stkt1(sp);low order digits of z
      31 AE 0257:
      6A 5B 31 AE 0F 20 0259 410      addp4 #15,stkt1(sp),r11,(r10) ;z=z+z2
      56 D5 025F 411      tstl r6 ;a = 0 ?
      52 13 0261 412      beql 395$: ;branch if a=0 (last iteration)
      31 AE 5B 00 6A 5B 56 F8 0263 413      ashp r6,r11,(r10),#0,r11,stkt1(sp);adjust for scale
      50 7A AE E8 026B 414 350$:
      6A 31 AE 5B 34 026F 415      blbs stksign(sp),410$: ;branch if quotient < 0
      04 0274 416      movp r11,stkt1(sp),(r10) ;back into quotient
      53 19 0275 417 370$:
      0277 418: blss 500$                                ;branch if R(H)<0
      0277 419:*****
      0277 420:/*R(H) > 0 , i.e.
      0277 421:/* t1 > 0
      0277 422:*
      0277 423:*****
      0277 424:
      6A AE 5B 00 6A 5B 59 F8 0277 425 380$:
      6A 6A AE 5B 34 027F 426      ashp r9,r11,(r10),#0,r11,stkt4(sp) ;t4=z shifted left by r9=r
      56 D5 0284 427      movp r11,stkt4(sp),(r10) ;copy back quotient
      1B 13 0286 428      tstl r6 ;a = 0 ?
      0288 429: beql 390$                                ;branch if a=0 (last iteration)

```

PLI
Sym
BY1
CON
CON
DIR
PLI
PY
PZ
STK
STK
STK
STK
STK
X
Y
Z
ZER
PSE

\$AE
_PI
Pha

In
Con
Pas
Syn
Pas
Syn
Pas
Syn
Pas
Cra
Ass
The
64
The
28
3

					0288	430	: a not 0			
					0288	431	:			
1F	6A	5B	21 AE	1F	OF	20	0288	432 addp4 #15,stkz2(sp),r11,(r10) ;z=z+z2		
	00	31	AE	1F	OF	F8	028E	433 ashp #15,#31,stkt1(sp),#0,#31,stkt4(sp) ;t4=t1 shifted left 15		
1F	41	AE	OF	6A	AE	6A	0295	434 addp6 #31,stkt4(sp),#15,stkt2(sp),#31,(sp) ;x=t4+t2		
						21	0297	435 brw 230\$		
							31	02A0	436 :	
								02A3	437 : a = 0	
								02A3	438 :	
OF	00	21	AE	59	OF	C2	02A3	439 390\$: subl2 #15,r9 ;shift needed to leave r9 digits		
				OF	59	F8	02A6	440 ashp r9,#15,stkz2(sp),#0,#15,stkt1(sp);low order digits of z		
				31	AE	6A	02AD	441 addp4 #15,stkt1(sp),r11,(r10) ;z=z+z2		
6A	5B	31	AE	0F	20	02AF	442 395\$: blbs stksign(sp),400\$;branch if quotient <0			
				01	7A	AE	E8	443 ret :		
						04	02B5	444 400\$: :		
							02B9	445 movp r11,(r10),stkt1(sp) ;make copy of quotient		
5B	FD3A	CF	31	AE	6A	5B	34	02BA	446 410\$: subp6 r11,stkt1(sp),#0,zero,r11,(r10) ;make z negative	
	00	31	AE	5B	23	02BF	447 ret			
				6A	04	02C8	448 :			
						02CA	449 :*****			
						02CA	450 :*			
						02CA	451 :*R(H) < 0			
						02CA	452 :* so check if R(H) + Y > = 0			
						02CA	453 :*			
						02CA	454 :*****			
						02CA	455 :*			
						02CA	456 500\$: :			
FD52	CF	10	31	AE	1F	37	02CA	457 cmpp4 #31,stkt1(sp),#16,neg9 ;t1< -(10**16-1) ?		
1F	00	31	AE	1F	66	19	02D2	458 blss 600\$;branch if R(H) + Y < 0		
				OF	F8	02D4	459 ashp #15,#31,stkt1(sp),#0,#31,stkt4(sp) ;shift t1 left 15			
1F	6A	AE	1F	41	AE	6A	02DB	460 subp6 #15,stkt2(sp),#31,stkt4(sp),#31,stkt1(sp) ;t1=t1-t2		
						23	02DD	461 addp4 r8,(r7),#31,stkt1(sp) ;t1=t1+y		
							02E5	462 blss 600\$;branch if R(H) + Y < 0		
						20	02E7	463 beql 530\$;branch if no remainder (first adjust z2)		
						4B	02ED	464 subp4 #1,one,#15,stkz2(sp) ;z2=z2-1		
						3E	02EF	465 ashp r9,r11,(r10),#0,r11,stkt4(sp) ;t4=z shifted left by r9=r		
21	AE	OF	FDOA	CF	01	22	02F1	466 movp r11,stkt4(sp),(r10) ;copy back quotient		
6A	AE	5B	00	6A	5B	59	02F9	467 tstl r6 ;a = 0 ?		
