

LONG

LONG is a facility to allow long integers handled in BDS C. A long integer is a four array with the least significant part of the stored in bytearray[0]. The integer is stored in 2's complement number with 31 bits of precision.

Operations supported by LONG include addition, subtraction, multiplication (least significant bits returned), division, and modulus. Other operations, such as ASCII to long, long to ASCII, etc., can be programmed efficiently in C.

Calls to LONG are normally "wrapped up" in C functions which, in turn, call the functions

```
char *li(CODE, arg1, arg2, arg3)
char CODE, *arg1, *arg2, *arg3;
```

which returns a pointer to the result. Arg1 and arg3 must be pointers to four byte representations of long integers in the format defined above. In general the operation performed is as if BDS had a data type long and

```
long *arg1, *arg2, *arg3;
*arg1 = *arg2 op *arg3;
```

where op is defined by the following table:

CODE	op	comment
0	+	signed 31 bit result
1	-	signed 31 bit result
2	*	signed low order 31 bits
3	/	signed 31 bit quotient
4	%	positive 31 bit remainder

and, in each case, any overflow is both lost and noted.

TITLE LONG

PAGE 60

```

;
; BDS C is copyright (c) 1980 by Leor Zolman.
; LONG is copyright (c) 1981 by Paul J. Gans.
;
; A notable strangeness in the listing below i
; my version of this assembler REQUIRES that t
; code ex af,af' be CAPITALIZED or it will not
; recognized...-pjpg.
;

```

```

.z80
;

```

```

; Note that the coding technique used here i
; that of William C. Colley, III as reported i
; User's Guide Addenda, v1.32, dated May, 1980
; that Colley's technique is simplified by usi
; MACRO-80 pseudo-op DC to set the high order
; the last character of a string.
;

```

```

0000'
;
; aseq
;
; org 0000h
;
0000 4C C9 dc 'LI' ; first directory entry
0002 0205 dw long
;
0004 80 db 80h ; end of directory
0005 0336 dw f.free ; next free file locatio
;
; org 0200h
;
0200 00 00 00 00 db 0,0,0,0,0 ; always zero if
0204 00
;
0205 00 long: db 0 ; no fn's called by LONG
;
0206 0112 dw f.1rel-f.1beg ; length of LONG
;
; .phase 0
;
; At the start of this function the stack look
; arg3, arg2, arg1, CODE, return address
; with the return address at the top of the st
;
0000 D1 f.1beg: pop de ; DE=returnaddress
0001 E1 pop hl ; CODE
0002 7D ld a,l ; A=CODE
0003 E1 pop hl ; HL=arg1 (result address
0004 DD E1 pop ix ; IX=arg2
0006 FD E1 pop iy ; IY=arg3
0008 E5 push hl ; now restore the stack
0009 E5 push hl
000A E5 push hl
000B E5 push hl
000C D5 push de ; restore return address

```

```

000D    C5                push    bc        ; save BC for caller
000E    E5                push    hl        ; and a copy of arg1 for
;
000F    D9                exx                    ; goto prime register sp
0010    FD 4E 00         ld      c,(iy+0)      ; low order of a
0013    FD 46 01         ld      b,(iy+1)
0016    DD 5E 00         ld      e,(ix+0)
0019    DD 56 01         ld      d,(ix+1)
001C    21 0000         ld      hl,0        ; clear result
;
001F    D9                exx                    ; goto normal register s
0020    FD 4E 02         ld      c,(iy+2)      ; high order of
0023    FD 46 03         ld      b,(iy+3)
0026    DD 5E 02         ld      e,(ix+2)
0029    DD 56 03         ld      d,(ix+3)
002C    21 0000         ld      hl,0        ; clear result
;
002F    FE 00                cp      0            ; check code
0031    CA 00AE         f.1001: jp      z,add
0034    FE 01                cp      1
0036    CA 00B7         f.1002: jp      z,sub
0039    FE 02                cp      2
003B    CA 0086         f.1003: jp      z,mul
;
; The division routine returns two possible va
; the quotient, if CODE was 3, or the modulus,
; CODE was 4. As a sloppy error exit, CODEs h
; than 4 or lower than 0 default to 4. I SAID
; was sloppy.
;
; This routine expects a 64 bit dividend in re
; HLH'L'DED'E' and a 32 bit divisor in registe
; A 32 bit quotient is generated in DED'E' and
; remainder is generated in HLH'L'. For the p
; application the high order 32 bits of the di
; (registers HLH'L') are zeroed.
;
;
003E    08                div:    EX      AF,AF' ; save CODE for later
;
; Because signed divisions are a giant pain, t
; of the result is computed and saved on the s
; Then any negative operands are made positive
; calls to the proper routine.
;
003F    CD 00D7         f.1004: call    sign
;
0042    3E 20                ld      a,32        ; number of iterations
0044    B7                div1:   or      a            ; reset carry flag
;
0045    D9                exx                    ; enter prime register s
0046    ED 42                sbc     hl,bc        ; can we subtract?
;
0048    D9                exx                    ; enter normal register
0049    ED 42                sbc     hl,bc
004B    30 05                jr      nc,div2     ; a carry means no

```

```

;
004D D9          exx          ; enter prime register s
004E 09          add          hl,bc ; restore dividend
;
004F D9          exx          ; enter normal register
0050 ED 4A       adc          hl,bc
0052 3F          div2: ccf          ; quotient bit
;
0053 D9          exx          ; enter prime register s
0054 CB 13       rl          e      ; left shift dividend, s
0056 CB 12       rl          d      ; in new quotient bit
;
0058 D9          exx          ; enter normal register
0059 CB 13       rl          e
005B CB 12       rl          d
;
005D D9          exx          ; prime register space
005E ED 6A       adc          hl,hl ; it's a 64 bit shift, g
;
0060 D9          exx          ; normal register space
0061 ED 6A       adc          hl,hl
0063 3D          dec          a      ; done?
0064 F2 0044     f.1005: jp          p,div1 ; no
;
; CODE must now be tested so that HL can be se
; properly.
;
0067 08          EX          AF,AF' ; regain CODE
0068 FE 03       cp          3
006A 20 0C       jr          nz,modu ; it's a modulus by defa
;
006C D9          exx          ; prime space
006D EB          ex          de,hl ; return quotient
;
006E D9          exx          ; normal space
006F EB          ex          de,hl
0070 F1          pop         af      ; regain sign of result
0071 B7          or          a      ; to flags
0072 FC 0104     f.1006: call        m,negl ; if negative
0075 C3 00C2     f.1007: jp          fin      ; to clean up and go hom
;
0078 CB 3C       modu: srl         h      ; adjust remainder for 1
007A CB 1D       rr          l      ; overshift
;
007C D9          exx          ; prime space
007D CB 1C       rr          h
007F CB 1D       rr          l
;
0081 D9          exx          ; normal space
0082 D1          pop         de      ; dump saved sign, mod i
0083 C3 00C2     f.1008: jp          fin      ; to clean up and go hom
;
;
; The multiplication routine multiplies the co
; of registers BCB'C' by the contents of regis
; and returns the low order 31 bits of the res

```

```

; registers HLH'L'.
;
; Multiplication is also best done on positive
; so we go to the routine again.
;
0086 CD 00D7 mul: call sign
;
0089 3E 20 ld a,32
;
008B D9 mull1: exx ; enter prime space
008C CB 21 sla c ; left shift plier 1 pla
008E CB 10 rl b
;
0090 D9 exx ; enter normal space
0091 CB 11 rl c
0093 CB 10 rl b
0095 30 05 jr nc,mul2 ; if high bit was 0
;
0097 D9 exx ; prime space
0098 19 add hl,de ; add in multiplicand
;
0099 D9 exx ; normal space
009A ED 5A adc hl,de
009C 3D mul2: dec a ; done?
009D 28 07 jr z,mul3 ; yes, clean up and go h
;
009F D9 exx ; hyperspace
00A0 29 add hl,hl ; left shift product
;
00A1 D9 exx ; real space
00A2 ED 6A adc hl,hl
00A4 18 E5 jr mull1 ; and repeat
;
00A6 F1 mul3: pop af ; regain sign of result
00A7 B7 or a ; sign to flags
00A8 FC 0104 f.1009: call m,negl ; if negative
00AB C3 00C2 f.100a: jp fin ; and so to rest at last
;
; The contents of BCB'C' are added to the cont
; DED'E' and the results returned in HLH'L'.
;
00AE D9 add: exx ; to prime
00AF EB ex de,hl
00B0 09 add hl,bc
;
00B1 D9 exx ; to normal
00B2 EB ex de,hl
00B3 ED 4A adc hl,bc
00B5 18 0B jr fin ; to quit
;
; The contents of BCB'C' are subtracted from t
; of DED'E' and the results returned in HLH'L'
;
00B7 D9 sub: exx ; to prime
00B8 B7 or a ; reset carry flag
00B9 EB ex de,hl

```

```

00BA    ED 42                sbc     hl, bc
;
00BC    D9                  exx                    ; to normal
00BD    EB                  ex      de, hl
00BE    ED 42                sbc     hl, bc
00C0    18 00               jr      fin           ; to quit
;
; This is the terminal section of code. It st
; result from HLH'L' into the locations specif
; arg1, restores BC and SP, and exits with HL
; arg1.
;
00C2    DD E1               fin:    pop     ix      ; IX=arg1 (result address
00C4    C1                  pop     bc      ; restore BC while we ar
;
00C5    D9                  exx                    ; to momentum space
00C6    DD 75 00            ld      (ix+0), l
00C9    DD 74 01            ld      (ix+1), h
;
00CC    D9                  exx                    ; to cartesian space
00CD    DD 75 02            ld      (ix+2), l
00D0    DD 74 03            ld      (ix+3), h
00D3    DD E5               push    ix      ; get result address
00D5    E1                  pop     hl      ; into HL
;
00D6    C9                  ret                    ; to real world
;
; This subroutine computes the sign of the res
; multiplication and division and saves it as
; the A register on the stack. It also makes
; negative operands positive. Note that it as
; that HLH'L' are zeroed on entry.
;
00D7    7A                  sign:  ld      a, d    ; contains sign of arg2
00D8    A8                  xor     b        ; generate result sign
00D9    DD E1               pop     ix      ; save subs return address
00DB    F5                  push    af      ; save result sign
;
00DC    7A                  ld      a, d    ; sign of arg2 again
00DD    B7                  or     a        ; to flags
00DE    F2 00EE            f.100b: jp     p, sign1 ; if non-negative
;
; Form the 2's complement of the second argume
; (DED'E').
;
00E1    D9                  exx                    ; far out space
00E2    AF                  xor     a        ; reset A and carry bit
00E3    ED 52                sbc     hl, de
00E5    EB                  ex      de, hl    ; restore answer
00E6    6F                  ld      l, a     ; clean things up
00E7    67                  ld      h, a
;
00E8    D9                  exx                    ; home space
00E9    ED 52                sbc     hl, de
00EB    EB                  ex      de, hl    ; more restore
00EC    6F                  ld      l, a     ; clean here too

```

```

00ED      67                ld      h,a
;
00EE      78      sign1:  ld      a,b      ; sign of arg3
00EF      B7                or      a      ; to flags
00F0      F2 0102      f.100c:  jp      p,sign2 ; if non-negative
;
;      The two's complement of the third argument i
;      in place (BCB'C').
;
00F3      D9                exx                ; prime
00F4      AF      xor      a      ; reset A and carry
00F5      ED 42      sbc      hl,bc
00F7      4D      ld      c,l
00F8      44      ld      b,h
00F9      6F      ld      l,a      ; rezero things
00FA      67      ld      h,a
;
00FB      D9      exx                ; normal
00FC      ED 42      sbc      hl,bc
00FE      4D      ld      c,l
00FF      44      ld      b,h
0100      6F      ld      l,a
0101      67      ld      h,a
;
0102      DD E9      sign2:  jp      (ix)      ; that's all, folks!
;
;      This routine forms the 2's complement of the
;      in HLH'L'.
;
0104      D9      neg1:   exx                ; enter prime space
0105      AF      xor      a      ; zero A and carry flag
0106      EB      ex      de,hl
0107      6F      ld      l,a      ; zero HL register
0108      67      ld      h,a
0109      ED 52      sbc      hl,de
;
010B      D9      exx                ; enter normal space
010C      EB      ex      de,hl
010D      6F      ld      l,a      ; zero HL register
010E      67      ld      h,a
010F      ED 52      sbc      hl,de
;
0111      C9                ret
;
0112      000D      f.1rel:  dw      (f.1end-$)/2      ; num of reloc p
0114      0032      dw      f.1001+1      ; relocation add
0116      0037      dw      f.1002+1
0118      003C      dw      f.1003+1
011A      0040      dw      f.1004+1
011C      0065      dw      f.1005+1
011E      0073      dw      f.1006+1
0120      0076      dw      f.1007+1
0122      0084      dw      f.1008+1
0124      0087      dw      mul+1
0126      00A9      dw      f.1009+1
0128      00AC      dw      f.100a+1

```



```
012A      00DF                      dw      f.100b+1
012C      00F1                      f.lend: dw      f.100c+1
                                     ;
                                     .dephase
                                     ;
0336                                  f.free:                                  ; next free location
                                     ;
                                     end
```


LONG MACRO-80 3.36 17-