						6A	34	0301	468 beql 510\$;branch if a=0 (last iteration)	
							D5	0306	469 :	
							OE	13	0308	470 : a not 0
								030A	471 :	
								030A	472 addp4 #15,stkz2(sp),r11,(r10) ;z=z+z2	
6A	5B	21	AE	OF	20	030A	473 movp #31,stkt1(sp),(sp) ;x=t1			
6E	31	AE	1F	34	0310	474 brw 230\$				
				FE5A	31	0315	475 :			
						0318	476 : a = 0			
						0318	477 :			
OF	00	21	AE	59	OF	C2	0318	478 510\$: subl2 #15,r9 ;shift needed to leave r9 digits		
				59	F8	031B	479 ashp r9,#15,stkz2(sp),#0,#15,stkt1(sp);low order digits of z			
				31	AE	0322				

```

6A 5B 31 AE OF 20 0324 480      addp4 #15,stkt1(sp),r11,(r10) ;z=z+z2
     8C 7A AE E8 032A 481      blbs stksign(sp),400$ ;branch if quotient <0
     04 032E 482      ret
     032F 483      ;
     032F 484      ;remainder is zero
     032F 485      ;
21 AE OF FCCC CF 01 22 032F 486 530$: subp4 #1,one,#15,stkz2(sp) ;z2=z2-1
     FF06 31 0337 487      brw 340$ ;go add to quotient, then ret
     033A 488      ;
     033A 489      ;*****
     033A 490      ;*
     033A 491      ;*R(H) + Y < 0
     033A 492      ;*calculate L = min(B-1,[X/(y1+1)])
     033A 493      ;* by theorem 5, L=[X/(y1+1)]
     033A 494      ;*
     033A 495      ;*****
     033A 496      ;
10 10 AE OF FCC1 CF 01 21 033A 497 600$: addp6 #1,one,#15,stky1(sp),#16,stkt3(sp) ;t3=y1+1
     61 AE 0343 498      divp #16,stkt3(sp),#31,(sp),#16,stkt1(sp) ;t1=x/t3
     10 6E 1F 61 AE 10 27 0345 499      ;
     31 AE 034C 500      ;split up y2*L
     034E 501      ;t3 = high order 16 digits of y2*L
     034E 502      ;t2 = low order 15 digits of y2*L
     OF 00 31 AE 10 FF 8F F8 034E 503      ashp #-1,#16,stkt1(sp),#0,#15,stkz2(sp) ;we only get 15 digits
     21 AE 0356 504      mulp #15,stkz2(sp),#16,stky2(sp),#31,stkt1(sp) ;t1=y2*z2
     1F 18 AE 10 21 AE OF 25 0358 505      ashp #-15,#31,stkt1(sp),#0,#16,stkt3(sp) ;t3(16)=t1 shifted right 15
     31 AE 0360 506      ashp #15,#16,stkt3(sp),#0,#31,stkt4(sp) ;t4(31) = t3 shifted left 15
     10 00 31 AE 1F F1 8F F8 0362 507      subp6 #31,stkt4(sp),#31,stkt1(sp),#15,stkt2(sp) ;t2(15)=t1-t4
     61 AE 036A 508      ;
     OF 31 AE 1F 6A AE 1F 23 0375 509      ;*****
     41 AE 037D 510      ;*
     037F 511      ;* calculate R(L)
     037F 512      ;*
     037F 513      ;*****
     037F 514      ;
41 13 037F 515      beql 630$ ;branch if no borrow required
     0381 516      ;
     0381 517      ;borrow is -1, t2 not 0
     0381 518      ;calculate R(L) =
     0381 519      ;t1(31) = 31 high order digits of x(46) - y*z2
     0381 520      ;t2(15) = 15 low order digits of x(46) - y*z2
     0381 521      ;note: it is always true that R(L) >= 0
     0381 522      ;
     OF FC92 CF 10 41 AE OF 23 0381 523      subp6 #15,stkt2(sp),#16,ten15,#15,stkt4(sp) ;t4=10**15-t2
     6A AE 038A 524      movp #15,stkt4(sp),stkt2(sp) ;copy back into t2
     1E 10 AE 41 AE 6A AE OF 34 038C 525      mulp #15,stkz2(sp),#15,stky1(sp),#30,stkt4(sp) ;t4(30)=y1*z2
     21 AE 0392 526      ashp #1,#30,stkt4(sp),#0,#31,stkt1(sp) ;high order 31 of 46
     039A 039C 527      ;
     1F 00 6A AE 1E 01 F8 039C 528      ;
     31 AE 03A3 529      ;

```

31 AE 1F 6E 1F 31 AE 10 20 03A5 527 1F 6A AE 1F FC46 CF 01 23 03AC 528 6A AE 31 AE 23 03B3 528 1F 6A AE 1F 31 AE 23 03B5 529 31 AE 23 03BE 529 23 11 03C0 530 03C2 530 03C2 531 : 03C2 532 :no borrow, t2 = 0 03C2 533 :calculate R(L) = t1(31) = x - y1*z2 03C2 534 : 03C2 535 630\$: 1E 10 AE OF 21 AE 31 AE 0F 25 03C2 536 1F 00 31 AE 1E 31 AE 01 F8 03CA 537 6A AE 6A AE 03CC 537 6A AE 1F 61 AE 10 20 03D5 538 1F 6E 1F 6A AE 1F 23 03DC 539 31 AE 03E3 539 03E5 540 : 03E5 541 :***** 03E5 542 :* 03E5 543 :* calculate Z2 = L + R(L)/Y 03E5 544 :* 03E5 545 :***** 03E5 546 : 1F 00 31 AE 1F OF F8 03E5 547 6A AE 1F 41 AE 6A AE 0F 20 03EE 548 OF 6A AE 1F 67 58 27 03F5 549 31 AE 03FC 549 21 AE OF 31 AE 0F FD9E 20 03FE 550 0405 550 0408 551 0408 552 0408 553	addp4 #16,stkt3(sp),#31,stkt1(sp) ;t1(31)=t1(31)+t3(16) subp6 #31,stkt1(sp),#31,(sp),#31,stkt4(sp) ;t4=x-t1 subp6 #1,one,#31,stkt4(sp),#31,stkt1(sp) ;t1=t4-1 (borrow 1) brb 700\$; 531 : 532 :no borrow, t2 = 0 533 :calculate R(L) = t1(31) = x - y1*z2 534 : 535 630\$: mulp #15,stkz2(sp),#15,stky1(sp),#30,stkt1(sp) ;t1(30)=y1*z2 ashp #1,#30,stkt1(sp),#0,#31,stkt4(sp) ;high order 31 of 46 addp4 #16,stkt3(sp),#31,stkt4(sp) ;t4(31)=t4(31)+t3(16) subp6 #31,stkt4(sp),#31,(sp),#31,stkt1(sp) ;t1=x-t4 540 : 541 :***** 542 :* 543 :* calculate Z2 = L + R(L)/Y 544 :* 545 :***** 546 : 700\$: ashp #15,#31,stkt1(sp),#0,#31,stkt4(sp) ;shift t1 left 15 addp4 #15,stkt2(sp),#31,stkt4(sp) ;t4=t4+t2 divp r8,(r7),#31,stkt4(sp),#15,stkt1(sp) ;t1=t4/y addp4 #15,stkt1(sp),#15,stkz2(sp) ;z2=z2+t1 brw 320\$; .end
--	--

PLISDIV PK LONG Symbol Table

κ 9

16-SEP-1984 02:22:23 VAX/VMS Macro V04-00 Page 12
6-SEP-1984 11:39:01 [PLIRTL.SRC]PLIPKDIVL.MAR;1 (1)

BIGNINE	0000000B	R	02
BYTES_TO_SIGN	= 0000000F		
CONSTA	00000018		
DIR...	= 00000001		
NEG9	00000024	R	02
NINE15	00000003	R	02
NINES	00000002	R	02
ONE	00000000	R	02
PLI\$DIV_PK_LONG	0000002D	RG	02
PY	0000000C		
PZ	00000014		
STKLEN	0000007B		
STKSIGN	0000007A		
STKT1	00000031		
STKT2	00000041		
STKT3	00000061		
STKT4	0000006A		
STKY	00000051		
STKY1	00000010		
STKY2	00000018		
STKZ2	00000021		
TEN15	00000018	R	02
X	00000004		
Y	00000008		
Z	00000010		
ZERO	00000001	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
\$ABSS	0000007B (123.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI\$CODE	00000408 (1032.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	13	00:00:00.07	00:00:00.31
Command processing	74	00:00:00.57	00:00:02.13
Pass 1	92	00:00:02.36	00:00:04.43
Symbol table sort	0	00:00:00.05	00:00:00.05
Pass 2	105	00:00:01.19	00:00:02.95
Symbol table output	2	00:00:00.03	00:00:00.20
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	287	00:00:04.29	00:00:10.09

The working set limit was 900 pages.

15119 bytes (30 pages) of virtual memory were used to buffer the intermediate code.

15119 bytes (50 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 26 non-local and 37 local symbols.
553 source lines were read in Pass 1, producing 14 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
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

3
0
3

44 GETS were required to define 3 macros.

There were no errors, warnings or information messages.

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

0308 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIMSGTXT
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECPT
LIS

PLIREAD
LIS

PLIREWRT
LIS

PLIOPEN
LIS

PLIPROTEC
LIS

PLIPUTEDI
LIS

PLIGETLIS
LIS

PLIPKDIVL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIVS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS