```
; SP-2-ZX81 v2.00
; The 2020 ZX81-emulator for the 48K ZX Spectrum

; changes to 1.03
; bug fixes in opcodes
; Flags are not affected in mainprogram so
; A register and F are seperated to add speed to over 40 opcodes
; by not needing to alter F-register
; intruptroutine can be called from any place so
; each opcode gets own mainroutine to speedup emulation
; Mainmenu ED/FD-opcodes placed within own table, saving 4 tstates


; memory
; emulatorregisters A,B,DE,HL F'
; programregisters BC' DE' HL' C(=A) F

; qdel = delay killroutine

brkp equ $83f6      ; debugging option
; main in final version takes out brktst and fit in 3 opcode
macro main

;    jp brktst
     ld a,(de)
     ld l,a
     ld h,b
     ld h,(hl)
     jp (hl)

endmacro


macro   b1main
     inc de
     main
endmacro


frames    equ 16436+32768          ; 49204/49205


; IN port IN A,(N) and IN A,(C) only right emulated to #7F
; in ROM set IY to C000 2x
; no emulation of IX, some unused IX-opcodes are used as extra
opcode
; ISR-table only partly, not 257 bytes perfect
; RST without res 7 saves a few tstates!
; ED does not emulate irrelevant opcodes like OUTI, INI(R), IM.

start     equ #a000

     org  start
```

```
cdflag    equ #c03b

; highbytetable of first 256 opcodes, no shifted (ED/IX/IY/CB)
tab1 equ $/256
     db i00/256, i01/256, i02/256, i03/256
     db i04/256, i05/256, i06/256, i07/256
     db i08/256, i09/256, i0a/256, i0b/256
     db i0c/256, i0d/256, i0e/256, i0f/256

     db i10/256, i11/256, i12/256, i13/256
     db i14/256, i15/256, i16/256, i17/256
     db i18/256, i19/256, i1a/256, i1b/256
     db i1c/256, i1d/256, i1e/256, i1f/256

     db i20/256, i21/256, i22/256, i23/256
     db i24/256, i25/256, i26/256, i27/256
     db i28/256, i29/256, i2a/256, i2b/256
     db i2c/256, i2d/256, i2e/256, i2f/256

     db i30/256, i31/256, i32/256, i33/256
     db i34/256, i35/256, i36/256, i37/256
     db i38/256, i39/256, i3a/256, i3b/256
     db i3c/256, i3d/256, i3e/256, i3f/256

     db i40/256, i41/256, i42/256, i43/256
     db i44/256, i45/256, i46/256, i47/256
     db i48/256, i49/256, i4a/256, i4b/256
     db i4c/256, i4d/256, i4e/256, i4f/256

     db i50/256, i51/256, i52/256, i53/256
     db i54/256, i55/256, i56/256, i57/256
     db i58/256, i59/256, i5a/256, i5b/256
     db i5c/256, i5d/256, i5e/256, i5f/256

     db i60/256, i61/256, i62/256, i63/256
     db i64/256, i65/256, i66/256, i67/256
     db i68/256, i69/256, i6a/256, i6b/256
     db i6c/256, i6d/256, i6e/256, i6f/256

     db i70/256, i71/256, i72/256, i73/256
     db i74/256, i75/256, i76/256, i77/256
     db i78/256, i79/256, i7a/256, i7b/256
     db i7c/256, i7d/256, i7e/256, i7f/256

     db i80/256, i81/256, i82/256, i83/256
     db i84/256, i85/256, i86/256, i87/256
     db i88/256, i89/256, i8a/256, i8b/256
     db i8c/256, i8d/256, i8e/256, i8f/256

     db i90/256, i91/256, i92/256, i93/256
     db i94/256, i95/256, i96/256, i97/256
     db i98/256, i99/256, i9a/256, i9b/256
```

```
     db i9c/256, i9d/256, i9e/256, i9f/256

     db ia0/256, ia1/256, ia2/256, ia3/256
     db ia4/256, ia5/256, ia6/256, ia7/256
     db ia8/256, ia9/256, iaa/256, iab/256
     db iac/256, iad/256, iae/256, iaf/256

     db ib0/256, ib1/256, ib2/256, ib3/256
     db ib4/256, ib5/256, ib6/256, ib7/256
     db ib8/256, ib9/256, iba/256, ibb/256
     db ibc/256, ibd/256, ibe/256, ibf/256

     db ic0/256, ic1/256, ic2/256, ic3/256
     db ic4/256, ic5/256, ic6/256, ic7/256
     db ic8/256, ic9/256, ica/256, icb/256
     db icc/256, icd/256, ice/256, icf/256

     db id0/256, id1/256, id2/256, id3/256
     db id4/256, id5/256, id6/256, id7/256
     db id8/256, id9/256, ida/256, idb/256
     db idc/256, idd/256, ide/256, idf/256

     db ie0/256, ie1/256, ie2/256, ie3/256
     db ie4/256, ie5/256, ie6/256, ie7/256
     db ie8/256, ie9/256, iea/256, ieb/256
     db iec/256, ied/256, iee/256, ief/256

     db if0/256, if1/256, if2/256, if3/256
     db if4/256, if5/256, if6/256, if7/256
     db if8/256, if9/256, ifa/256, ifb_/256
     db ifc/256, ifd/256, ife/256, iff/256

; Now 256 bytes reserved for IY-opcodes and table value
; not used spaced is filled with emulation of some opcodes
; if possible
fdbyte    equ $

i00  b1main
; decimal 41222 THE EMULATOR STARTS HERE WITH JP TO EMULST
emstrt    jp emulst      ; FIXED POINT TO START EMULATOR

fd09 db fd2ok%256        ; do "ADD IY,BC",

; Most 2 bytes FD-opcodes are doen here
fd2ok     ld (opix2+1),a      ; write command to opix2
     exx            ; flags are OK
opix2     db #fd,0       ; the command
     exx
     b1main

     block fdbyte+#19-$,0
fd19 db fd2ok%256   ; do "ADD IY,DE"
```

```
     block fdbyte+#21-$,0
fd21 db fd4ok%256   ; do "LD IY,NN"
fd22 db fd4inn%256  ; do "LD (NN),IY"
fd23 db fd2ok%256   ; do "INC IY"

     db 0,0,0,0,0
fd29 db fd2ok%256   ; do "ADD IY,IY"
fd2a db fd4inn%256  ; do "LD IY,(NN)"
fd2b db fd2ok%256   ; do "DEC IY"

     block fdbyte+#34-$
fd34 db fd3ok%256   ; do "INC (IY)"
fd35 db fd3ok%256   ; do "DEC (IY)"
fd36    db fd4ok%256     ; do "LD (IY),N"
        db 0,0
fd39 db fd2ok%256   ; do "ADD IY,SP"

     db 0,0
i3c  inc c          ; INC A
     b1main

     block fdbyte+#46-$,0
fd46 db fd3ok%256   ; do "LD B,(IY)"

     block fdbyte+#4e-$,0
fd4e    db fd3ok%256     ; do "LD C,(IY)"

     db 0,0,0
i52     jp i00

     block fdbyte+#56-$,0
fd56 db fd3ok%256   ; do "LD D,(IY)"

     block fdbyte+#5e-$,0
fd5e    db fd3ok%256     ; do "LD E,(IY)"

     block fdbyte+#66-$,0
fd66 db fd3ok%256   ; do "LD H,(IY)"

     block fdbyte+#6e-$,0
fd6e    db fd3ok%256     ; do "LD L,(IY)"

     block fdbyte+#70-$,0
fd70    db fd3ok%256     ; do "LD (IY),B"
fd71    db fd3ok%256     ; do "LD (IY),C"
fd72    db fd3ok%256     ; do "LD (IY),D"
fd73    db fd3ok%256     ; do "LD (IY),E"
fd74    db fd3ok%256     ; do "LD (IY),H"
fd75    db fd3ok%256     ; do "LD (IY),L"
     db 0
fd77    db fd3ok%256     ; do "LD (IY),A"
fd3ok   jp dofd3
```

```
     block fdbyte+#7e-$,0
fd7e    db fd3ok%256     ; do "LD A,(IY)"

     block fdbyte+#86-$,0
fd86 db fd3ok%256   ; do "ADD A,(IY)"


     block fdbyte+#8e-$,0
fd8e    db fd3ok%256     ; do "ADC A,(IY)"

     block fdbyte+#96-$,0
fd96 db fd3ok%256   ; do "SUB (IY)"

     block fdbyte+#9e-$,0
fd9e    db fd3ok%256     ; do "SBC A,(IY)"

     block fdbyte+#a1-$,0

isr  di             ; INTRUPT JUMPS TO #A1A1
isrmod    jp isr1             ; Can go to other routine after
loading

     block fdbyte+#a6-$,0
fda6 db fd3ok%256   ; do "AND,(IY)"

     block fdbyte+#ae-$,0
fdae    db fd3ok%256     ; do "XOR (IY)"

     block fdbyte+#b6-$,0
fdb6 db fd3ok%256   ; do "OR (IY)"

     block fdbyte+#be-$,0
fdbe    db fd3ok%256     ; do "CP (IY)"

fd4i2     ld (opc4+2),hl
opc4 db #fd,0,0,0
     jp i00         ; byte short of code

     block fdbyte+#cb-$,0
fdcb db fd4ok%256   ; all IY CB opcodes

fd4ok   jp fd4ok1

fd4inn    ld (opc4+1),a
     inc de
     ld a,(de)
     ld l,a
     inc de
     ld a,(de)
     set 7,a
     ld h,a
     jp fd4i2
```

```
     block fdbyte+#e1-$,0
fde1 db fd2ok%256
     nop
fde3 db fd2ok%256
     nop
fde5 db fd2ok%256
     db 0,0,0
fde9 db fd2ok%256

; a small piece of translated ROM starts here
wrchar    ld a,c         ; get value of A
     exx       ; get mainregisters
     call L083E     ; execute translated ROM
     exx       ; save mainregisters
     ld c,a         ; save A-reg
     jp ic9         ; exit with RET

     block fdbyte+#f9-$,0
fdf9 db fd2ok%256
     db 0,0,0
; The SHIFTED opcodes FD and ED (CB not!) are placed
; inside their own table. This saves 4 tstates to set H-reg
ifd  inc de         ; prefix fd
     ld a,(de) ; fetch code
     ld l,a         ; point to code of ix/iy
     ld l,(hl) ; fetch code ; NO FETCHING H NOW!!
     jp (hl)         ; in table on unused positions do all code

; next 253 bytes (FD runs over begin of this part) for ED-opcodes
edbyte    equ fdbyte+256
     block edbyte+3-$,0

; Extra opcodes for own commands
     db tred%256         ; ed03
     db tred%256         ; ed03
     db tred%256         ; ed03

     db tred%256         ; ed06
     db tred%256         ; ed06
     db tred%256         ; ed06

     db tred%256         ; ed09
     db tred%256         ; ed09
     db tred%256         ; ed09

     db tred%256         ; ed0c
     db tred%256         ; ed0c
     db tred%256         ; ed0c

     db tred%256         ; ed0f
     db tred%256         ; ed0f
     db tred%256         ; ed0f
```

```
db tred%256         ; ed12
db tred%256         ; ed12
db tred%256         ; ed12

db tred%256         ; ed15
db tred%256         ; ed15
db tred%256         ; ed15

db tred%256         ; ed18
db tred%256         ; ed18
db tred%256         ; ed18

db tred%256         ; ed1b
db tred%256         ; ed1b
db tred%256         ; ed1b

db tred%256         ; ed1e
db tred%256         ; ed1e
db tred%256         ; ed1e

db tred%256         ; ed21
db tred%256         ; ed21
db tred%256         ; ed21

db tred%256         ; ed24
db tred%256         ; ed24
db tred%256         ; ed24

db tred%256         ; ed27
db tred%256         ; ed27
db tred%256         ; ed27

db tred%256         ; ed2a
db tred%256         ; ed2a
db tred%256         ; ed2a

db tred%256         ; ed2d
db tred%256         ; ed2d
db tred%256         ; ed2d

db tred%256         ; ed30    CALC1
db tred%256         ; ed30
db tred%256         ; ed30

db tred%256         ; ed33    CALC2
db tred%256         ; ed33
db tred%256         ; ed33

db tred%256         ; ed36    CALC3
db tred%256         ; ed36
db tred%256         ; ed36

db tred%256         ; ed39    CALC4
```

```
        db tred%256         ; ed39
        db tred%256         ; ed39

        db tred%256         ; ed3c
        db tred%256         ; ed3c
        db tred%256         ; ed3c

        block edbyte+#40-$

        db edb2%256         ; ed40 IN B,(C)
        db mained%256       ; ed41 OUT (C),B
        db edb2%256         ; ed42 SBC HL,BC
        db edinn%256        ; ed43 LD (NN),BC
        db edb2%256         ; ed44 NEG
        db edb2%256         ; ed45 RETN
        db mained%256       ; ed46 IM 0
        db mained%256       ; ed47 LD I,A
        db edb2%256         ; ed48 IN C,(C)
        db mained%256       ; ed49 OUT (C),C
        db edb2%256         ; ed4a ADC HL,BC
        db edinn%256        ; ed4b LD BC,(NN)
        db 0
        db edb2%256         ; ed4d RETI
        db 0,edb2%256       ; ed4f LD R,A
        db edb2%256         ; ed50 IN D,(C)
        db mained%256       ; ed51 OUT (C),D
        db edb2%256         ; ed52 SBC HL,DE
        db edinn%256        ; ed53 LD (NN),DE
        db 0,0
        db mained%256       ; ed56 IM 1
        db mained%256       ; ed57 LD A,I
        db edb2%256         ; ed58 IN E,(C)
        db mained%256       ; ed59 OUT (C),E
        db edb2%256         ; ed5a ADC HL,DE
        db edinn%256        ; ed5b LD DE,(NN)
        db 0,0
        db mained%256       ; ed5e IM 2
        db edb2%256         ; ed5f LD A,R

        db edb2%256         ; ed60 IN H,(C)
        db mained%256       ; ed61 OUT (C),H
        db edb2%256         ; ed62 SBC HL,HL
        db edinn%256        ; ed63 LD (NN),HL

        db 0,0,0
        db edirr%256        ; ed67 RRD
        db edb2%256         ; ed68 IN L,(C)
        db mained%256       ; ed69 OUT (C),L
        db edb2%256         ; ed6a ADC HL,HL
        db edinn%256        ; ed6b LD HL,(NN)

edspi   jp spinn            ; no room here, so JP outside
```

```
     db edirr%256        ; ed6f RLD

     db 0,0
     db ed72%256         ; ed72    SBC HL,SP
     db edisp%256        ; ed73    LD (NN),SP

edisp   jp innsp         ; also outside these table

     db 0
     db edin%256         ; ed78 IN A,(C)
     db mained%256       ; ed79 OUT (C),A
     db eder%256         ; ed7a ADC HL,SP
     db edspi%256        ; ed7b LD SP,(NN)

; same method as FD, write opcode in routine and execute
edb2 ld (ed2ex+1),a
     ld a,c
     exx
ed2ex     db #ed,0
     exx
     ld c,a
     jp i00

edinn     ex de,hl
     ld e,edinn2%256+1
     ld (de),a
     inc hl
     inc de          ; NOT "INC E", CHANGES FLAGS!
     ld a,(hl)
     ld (de),a
     inc de
     inc hl
     ld a,(hl)
     set 7,a
     ld (de),a
     exx
edinn2  db #ed,0,0,0
     exx
     ex de,hl
     jp i00

     block edbyte+#a0-$,0

     db edirr%256        ; eda0 LDI
     db edirr%256        ; eda1 CPI
     db edirr%256        ; eda2 INI
     db mained%256       ; eda3 OUTI
     db 0,0,0,0
     db edirr%256        ; eda8 LDD
     db edirr%256        ; eda9 CPD
     db edirr%256        ; edaa IND
     db mained%256       ; edab OUTD
```

```
; Extra ED-opcodes all come here
tred ld l,a         ; opcode again
     ld h,edbyte/256+1   ; translated table (NOT INC H, changes
FLAGS)
     jp (hl)        ; goto translation

     db edirr%256        ; edb0 LDIR
     db edirr%256        ; edb1 CPIR
     db edirr%256        ; edb2 INIR
     db mained%256       ; edb3 OTIR
     db 0,0,0,0
     db edirr%256        ; edb8 LDDR
     db edirr%256        ; edb9 CPDR
     db edirr%256        ; edba INDR
     db mained%256       ; edbb OTDR

; Some ED-opcodes are not emulated but will show this error
; I don't expect these to be used in a ZX81, If so please report!
eder    ld hl,16384
     dec (hl)
     jr eder

; Commands like LDIR / LDDR come here
edirr     ld (edirr2+1),a     ; write opcode
     ld a,c         ; get original A
     exx       ; get mainregisters
     set 7,h        ; point to right HL
     set 7,d        ; point to right DE
edirr2    db #ed,0
     res 7,d        ; undo change highbyte
     res 7,h        ; undo change highbyte
     exx
     ld c,a
mained  b1main

ed72    ld hl,#8000
     ex af,af' ; SAVE F
     add hl,sp ; Undo shift of SP
     ex af,af'
     push hl        ; "normal" SP on stack
     exx
     ex de,hl  ; swap DE,HL
     ex (sp),hl     ; SP in HL, HL in DE, DE on stack
     ex de,hl  ; SP in DE
     sbc hl,de ; SBC HL,SP
     pop de         ; original DE
     exx
     jp i00
     db 0,0

; Like FD, inside table
ied  inc de
     ld a,(de)
```

```
     ld l,a
     ld l,(hl) ; table has room to store emulation
     jp (hl)

edin exx
     in a,(c)
     exx
     and #7f
     or 64
     ld c,a
     b1main

; the jumps to all extra ED-opcodes
     block edbyte+256+3-$,0
ed03    jp testrm   ; TEST ROOM    L0EC5 v
ed06    jp mkroom   ; MAKE ROOM    L099E v
ed09 jp wrchar ; WR CHAR      L083E v
ed0c jp decode ; DECODE  L07BD v
edof jp locate ; LOCATE  L0918 v
ed12 jp rest18 ; RST #18 L0018 v
ed15 jp rest20 ; RST #20 L0020 v
ed18 jp loader
ed1b jp runin  ; v after this again full run
ed1e jp scann  ; SCANNING      L0F55 v
ed21 jp scankb ; SCANKB v
ed24 jp ench        ; L0809 v
ed27    jp prsp          ; L07F5 v
ed2a jp blines ; L0A2C v
ed2d jp nxtl        ; L066C v

; CALCULATOR ROUTINES
ed30 jp calc1  ; L199D v
ed33 jp calc2  ; L19A0 v
ed36 jp calc3  ; L19A7 v
ed39 jp calc4  ; L19Ae v

ed3c jp new         ; New command v

; Now the opcode-emulation follows.
; Each first available opcode is placed after the emulation
; Due to found bugs sometimes a JP is made to fit changed code
i3f  ccf       ; CCF
i40  b1main         ; LD B,B

i46  exx       ; LD B,(HL)
     set 7,h
     ld b,(hl)
     res 7,h
     exx
     b1main

i53  exx       ; LD D,E
     ld d,e
```

```
     exx
     b1main

i5c  exx
     ld e,h
     exx
     b1main         ; 6 bytes is each bmain-macro

i65  exx
     ld h,l
     exx
     b1main         ; 6

i6e  exx       ; LD L,(HL)
     set 7,h
     ld l,(hl)
     res 7,h
     exx
     b1main

i7b  exx
     ld a,e
     exx
     ld c,a         ; keep A in C
i7f  b1main

i85  ld a,c
     exx
     add a,l
     exx
     ld c,a
     b1main

i90     ld a,c      ; SUB B
     exx
     sub b
     exx
     ld c,a
     b1main

i9b     ld a,c ; SBC A,E
     exx
     sbc a,e
     exx
     ld c,a
     b1main

ia6  ld a,c
     exx
     set 7,h
     and (hl)
     res 7,h
     exx
```

```
     ld c,a
     b1main

ib5  ld a,c
     exx
     or l
     exx
     ld c,a
     b1main

ic0  jp nz,ic9 ; RET NZ
     b1main

ic9  pop de
     set 7,d
     main

id1  exx
     pop de
     exx
     b1main

ida  jp c,ic3
     inc de
     inc de
     b1main

ie5  exx
     push hl
     exx
     b1main

iee  inc de          ; XOR N
     ld a,(de)
     xor c
     ld c,a
     b1main

if8  jp m,ic9  ; RET M
     b1main

i01  inc de          ; LD BC,nn
     ld a,(de)
     exx
     ld c,a
     exx
     inc de
     ld a,(de)
     exx
     ld b,a
     exx
     b1main
```

```
i11  inc de          ; LD DE,nn
     ld a,(de)
     exx
     ld e,a
     exx
     inc de
     ld a,(de)
     exx
     ld d,a
     exx
     b1main          ; 6

i21  inc de          ; LD HL,nn
     ld a,(de)
     exx
     ld l,a
     exx
     inc de
     ld a,(de)
     exx
     ld h,a
     exx
     b1main          ; 6

i31  inc de
     ld a,(de)
     ld l,a
     inc de
     ld a,(hl)
     ld h,a
     set 7,h
     ld sp,hl
     b1main          ; 6

     nop
i41  exx        ; LD B,C
     ld b,c
     exx
     b1main          ; 6

i4a  exx
     ld c,d
     exx
     b1main          ; 6

     db 0
i54  exx
     ld d,h
     exx
     b1main

i5d  exx
     ld e,l
```

```
     exx
     b1main

i66  exx             ; LD H,(HL)
     set 7,h
     ld h,(hl)
     exx
     b1main

i71  exx             ; LD (HL),C
     set 7,h
     ld (hl),c
     res 7,h
     exx
     b1main

i7e  exx
     set 7,h
     ld a,(hl)
     res 7,h
     exx
     ld c,a
     b1main

i8c  ld a,c
     exx
     adc a,h
     exx
     ld c,a
     b1main

i97  sub a
     ld c,a
     b1main

i9f  sbc a,a
     ld c,a
     b1main

ia7  ld a,c
     and a           ; ONLY CHANGE WILL B RESET CARRY
     b1main

iaf  xor a
     ld c,a
     b1main

ib7  ld a,c
     or a
     b1main

cls  ld (hl),b
     inc hl
```

```
     cp h
     jr nz,cls
     ret

     block 2,0
ic7  jp romstrt          ; RST 0

ica  jp z,ic3
     inc de
     inc de
     b1main

id5  exx
     push de
     exx
     b1main

ide  inc de
     ld a,(de)
     ld h,a
     ld a,c
     sbc a,h
     ld c,a
     b1main

iea  jp pe,ic3
     inc de
     inc de
     b1main

if5  ld a,c
     push af
     b1main

     db 0
ife  inc de
     ld a,(de)
     ld h,a
     ld a,c
     cp h
     b1main

i09  exx
     add hl,bc
     exx
     jp i00              ; same speed as before to fit in i0f
i0f  jp i0fc             ; same speed as before main in i0fc

i12  ld a,c         ; LD (DE),A
     exx
     set 7,d
     ld (de),a
     res 7,d
```

```
     exx
     b1main

i20  inc de
     jp nz,i10tr
     b1main

i2a  inc de          ; LD HL,(NN)
     ld a,(de)
     ld l,a
     inc de
     ld a,(de)
     ld h,a
     set 7,h
     ld a,(hl)
     inc hl
     ld h,(hl)
     ld l,a
     push hl
     exx
     pop hl
     exx
     b1main

     db 0,0
i42     exx          ; LD B,D
     ld b,d
     exx
     b1main

i4b  exx
     ld c,e
     exx
     b1main

     db 0
i55     exx               ; LD D,L
     ld d,l
     exx
     b1main

i5e  exx             ; LD E,(HL)
     set 7,h
     ld e,(hl)
     res 7,h
     exx
     b1main

i6b  exx
     ld l,e
     exx
     b1main
```

```
i74  exx            ; LD (HL),H
     ld a,h
     set 7,h
     ld (hl),a
     ld h,a
     exx
     b1main

i81  ld a,c              ; ADD A,C
     exx
     add a,c
     exx
     ld c,a
     b1main

     nop
i8d  ld a,c
     exx
     adc a,l
     exx
     ld c,a
     b1main

i98     ld a,c ; SBC A,B
     exx
     sbc a,b
     exx
     ld c,a
     b1main

ia3     ld a,c ; AND E
     exx
     and e
     exx
     ld c,a
     b1main

iae  ld a,c
     exx
     set 7,h
     xor (hl)
     res 7,h
     exx
     ld c,a
     b1main

     db 0
ibe  ld a,c
     exx
     set 7,h
     cp (hl)
     res 7,h
     exx
```

```
     b1main

icc  jp z,icd
     inc de
     inc de
     b1main

id7  inc de
     push de
     ld de,#8010
     main

ie1  exx
     pop hl
     exx
     b1main

     db 0
ieb  exx
     ex de,hl
     exx
     b1main

if4  jp p,icd  ; CALL P
     inc de
     inc de
     b1main

iff  inc de              ; probably not used on zx81
     push de             ; RET will work ok
     ld de,#8038         ; NOT USED, SO NOT FULL SPEEDED
     jp i00+1

i07  ld a,c
     rlca      ; RLC A is NOT the SAME
     ld c,a
     b1main

i10  inc de
     exx
     djnz i10tr-1
     exx
     b1main

i1b  exx
     dec de
     exx
     b1main

     exx
i10tr     ex af,af' ; save F A' is in memory
     ld a,(de)
     ld l,a
```

```
     add a,a
     sbc a,a
     ld h,a
     add hl,de
     ex af,af' ; get F
     ex de,hl
     b1main


i34  exx
     set 7,h
     inc (hl)
     res 7,h
     exx
     b1main

     db 0,0
i43  exx             ; LD B,E
     ld b,e
     exx
     b1main

i4c  exx             ; LD C,H
     ld c,h
     exx
     b1main

     nop
i56  exx
     set 7,h
     ld d,(hl)
     res 7,h
     exx
     b1main

i63  exx
     ld h,e
     exx
     b1main

i6c  exx
     ld l,h
     exx
     b1main

i75  exx
     set 7,h
     ld (hl),l
     res 7,h
     exx
     b1main

i82  ld a,c
```

```
     exx
     add a,d
     exx
     ld c,a
     b1main

     nop
i8e  ld a,c
     exx
     set 7,h
     adc a,(hl)
     res 7,h
     exx
     ld c,a
     b1main

i9d  ld a,c
     exx
     sbc a,l
     exx
     ld c,a
     b1main

ia8     ld a,c
     exx
     xor b
     exx
     ld c,a
     b1main

ib3     ld a,c
     exx
     or e
     exx
     ld c,a
     b1main

     nop
     db 0,0,0,0,0,0,0
ic6  inc de         ; ADD A,N
     ld a,(de) ; ld a,N
     add a,c        ; ADD N,A
     ld c,a         ; save A
     b1main

id0  jp nc,ic9 ; RET NC
     b1main

id9  push de             ; save PC
     ld a,c
hla  ld hl,0        ; get hl'
dea  ld de,0             ; get de'
bca  ld bc,0             ; get bc'
```

```
     exx            ; swap registers
     ld (hla+1),hl       ; store hl'
     ld (dea+1),de       ; store de'
     ld (bca+1),bc       ; store bc'
     pop de              ; retrieve PC
     ld b,tab1      ; repair table pointer normtab
     ld c,a
     b1main

     db 0,0
ifc  jp m,icd
     inc de
     inc de
     b1main

locate    ld a,c
     exx
     call L0918
     exx
     ld c,a
     jp ic9

     db 0,0
i13  exx
     inc de
     exx
     b1main

i1c  exx
     inc e
     exx
     b1main

i25     exx
     dec h
     exx
     jp i00
i2b  jp i2bc

i2e  inc de         ; LD L,n
        ld a,(de)
     exx
        ld l,a
        exx
        b1main

i39  ld (spsv+1),sp
     exx
     push de
spsv ld de,0
     res 7,d
     add hl,de
     pop de
```

```
     exx
     b1main

i4d  exx
     ld c,l
     exx
     b1main

     db 0
i57     ld a,c
     exx
     ld d,a
     exx
i5b  b1main

i61  exx
     ld h,c
     exx
i64  b1main

i6a  exx
     ld l,d
     exx
i6d  b1main

i73  exx
     set 7,h
     ld (hl),e
     res 7,h
     exx
     b1main

i80     ld a,c
     exx
     add a,b
     exx
     ld c,a
     b1main

i8b     ld a,c
     exx
     adc a,e
     exx
     ld c,a
     b1main

i96  ld a,c
     exx
     set 7,h
     sub (hl)
     res 7,h
     exx
     ld c,a
```

```
     b1main

ia5  ld a,c
     exx
     and l
     exx
     ld c,a
     b1main

ib0     ld a,c
     exx
     or b
     exx
     ld c,a
     b1main

ibb     ld a,c
     exx
     cp e
     exx
     b1main

ic5  exx
     push bc
     exx
     b1main

ice  inc de
     ld a,(de)
     adc a,c
     ld c,a
     b1main

id8  jp c,ic9
     b1main

     nop
ie2  jp nc,ic3
     inc de
     inc de
     b1main

     nop
     nop
ief  inc de
     push de
     ld de,#8028
     jp i00+1  ; enough translated during CALC

if7  inc de
     push de        ; RET will work ok
     ld de,#8030
     main
```

```
     nop
i02  ld a,c
     exx
     set 7,b
     ld (bc),a
     res 7,b
     exx
     b1main

     db 0,0,0,0
i14  exx
     inc d
     exx
     b1main

i1d     exx
     dec e
     exx
     b1main

i26  inc de          ; LD H,n
        ld a,(de)
     exx
        ld h,a
        exx
     b1main

     nop
i32  inc de          ; LD (NN),A
     ld a,(de)
     ld l,a
     inc de
     ld a,(de)
     ld h,a
     set 7,h
     ld (hl),c
     b1main

     db 0,0,0
i44  exx
     ld b,h
     exx
     b1main

     nop
i4e  exx        ; LD C,(HL)
     set 7,h
     ld c,(hl)
     res 7,h
     exx
     b1main
```

```
     db 0,0,0,0
i5f     ld a,c
     exx
     ld e,a
     exx
     b1main

i69  exx
     ld l,c
     exx
     b1main

i72  exx
     set 7,h
     ld (hl),d
     res 7,h
     exx
     b1main

     db 0,0,0,0
i83     ld a,c
     exx
     add a,e
     exx
     ld c,a
     b1main

     nop
i8f  ld a,c
     adc a,a
     ld c,a
     b1main

     db 0
i99  ld a,c
     exx
     sbc a,c
     exx
     ld c,a
     b1main

ia4  ld a,c
     exx
     and h
     exx
     ld c,a
     b1main

     db 0,0
ib1  ld a,c
     exx
     or c
     exx
```

```
     ld c,a
     b1main

ibc  ld a,c
     exx
     cp h
     exx
     b1main

     db 0,0
ic8  jp z,ic9  ; RET Z
     b1main

     db 0
id2  jp nc,ic3
     inc de
     inc de
     b1main

idd  inc de
     ld a,(de) ; stepsize is 3
     ld l,a
     ld h,ddtab     ; table holds JP NN only
     jp (hl)

ie3  exx
     ex (sp),hl
     exx
     b1main

iec  jp pe,icd
     inc de
     inc de
     b1main

     db 0,0,0
ifa  jp m,ic3
     inc de
     inc de
     b1main

i05  exx
     dec b
     exx
     b1main

i0e  inc de         ; LD C,n
        ld a,(de)
     exx
        ld c,a
        exx
     b1main
```

```
i19  exx
     add hl,de
     exx
     b1main

i22  ex de,hl       ; LD (NN),HL
     inc hl
     ld e,(hl)
     inc hl
     ld d,(hl)
     set 7,d
     ex de,hl
     ld (nnhl+1),hl
     exx
nnhl ld (0),hl
     exx
     b1main

i38  inc de
     jp c,i10tr
     b1main

     db 0,0,0
i45  exx
     ld b,l
     exx
     b1main

     nop
i4f     ld a,c
     exx
     ld c,a
     exx
     b1main

i59  exx
     ld e,c
     exx
     b1main

i62  exx
     ld h,d
     exx
     b1main

     db 0,0,0,0
i6f     ld a,c
     exx
     ld l,a
     exx
     b1main

i79  exx
```

```
     ld a,c
     exx
     ld c,a
     b1main

     nop
i84  ld a,c
     exx
     add a,h
     exx
     ld c,a
     b1main

     db 0,0
i91  ld a,c
     exx
     sub c
     exx
     ld c,a
     b1main

i9c  ld a,c
     exx
     sbc a,h
     exx
     ld c,a
     b1main

     db 0,0
ia9  ld a,c
     exx
     xor c
     exx
     ld c,a
     b1main

ib4  ld a,c
     exx
     or h
     exx
     ld c,a
     b1main

ibf  ld a,c
     cp a
     b1main


     db 0,0,0

     db 0
icb  inc de
     ex af,af'
```

```
     ld a,(de)
     and 7
     cp 6
     ld a,(de)
     jp z,icbhl
     ld (icbn+1),a
     ex af,af'
     ld a,c
     exx
icbn bit 1,a
     exx
     ld c,a
     b1main

     db 0,0
ie8  jp pe,ic9 ; RET PE
     b1main

     db 0,0,0,0,0
if6  inc de
     ld a,(de)
     or c
     ld c,a
     b1main

     db 0,0,0
i03  exx
     inc bc
     exx
     b1main          ; 6

i0c  exx
     inc c
     exx
     b1main

i15     exx
     dec d
     exx
     b1main

i1e  inc de          ; LD E,n
        ld a,(de)
     exx
        ld e,a
        exx
        b1main

i29  exx
     add hl,hl
     exx
     b1main
```

```
     db 0
i33  inc sp
     b1main

i3a  inc de
     ld a,(de)
     ld l,a
     inc de
     ld a,(de)
     ld h,a
     set 7,h
     ld c,(hl)       ; LD A,(NN)
     b1main

i49  b1main

     db 0
i50  exx
     ld d,b
     exx
     b1main

     nop
i5a  exx
     ld e,d
     exx
     b1main

     db 0,0,0,0
i67     ld a,c
     exx
     ld h,a
     exx
     b1main

     db 0,0,0,0,0,0
i77     ld a,c
     exx
     set 7,h
     ld (hl),a
     res 7,h
     exx
     b1main

     db 0
i86  ld a,c
     exx
     set 7,h
     add a,(hl)
     res 7,h
     exx
     ld c,a
     b1main
```

```
i95  ld a,c
     exx
     sub l
     exx
     ld c,a
     b1main

ia0     ld a,c
     exx
     and b
     exx
     ld c,a
     b1main

iab     ld a,c
     exx
     xor e
     exx
     ld c,a
     b1main

ib6  ld a,c
     exx
     set 7,h
     or (hl)
     res 7,h
     exx
     ld c,a
     b1main

     db 0,0,0,0,0,0,0,0,0,0

icf  inc de
     push de
     ld de,#8008
     jp i00+1       ; ERROR RST DOESN'T NEED SPEEDUP

     db 0,0,0,0
idb  inc de              ; IN A,(N)
     ld a,(de)
     ld (idbin+1),a
     ld a,c
idbin     in a,(0)
     and #7f
     or 64
     ld c,a
     b1main

     db 0,0
if0  jp p,ic9  ; RET P
if3  b1main
```

```
if9  exx
     set 7,h
     ld sp,hl
     res 7,h
     exx
     b1main


     db 0,0

i08  jp i08a

i0b  exx
     dec bc
     exx
     b1main

     db 0,0,0
i17  ld a,c
     rla
     ld c,a
     b1main

     db 0,0,0
i23  exx
     inc hl
     exx
     b1main

i2c  exx
     inc l
     exx
     b1main

i35  exx
     set 7,h
        dec (hl)
     res 7,h
     exx
     b1main

     db 0,0,0,0,0
i47     ld a,c
     exx
     ld b,a
     exx
     b1main

i51  exx
     ld d,c
     exx
     b1main
```

```
     db 0,0,0,0,0,0
i60  exx             ; LD H,B
     ld h,b
     exx
     b1main

     db 0,0,0,0,0,0,0
i70  exx             ; LD (HL),B
     set 7,h
     ld (hl),b
     res 7,h
     exx
     b1main

i7d  exx
     ld a,l
     exx
     ld c,a
     b1main

i87  ld a,c
     add a,a
     ld c,a
     b1main

     db 0,0
i92  ld a,c
     exx
     sub d
     exx
     ld c,a
     b1main

     db 0
i9e  ld a,c
     exx
     set 7,h
     sbc a,(hl)
     exx
     ld c,a
     b1main

     db 0,0
iad  ld a,c
     exx
     xor l
     exx
     ld c,a
     b1main

     db 0,0
iba  ld a,c
     exx
```

```
     cp d
     exx
     b1main

ic4  jp nz,icd
     inc de
     inc de
     b1main


     db 0,0,0,0,0
id4  jp nc,icd
     inc de
     inc de
     b1main

idf  inc de
     push de
     ld de,#8018
     main

ie9  exx       ; JP (HL)
     push hl
     exx
     pop de
     set 7,d
     main

     db 0,0,0,0,0,0,0
ifb_ ei
     b1main

     db 0,0
i04  exx
     inc b
     exx
     b1main

i0d  exx
     dec c
     exx
     b1main

i16  inc de              ; LD D,n
        ld a,(de)
     exx
        ld d,a
        exx
        b1main

     db 0,0,0
i24  exx
     inc h
```

```
     exx
     b1main

i2d     exx
     dec l
     exx
     b1main

i36  inc de
     ld a,(de)
     exx       ; LD (HL),n
     set 7,h
     ld (hl),a
        res 7,h
        exx
     b1main

     db 0,0,0
i48  exx       ; LD C,B
     ld c,b
     exx
     b1main


     db 0,0,0,0,0,0,0
i58       exx       ; LD E,B
     ld e,b
     exx
     b1main

     db 0,0,0,0,0,0,0
i68       exx       ; LD L,B
     ld l,b
     exx
     b1main

     db 0,0,0,0,0,0,0
i78     exx
     ld a,b
     exx
     ld c,a
     b1main

     db 0,0,0,0,0,0
i88     ld a,c
     exx
     adc a,b
     exx
     ld c,a
     b1main

i93     ld a,c
     exx
```

```
     sub e
     exx
     ld c,a
     b1main

     db 0,0,0
ia1  ld a,c
     exx
     and c
     exx
     ld c,a
     b1main

iac  ld a,c
     exx
     xor h
     exx
     ld c,a
     b1main

     db 0
ib8     ld a,c
     exx
     cp b
     exx
     b1main

     db 0
ic3  ex de,hl  ; JP NN
     inc hl
     ld e,(hl)
     inc hl
     ld d,(hl)
     set 7,d
     main

     db 0,0,0,0
id3  inc de         ; OUT (N),A as nop
     b1main

     db 0,0
idc  jp c,icd
     inc de
     inc de
     b1main

ie7  inc de
     push de
     ld de,#8020
     main

hb2  equ $/256*256+256
```

```
     block hb2-256+$f2-$,0
if2  jp p,ic3
     inc de
     inc de
     b1main

     block hb2+$0a-$,0
i0a  exx
     set 7,b
     ld a,(bc)
     res 7,b
     exx
     ld c,a
     b1main

i18  inc de
     ex af,af' ; save F A' is in memory
     ld a,(de)
     ld l,a
     add a,a
     sbc a,a
     ld h,a
     add hl,de
     ex af,af' ; get F
     ex de,hl
     b1main

i28  inc de
     jp z,i10tr
     jp i00

     db 0
i30  inc de
     jp nc,i10tr
     b1main


     db 0,0,0
i3d  dec c          ; DEC A
     b1main



i08a ld hl,0             ; AF' storage
     push hl             ; AF is used for screenrefresh
     ld a,c
     push af             ; no speed up needed
     pop hl              ; but it is emulated
     pop af
     ld (i08a+1),hl
     ld c,a
     b1main
```

```
getreg    ld bc,(bca+1)
     ld de,(dea+1)
     ld hl,(hla+1)
     ret


     block hb2+$7a-$,0
i7a  exx
     ld a,d
     exx
     ld c,a
     b1main

     db 0,0,0,0,0
i89  ld a,c
     exx
     adc a,c
     exx
     ld c,a
     b1main

i94  ld a,c
     exx
     sub h
     exx
     ld c,a
     b1main

     db 0,0,0
ia2  ld a,c
     exx
     and d
     exx
     ld c,a
     b1main

     db 0,0,0,0,0
ib2  ld a,c
     exx
     or d
     exx
     ld c,a
     b1main

ibd  jp ibd2
     db 0,0

ic2  jp nz,ic3
     inc de
     inc de
     b1main

icd  ex de,hl       ; CALL NN
```

```
     inc hl
     ld e,(hl)
     inc hl
     ld d,(hl)
     inc hl
     res 7,h
     push hl
     set 7,d
     main

     db 0,0,0
ie0  jp po,ic9      ; RET PO not often used
     jp i00

ie6  inc de
     ld a,(de)
     and c
     ld c,a
     b1main

     db 0
if1  pop af
     ld c,a
     b1main

hb   equ $/256*256+256

ibd2 ld a,c         ; get A-reg
     exx
     cp l      ; NO SAVE NEEDED WITH CP r
     exx       ; 4 tstates quicker than before
     b1main

     block hb+#1a-$,0
i1a  jp i1ac

     db 0,0
i1f  ld a,c
     rra
     ld c,a
     jp i00

     db 0,0
i27  ld a,c
     daa
     ld c,a
     jp i00


     db 0,0
i2f  ld a,c
     cpl
     ld c,a
```

```
     jp i00

     db 0,0
i37  scf
     jp i00

i3b  jp i3bc

i3e  inc de
     ld a,(de)
i1am ld c,a
     b1main

i0fc ld a,c
     rrca
     ld c,a
     b1main

i2bc exx
     dec hl
     exx
     b1main

i3bc dec sp
     b1main

icbhl   ld (icbh+1),a
     ex af,af'
     ld a,c
     exx
     set 7,h
icbh bit 0,(hl)
     res 7,h
     exx
     ld c,a
     b1main


     block hb+#7c-$
i7c  exx
     ld a,h
     exx
     ld c,a
     b1main

     block hb+#8a-$
i8a  ld a,c
     exx
     adc a,d
     exx
     ld c,a
     b1main
```

```
     block hb+#9a-$
i9a  ld a,c
     exx
     sbc a,d
     exx
     ld c,a
     b1main

     block hb+#aa-$
iaa  ld a,c
     exx
     xor d
     exx
     ld c,a
     b1main

     block hb+#b9-$
ib9  ld a,c
     exx
     cp c
     exx
     b1main

shdel     xor a
     out (254),a    ; border black
     ld d,40        ; delay to show speed up find
shblck    dec a
     ex (sp),hl
     ex (sp),hl
     jr nz,shblck
     dec d
     jr nz,shblck
     ld a,7
     out (254),a
     ret

     block hb+#d6-$
id6  inc de
     ld a,(de)
     ld h,a
     ld a,c
     sub h
     ld c,a
     b1main

     block hb+#e4-$
ie4  jp po,icd
     inc de
     inc de
     b1main

new  ld bc,(#c004)  ; RAMTOP
     ld a,b
```

```
     add a,#80
     ld b,a
     jp newin

fd4ok1    ex de,hl
     ld de,fd4opc+1
     ld a,c
     ld c,b         ; > 4 to keep B unchanged in LDI
     ex af,af' ; LDI effects flags, save F
     ldi
     ldi
     ldi
     ex af,af'
     exx
fd4opc  db #fd,0,0,0
     exx
     ld c,a
     ex de,hl
     main

; After loading some time (>1/2 sec) no intrupt, keep timer
; This is needed to be sure some opcodes coded over sysvar
; are emulated before an intrupt occurs to update timer
; Dr Beep uses this trick to code code over sysvar
; On normal ZX81 this is quick enough, emulated it needs extra
time

isr2 push af
     push hl
keepcnt   ld a,0
     dec a
     and 31         ; > 1/2 sec time to start a game
     ld (keepcnt+1),a
     jr nz,keep
     ld hl,isr1     ; and back to normal intrupt
     ld (isrmod+1),hl
keep pop hl
     pop af
     ei
     ret

; You can check software for fixed delayroutines. Some
; can be sped-up. This piece can do that 3x for a special delay
routine
qdel2     inc a          ; speed up 3x
     jr z,nota0     ; but keep some delay
qdel1     inc a          ; speed up 2x
     jr z,nota0     ; but keep some delay
qdel inc a          ; speed up 1x
     jr nz,nota0+1  ; but keep some delay
nota0     dec a
     add a,(hl)
wfq  cp (hl)
```

```
     jr nz,wfq
     ret

spinn     inc de              ; LD SP,(NN)
     ex de,hl
     ld e,(hl)
     inc hl
     ld d,(hl)
     set 7,d
     ex de,hl
     ld a,(hl)
     inc hl
     ld h,(hl)
     ld l,a
     set 7,h
     ld sp,hl
     b1main

; this piece of code was used for debugging
brktst    ex af,af'
     ld a,e
     cp brkp mod 256
     jr nz,c1
     ld a,d
     cp brkp/256
c1   jp nz,cont
bpnt nop
cont      ex af,af'
     ld a,(de)
     ld l,a
     ld h,b
     ld h,(hl)
     jp (hl)

innsp     inc de              ; LD (NN),SP
     ex de,hl
     ld e,(hl)
     inc hl
     ld d,(hl)
     set 7,d
     ex de,hl
     ld (innsp2+1),hl
     ld hl,#8000
     ex af,af'
     add hl,sp
     ex af,af'
innsp2    ld (0),hl
i76  b1main

; We go back to BASIC when LOADING or when menu is activated
bk2bas    ex af,af'
     push af
     ld (savsp+1),sp
```

```
spret     ld sp,0             ; Get SP which started the emulator
     exx
     ld hl,10072         ; HL' for ZX Spectrum
     exx
     ld iy,#5c3a         ; IY for ZX Spectrum
     ld a,#3f
     ld i,a
     im 1           ; IM mode for ZX Spectrum
     ei
     ret

emulwrm di               ; BACK IN after LOADING or MENU-use
     ld iy,#c000         ; IY for ZX81
     call clrscr         ; Clear ZX Spectrum screen and ZX81-shadow
                    ; to built a full new screen later

savsp     ld sp,0             ; get SP from before step to BASIC
     pop af
     ex af,af'
     ld a,#82
     ld i,a
     im 2           ; IM-mode for ZX81 needed
     ld b,8              ; Set B here, elsewhere no room
     ret

clrscr    ld hl,#bd00         ; start of shadowscreen ZX81
     ld a,#c0       ; impossible value and endmarker
erassc    ld (hl),a      ; ALL BYTES GET IMPOSSIBLE VALUE
     inc hl              ; WHICH WILL REDRAW A FULL SCREEN
     cp h
     jr nz,erassc

     ld a,#58
     ld h,#40
cl2  ld (hl),0      ; We clear ZX Spectrum screen
     inc hl
     cp h           ; until ATTR-reached
     jr nz,cl2
     ret

; Another translated piece of ROM, filling gaps in the code
rest18    ld a,c
     exx
     call L0018
     exx
     ld c,a
     jp ic9

rest20    ld a,c
     exx
     call L0020
     exx
     ld c,a
```

```
     jp ic9


;; GET-CHAR
L0018:  LD      HL,($C016)      ; set HL to character address
CH_ADD.
     set 7,h
        LD      A,(HL)          ; fetch addressed character to A.
     res 7,h

;; TEST-SP
L001C:  AND     A               ; test for space.
        RET     NZ              ; return if not a space

; -----------------------------------
; THE 'COLLECT NEXT CHARACTER' RESTART
; -----------------------------------
; The character address in incremented and the new addressed
character is
; returned if not a space, or cursor, else the process is
repeated.
hd2  equ $/256*256+256

;; NEXT-CHAR
L0020:  CALL    L0049           ; routine CH-ADD+1 gets next
immediate
                                ; character.
        JR      L001C           ; back to TEST-SP.

;; CH-ADD+1
L0049:  LD      HL,($C016)      ; fetch character address to
CH_ADD.

;; TEMP-PTR1
L004C:  INC     HL              ; address next immediate location.

;; TEMP-PTR2
L004D:  LD      ($C016),HL      ; update system variable CH_ADD.
     set 7,h
        LD      A,(HL)          ; fetch the character.
     res 7,h
        CP      $7F             ; compare to cursor character.
        RET     NZ              ; return if not the cursor.
        JR      L004C           ; back for next character to TEMP-
PTR1.


i1ac    exx         ; SLOWER  JP IX
     set 7,d
     ld a,(de)
     res 7,d
     exx
     ld c,a
```

```
     b1main

; No room else to place opcode i06 on place "..06"
     block hd2+6-$,0
i06  inc de         ; LD B,n
        ld a,(de)
     exx
        ld b,a
        exx
        b1main

emulst    ld (spret+1),sp          ; save SP for back to BASIC
     ld sp,0             ; set SP at end of memory

; ENTRY FOR RST 0 is here!!!
romstrt   call clrscr         ; Make sure new screen will be drawn
     ld hl,#9e00         ; ROM characterset
     ld de,29696
     ld bc,512
     ldir           ; Copy first 64 characters
     ld b,2
     xor a
ch64127   ld (de),a
     inc de
     dec c
     jr nz,ch64127       ; Now make 64 characters as space
     djnz ch64127        ; Needed for "executable opcodes" on
screen
     ld hl,#9e00         ; Again ROM-character pointer
     ld b,2
invchar   ld a,(hl)
     cpl
     ld (de),a      ; Invert 64 characters for inverted display
     inc hl
     inc de
     dec bc
     ld a,b
     or c
     jr nz,invchar

     ld a,#82
     ld i,a
;    ld bc,#0000         BC already 0

; Translated ROM for startup
newin     ld hl,cdflag   ; BC is preloaded with (RAMTOP)-1
     res 7,(hl)
     dec bc
     di

     ld h,b
     ld l,c
     ld a,#3f+#80
```

```
l03cf     ld (hl),2
     dec hl
     cp h
     jr nz,l03cf
     res 7,b
l03d5     and a
     res 7,h
     sbc hl,bc
     add hl,bc
     ld a,h
     set 7,h
     jr nc,l03e2
     inc hl
     dec (hl)
     jr z,l03e2-1
     dec (hl)
     jr z,l03d5
     dec hl
l03e2     ld h,a
     inc hl
     ld (#c004),hl       ; RAMTOP

     exx
     ld c,a
     ld b,tab1
     ld de,#83e5         ; Continue emulated
     im 2           ; set IM 2
     jp i00+1



;; EDIT-INP
L046F:  CALL    L14AD           ; routine CURSOR-IN sets cursor
only edit line.

; ->

;; LOWER
L0472:  LD      HL,($C014)      ; fetch edit line start from
E_LINE.

;; EACH-CHAR
L0475:  set 7,h
     LD      A,(HL)          ; fetch a character from edit line.
     res 7,h
        CP      $7E             ; compare to the number marker.
        JR      NZ,L0482        ; forward if not to END-LINE

        LD      BC,$0006        ; else six invisible bytes to be
removed.
        CALL    L0A60           ; routine RECLAIM-2
        JR      L0475           ; back to EACH-CHAR
```

```
; ---

;; END-LINE
L0482:  CP      $76             ;
        INC     HL              ;
        JR      NZ,L0475        ; to EACH-CHAR

;; EDIT-LINE
L0487:  call emulnow
     CALL    $0537           ; routine CURSOR sets cursor K or L.
     db #ed,#1b

;; EDIT-ROOM
L048A:  CALL    L0A1F           ; routine LINE-ENDS
     LD      HL,($C014)      ; sv E_LINE_lo
        LD      (IY+$00),$FF    ; sv ERRNR
     call      emulnow
        CALL    $0766           ; routine COPY-LINE LEAVE EMULATED
FOR NOW
        BIT     7,(IY+$00)      ; sv ERR_NR
        JP      NZ,$04C1        ; to DISPLAY-6

        LD      A,($4022)       ; sv DF_SZ
        CP      $18             ;
        JP      NC,$04C1        ; to DISPLAY-6
     db   #ed,#1b

        INC     A               ;
        LD      ($C022),A       ; sv DF_SZ
        LD      B,A             ;
        LD      C,$01           ;
        CALL    L0918           ; routine LOC-ADDR
        LD      D,H             ;
        LD      E,L             ;
     set 7,h
        LD      A,(HL)          ;

;; FREE-LINE
L04B1:  DEC     HL              ;
     set 7,h
        CP      (HL)            ;
     res 7,h
        JR      NZ,L04B1        ; to FREE-LINE

        INC     HL              ;
        EX      DE,HL           ;
        LD      A,($C005)       ; sv RAMTOP_hi
        CP      $4D             ;
        CALL    C,L0A5D         ; routine RECLAIM-1
        JR      L048A           ; to EDIT-ROOM


nxtl ld a,c
```

```
     exx

;; NEXT-LINE
L066C:  LD      ($c029),HL      ; sv NXTLIN_lo
        EX      DE,HL           ;
        CALL    L004D           ; routine TEMP-PTR-2
     call emulnow
        CALL    $0CC1           ; routine LINE-RUN keeps crashing,
emulated goes OK
     db   #ed,#1b
L0676:    RES     1,(IY+$01)      ; sv FLAGS  - Signal printer not
in use
        LD      A,$C0           ;
        LD      (IY+$19),A      ; sv X_PTR_lo
        CALL    L14A3           ; routine X-TEMP
        RES     5,(IY+$2D)      ; sv FLAGX
        BIT     7,(IY+$00)      ; sv ERR_NR
        JR      Z,L06AE         ; to STOP-LINE

        LD      HL,($c029)      ; sv NXTLIN_lo
     set 7,h
        AND     (HL)            ;
        JR       NZ,L06AE       ; to STOP-LINE

        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
        LD      ($C007),DE      ; sv PPC_lo
        INC     HL              ;
        LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        INC     HL              ;
     res 7,h
        EX      DE,HL           ;
        ADD     HL,DE           ;
        CALL    L0F46           ; routine BREAK-1
        JR      C,L066C         ; to NEXT-LINE

        LD      HL,$C000        ; sv ERR_NR
        BIT     7,(HL)          ;
        JR      Z,L06AE         ; to STOP-LINE

        LD      (HL),$0C        ;

;; STOP-LINE
L06AE:  res 7,h
     BIT     7,(IY+$38)      ; sv PR_CC
     CALL      emulnow
        CALL    Z,$0871         ; routine COPY-BUFF, NOT ADDED!
     db #ed,#1b
        LD      BC,$0121        ;
        CALL    L0918           ; routine LOC-ADDR
```

```
        LD      A,($c000)       ; sv ERR_NR
        LD      BC,($c007)      ; sv PPC_lo
        INC     A               ;
        JR      Z,L06D1         ; to REPORT

        CP      $09             ;
        JR      NZ,L06CA        ; to CONTINUE
        INC     BC              ;

;; CONTINUE
L06CA:  LD      ($C02B),BC      ; sv OLDPPC_lo
        JR      NZ,L06D1        ; to REPORT

        DEC     BC              ;

L06D1:  CALL    L07EB           ; routine OUT-CODE
        LD      A,$18           ;
     call emulnow
        RST     10H             ; PRINT-A
     db #ed,#1b
        CALL    L0A98           ; routine OUT-NUM
        CALL    L14AD           ; routine CURSOR-IN
     call emulnow
        JP      $04C1           ; to DISPLAY-6 emulated is quick
enough


; ---

decode    ld a,c
     exx
     ld d,0
     call #87bf          ; ROM can do decode
     exx
     ld c,a
     jp ic9

L07BD     ld d,0
     JP #87bf       ; ROM can do decode

prsp    ld a,c
     exx
     call L07F5
     exx
     ld b,tab1
     ld c,a
     jp ic9

; -------------------------
; THE 'PRINTING' SUBROUTINE
; -------------------------
;
;
```

```
;; LEAD-SP
L07DC:  LD      A,E             ;
        AND     A               ;
        RET     M               ;

        JR      L07F1           ; to PRINT-CH


;; OUT-DIGIT
L07E1:  XOR     A               ;

;; DIGIT-INC
L07E2:  ADD     HL,BC           ;
        INC     A               ;
        JR      C,L07E2         ; to DIGIT-INC

        SBC     HL,BC           ;
        DEC     A               ;
        JR      Z,L07DC         ; to LEAD-SP

;; OUT-CODE
L07EB:  LD      E,$1C           ;
        ADD     A,E             ;

;; OUT-CH
L07EE:  AND     A               ;
        JR      Z,L07F5         ; to PRINT-SP

;; PRINT-CH
L07F1:  RES     0,(IY+$01)      ; update FLAGS - signal leading
space permitted


;; PRINT-SP

L07F5:  push bc
     push de
     PUSH    HL              ;
        BIT     1,(IY+$01)      ; test FLAGS - is printer in use ?
        JR      NZ,L0802        ; to LPRINT-A

        CALL    L0808           ; routine ENTER-CH
        JR      L0805           ; to PRINT-EXX


hb3  equ $/256*256

ench exx
     call L0809
     exx
     ld c,a
     jp ic9
```

```
     block hb3+$c1-$,0
ic1  exx
     pop bc
     exx
     b1main


; ---

;; LPRINT-A

L0802:  call emulnow
     CALL    #0851           ; routine LPRINT-CH
     db   #ed,#1b

;; PRINT-EXX
L0805:  POP     HL              ;
     pop de
     pop bc
     RET                     ;

; ---

;; ENTER-CH
L0808:  LD      D,A             ;
L0809:    LD      BC,($C039)      ; sv S_POSN_x
        LD      A,C             ;
        CP      $21             ;
        JR      Z,L082C         ; to TEST-LOW

;; TEST-N/L
L0812:  LD      A,$76           ;
        CP      D               ;
        JR      Z,L0847         ; to WRITE-N/L

        LD      HL,($C00E)      ; sv DF_CC_lo
     set 7,h
        CP      (HL)            ;
     res 7,h
        LD      A,D             ;
        JR      NZ,L083E        ; to WRITE-CH

        DEC     C               ;
        JR      NZ,L083A        ; to EXPAND-1

        INC     HL              ;
        LD       ($C00E),HL     ; sv DF_CC_lo
        LD      C,$21           ;
        DEC     B               ;
        LD      ($C039),BC      ; sv S_POSN_x
```

```
;; TEST-LOW
L082C:  LD      A,B             ;
        CP      (IY+$22)        ; sv DF_SZ
        JR      Z,L0835         ; to REPORT-5

        AND     A               ;
        JR      NZ,L0812        ; to TEST-N/L

;; REPORT-5

L0835:  call emulnow
     jp #0835

; ---

;; EXPAND-1
L083A:  CALL    L099B           ; routine ONE-SPACE
        EX      DE,HL           ;



;; WRITE-CH
L083E:  set 7,h
     LD      (HL),A          ;
     res 7,h
        INC     HL              ;
        LD      ($C00E),HL      ; sv DF_CC_lo
        DEC     (IY+$39)        ; sv S_POSN_x
        RET                     ;

;; WRITE-N/L
L0847:  LD      C,$21           ;
        DEC     B               ;
        SET     0,(IY+$01)      ; sv FLAGS  - Suppress leading
space
        JP      L0918           ; to LOC-ADDR



; ----------------------------
; THE 'LOCATE ADDRESS' ROUTINE
; ----------------------------

;; LOC-ADDR
L0918:  LD      ($C039),BC      ; sv S_POSN_x
        LD      HL,($C010)      ; sv VARS_lo
        LD      D,C             ;
        LD      A,$22           ;
        SUB     C               ;
        LD      C,A             ;
        LD      A,$76           ;
        INC     B               ;
```

```
     set 7,h
;; LOOK-BACK
L0927:  DEC     HL              ;
        CP      (HL)            ;
        JR      NZ,L0927        ; to LOOK-BACK

        DJNZ    L0927           ; to LOOK-BACK

        INC     HL              ;
        CPIR                    ;
     res 7,h
        DEC     HL              ;
        LD      ($C00E),HL      ; sv DF_CC_lo
        SCF                     ; Set Carry Flag
        RET     PO              ;

        DEC     D               ;
        RET     Z               ;

        PUSH    BC              ;
        CALL    L099E           ; routine MAKE-ROOM
        POP     BC              ;
        LD      B,C             ;
        LD      H,D             ;
        LD      L,E            ;

;; EXPAND-2
     set 7,h
L0940:  LD      (HL),$00        ;
        DEC     HL              ;
        DJNZ    L0940           ; to EXPAND-2
     res 7,h
        EX      DE,HL           ;
        INC     HL              ;
        LD      ($C00E),HL      ; sv DF_CC_lo
        RET                     ;


mkroom  ld a,c
     exx
     call L099E
     exx
     ld c,a
     jp ic9

;; ONE-SPACE
L099B:  LD      BC,$0001        ;
; --------------------------
; THE 'MAKE ROOM' SUBROUTINE
; --------------------------
L099E:  PUSH    HL              ;
        CALL    L0EC5           ; routine TEST-ROOM
        POP     HL              ;
```

```
        CALL    L09AD           ; routine POINTERS
        LD      HL,($C01C)      ; sv STKEND_lo
        EX      DE,HL           ;
     set 7,h
     set 7,d
        LDDR                    ; Copy Bytes
     res 7,d
     res 7,h
        RET                     ;

; -------------------------
; THE 'POINTERS' SUBROUTINE
; -------------------------
L09AD:  PUSH    AF              ;
        PUSH    HL              ;
        LD      HL,$400C        ; sv D_FILE_lo
        LD      A,$09           ;

;; NEXT-PTR
L09B4:  set 7,h
     LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
     res 7,h
        EX      (SP),HL         ;
        AND     A               ;
        SBC     HL,DE           ;
        ADD     HL,DE           ;
        EX      (SP),HL         ;
        JR      NC,L09C8        ; to PTR-DONE

        PUSH    DE              ;
        EX      DE,HL           ;
        ADD     HL,BC           ;
        EX      DE,HL           ;
     set 7,h
        LD      (HL),D          ;
        DEC     HL              ;
        LD      (HL),E          ;
     res 7,h
        INC     HL              ;
        POP     DE              ;

;; PTR-DONE
L09C8:  INC     HL              ;
        DEC     A               ;
        JR      NZ,L09B4        ; to NEXT-PTR

        EX      DE,HL           ;
        POP     DE              ;
        POP     AF              ;
        AND     A               ;
        SBC     HL,DE           ;
```

```
        LD      B,H             ;
        LD      C,L             ;
        INC     BC              ;
        ADD     HL,DE           ;
        EX      DE,HL           ;
        RET                     ;

;; NEXT-ONE
L09F2:  PUSH    HL              ;
     set 7,h
        LD      A,(HL)          ;
     res 7,h
        CP      $40             ;
        JR      C,L0A0F         ; to LINES

        BIT     5,A             ;
        JR      Z,L0A10         ; forward to NEXT-O-4

        ADD     A,A             ;
        JP      M,L0A01         ; to NEXT+FIVE

        CCF                     ; Complement Carry Flag

;; NEXT+FIVE
L0A01:  LD      BC,$0005        ;
        JR      NC,L0A08        ; to NEXT-LETT

        LD      C,$11           ;

;; NEXT-LETT
L0A08:  RLA                     ;
        INC     HL              ;
     set 7,h
        LD      A,(HL)          ;
     res 7,h
        JR      NC,L0A08        ; to NEXT-LETT

        JR      L0A15           ; to NEXT-ADD

; ---

;; LINES
L0A0F:  INC     HL              ;

;; NEXT-O-4
L0A10:  INC     HL              ;
     set 7,h
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        INC     HL              ;
     res 7,h
;; NEXT-ADD
```

```
L0A15:  ADD     HL,BC           ;
        POP     DE              ;

; --------------------------
; THE 'DIFFERENCE' SUBROUTINE
; --------------------------
;
;

;; DIFFER
L0A17:  AND     A               ;
        SBC     HL,DE           ;
        LD      B,H             ;
        LD      C,L             ;
        ADD     HL,DE           ;
        EX      DE,HL           ;
        RET                     ;

blines    ld a,c
     exx
     call L0A2C
     exx
     ld c,a
     jp ic9

; --------------------------
; THE 'LINE-ENDS' SUBROUTINE
; --------------------------
;
;

;; LINE-ENDS
L0A1F:  LD      B,(IY+$22)      ; sv DF_SZ
        PUSH    BC              ;
        CALL    L0A2C           ; routine B-LINES
        POP     BC              ;
        DEC     B               ;
        JR      L0A2C           ; to B-LINES



;; B-LINES
L0A2C:  RES     1,(IY+$01)      ; sv FLAGS  - Signal printer not
in use
        LD      C,$21           ;
        PUSH    BC              ;
        CALL    L0918           ; routine LOC-ADDR
        POP     BC              ;
        LD      A,($c005)       ; sv RAMTOP_hi
        CP      $4D             ;
        JR      C,L0A52         ; to COLLAPSED
```

```
        SET     7,(IY+$3A)      ; sv S_POSN_y

;; CLEAR-LOC
L0A42:  XOR     A               ; prepare a space
        CALL    L07F5           ; routine PRINT-SP prints a space
        LD      HL,($c039)      ; sv S_POSN_x
        LD      A,L             ;
        OR      H               ;
        AND     $7E             ;
        JR      NZ,L0A42        ; to CLEAR-LOC

        JP      L0918           ; to LOC-ADDR

; ---

;; COLLAPSED
L0A52:  LD      D,H             ;
        LD      E,L             ;
        DEC     HL              ;
        LD      C,B             ;
        LD      B,$00           ;
     set 7,h
     set 7,d
        LDIR                    ; Copy Bytes
     res 7,d
        LD      HL,($c010)      ; sv VARS_lo

; ----------------------------
; THE 'RECLAIMING' SUBROUTINES
; ----------------------------
;
;

;; RECLAIM-1
L0A5D:  CALL    L0A17           ; routine DIFFER

;; RECLAIM-2
L0A60:  PUSH    BC              ;
        LD      A,B             ;
        CPL                     ;
        LD      B,A             ;
        LD      A,C             ;
        CPL                     ;
        LD      C,A             ;
        INC     BC              ;
        CALL    L09AD           ; routine POINTERS
        EX      DE,HL           ;
        POP     HL              ;
        ADD     HL,DE           ;
        PUSH    DE              ;
     set 7,h
     set 7,d
        LDIR                    ; Copy Bytes
```

```
     res 7,d
        POP     HL              ;
        RET                     ;


;; OUT-NUM
L0A98:  PUSH    DE              ;
        PUSH    HL              ;
        XOR     A               ;
        BIT     7,B             ;
        JR      NZ,L0ABF        ; to UNITS

        LD      H,B            ;
        LD      L,C             ;
        LD      E,$FF           ;
        JR      L0AAD           ; to THOUSAND

; ---

;; OUT-NO
L0AA5:  PUSH    DE              ;
     set 7,h
        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
     res 7,h
        PUSH    HL              ;
        EX      DE,HL           ;
        LD      E,$00           ; set E to leading space.

;; THOUSAND
L0AAD:  LD      BC,$FC18        ;
        CALL    L07E1           ; routine OUT-DIGIT
        LD      BC,$FF9C        ;
        CALL    L07E1           ; routine OUT-DIGIT
        LD      C,$F6           ;
        CALL    L07E1           ; routine OUT-DIGIT
        LD      A,L             ;

;; UNITS
L0ABF:  CALL    L07EB           ; routine OUT-CODE
        POP     HL              ;
        POP     DE              ;
        RET                     ;

; ----------------------
; THE 'SEPARATOR' ROUTINE
; ----------------------

;; SEPARATOR
L0D10:  CALL   L0018             ; GET-CHAR
        CP      C               ;
        JR      NZ,L0D26        ; to REPORT-C2
```

```
                                ; 'Nonsense in BASIC'

        JP L0020            ; NEXT-CHAR

; --------------------------
; THE 'CHECK END' SUBROUTINE
; --------------------------
; Check for end of statement and that no spurious characters occur
after
; a correctly parsed statement. Since only one statement is
allowed on each
; line, the only character that may follow a statement is a
NEWLINE.
;

;; CHECK-END
L0D1D:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     NZ              ; return in runtime.

        POP     BC              ; else drop return address.

;; CHECK-2
L0D22:  set 7,h
     LD      A,(HL)          ; fetch character.
     res 7,h
        CP      $76             ; compare to NEWLINE.
        RET     Z               ; return if so.

;; REPORT-C2

L0D26:  call emulnow
     JP      $0D9A           ; to REPORT-C
                                ; 'Nonsense in BASIC'

;; CLASS-END
L0D3A:  PUSH    BC              ;
        RET                     ;


;; REPORT-2
L0D4B:  call emulnow
     jp   #0d4b

testrm    ld a,c
     exx
     call L0EC5
     exx
     ld c,a
     jp ic9


;; SYNTAX-Z
L0DA6:  BIT     7,(IY+$01)      ; test FLAGS  - checking syntax
```

```
only?
     RET

; This routine switches from executed code to emulated code
emulnow exx         ; Save mainregisters
     ld c,a         ; Save A-reg
     pop de         ; Get start of emulation
     ld b,tab1 ; set table
     jp i00+1  ; start emulating

; This routine goes from emulated to executed code
runin     inc de
     push de
     ld a,c
     exx
     ret

;; CLASS-6

L0D92:  call   emulnow
     CALL    #0F55           ; routine SCANNING
     db   #ed,#1b
        BIT     6,(IY+$01)      ; sv FLAGS  - Numeric or string
result?
        RET     NZ              ;

L0D9A     exx
     ld c,a
     ld de,#8d9a
     jp i00+1


; ----------------------------
; THE 'FIND INTEGER' SUBROUTINE
; ----------------------------
;
;

;; FIND-INT
L0EA7:  CALL    L158A           ; routine FP-TO-BC
     JR      C,L0EAD         ; forward with overflow to REPORT-B
        RET     Z               ; return if positive (0-65535).


;; REPORT-B
L0EAD:  call emulnow
     jp #0ead


; -------------------------
; THE 'TEST ROOM' SUBROUTINE
; -------------------------
L0EC5:  LD      HL,($C01C)      ; sv STKEND_lo
```

```
        ADD     HL,BC           ;
        JR      C,L0ED3         ; to REPORT-4

        EX      DE,HL           ;
        LD      HL,$0024        ;
        ADD     HL,DE           ;
     set 7,h
        SBC     HL,SP           ;
     res 7,h
        RET     C               ;

;; REPORT-4
L0ED3:  call emulnow
     jp #0ed3

scankb    ld a,c
     exx
     LD   HL,#FFFF
        call   $82BE           ; routine KEYBOARD, AFTER
TRANSLATION CODE
     exx
     ld c,a
     jp ic9

;; BREAK-1
L0F46:  LD      A,$7F           ; read port $7FFE - keys
B,N,M,.,SPACE.
        IN      A,($FE)         ;
        RRA                     ; carry will be set if space not
pressed.

; -------------------------
; THE 'DEBOUNCE' SUBROUTINE
; -------------------------
;
;

;; DEBOUNCE
L0F4B:  RES     0,(IY+$3B)      ; update system variable CDFLAG
        LD      A,$FF           ;
        LD      ($C027),A       ; update system variable DEBOUNCE
        RET                     ; return.



scann     ld a,c
     exx
     ei

; -------------------------
; THE 'SCANNING' SUBROUTINE
; -------------------------
```

```
; This recursive routine is where the ZX81 gets its power.
Provided there is
; enough memory it can evaluate an expression of unlimited
complexity.
; Note. there is no unary plus so, as on the ZX80, PRINT +1 gives
a syntax error.
; PRINT +1 works on the Spectrum but so too does PRINT + "STRING".

;; SCANNING
L0F55:  call    L0018             ; GET-CHAR
        LD      B,$00           ; set B register to zero.
        PUSH    BC              ; stack zero as a priority end-
marker.

;; S-LOOP-1
L0F59:  CP      $40             ; compare to the 'RND' character
        JR      NZ,L0F8C        ; forward, if not, to S-TEST-PI

; ------------------
; THE 'RND' FUNCTION
; ------------------

        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L0F8A         ; forward if checking syntax to S-
JPI-END

        LD      BC,($c032)      ; sv SEED_lo

     CALL    L1520           ; routine STACK-BC

     call    emulnow
        RST         28H             ;; FP-CALC
        DEFB    $A1             ;;stk-one
        DEFB    $0F             ;;addition
        DEFB    $30             ;;stk-data
        DEFB    $37             ;;Exponent: $87, Bytes: 1
        DEFB    $16             ;;(+00,+00,+00)
        DEFB    $04             ;;multiply
        DEFB    $30             ;;stk-data
        DEFB    $80             ;;Bytes: 3
        DEFB    $41             ;;Exponent $91
        DEFB    $00,$00,$80     ;;(+00)
        DEFB    $2E             ;;n-mod-m
        DEFB    $02             ;;delete
        DEFB    $A1             ;;stk-one
        DEFB    $03             ;;subtract
        DEFB    $2D             ;;duplicate
        DEFB    $34             ;;end-calc
        CALL    #158A           ; routine FP-TO-BC
     db   #ed,#1b        ; continue next as translated

        LD      ($C032),BC      ; update the SEED system variable.
     set 7,h
```

```
        LD      A,(HL)          ; HL addresses the exponent of the
last value.
        AND     A               ; test for zero
        JR      Z,L0F8A-2       ; forward, if so, to S-JPI-END

        SUB     $10             ; else reduce exponent by sixteen
        LD      (HL),A          ; thus dividing by 65536 for last
value.
     res 7,h

;; S-JPI-END
L0F8A:  JR      L0F99           ; forward to S-PI-END

; ---

;; S-TEST-PI
L0F8C:  CP      $42             ; the 'PI' character
        JR      NZ,L0F9D        ; forward, if not, to S-TST-INK

; -------------------
; THE 'PI' EVALUATION
; -------------------

        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L0F99         ; forward if checking syntax to S-
PI-END

     call    emulnow
        RST     28H             ;; FP-CALC
        DEFB    $A3             ;;stk-pi/2
        DEFB    $34             ;;end-calc
     db #ed,#1b

     set 7,h
        INC     (HL)            ; double the exponent giving PI on
the stack.
     res 7,h

;; S-PI-END
L0F99:  call    L0020             ; NEXT-CHAR advances character
pointer.
        JP      L1083           ; jump forward to S-NUMERIC to set
the flag
                                ; to signal numeric result before
advancing.

; ---

;; S-TST-INK
L0F9D:  CP      $41             ; compare to character 'INKEY$'
        JR      NZ,L0FB2        ; forward, if not, to S-ALPHANUM

; ----------------------
```

```
; THE 'INKEY$' EVALUATION
; -----------------------

     LD   HL,#FFFF
        CALL    $82BE           ; routine KEYBOARD, AFTER
TRANSLATION CODE
        LD      B,H             ;
        LD      C,L             ;
        LD      D,C             ;
        INC     D               ;
        CALL    NZ,L07BD        ; routine DECODE
        LD      A,D             ;
        ADC     A,D             ;
        LD      B,D             ;
        LD      C,A             ;
        EX      DE,HL           ;
        JR      L0FED           ; forward to S-STRING

; ---

;; S-ALPHANUM
L0FB2:  call   emulnow
     CALL    #14D2           ; routine ALPHANUM
     db   #ed,#1b
        JR      C,L1025         ; forward, if alphanumeric to S-
LTR-DGT

        CP      $1B             ; is character a '.' ?
        JP      Z,L1047         ; jump forward if so to S-DECIMAL

        LD      BC,$09D8        ; prepare priority 09, operation
'subtract'
        CP      $16             ; is character unary minus '-' ?
        JR      Z,L1020         ; forward, if so, to S-PUSH-PO

        CP      $10             ; is character a '(' ?
        JR      NZ,L0FD6        ; forward if not to S-QUOTE

        CALL    L0049           ; routine CH-ADD+1 advances
character pointer.

; SCANNING MUST BE STARTED OUT OF EMULATION
     call      emulnow
        CALL    $0F55           ; recursively call routine
SCANNING to
                                ; evaluate the sub-expression.
     db #ed,#1b          ; RUNNING CODE ON


        CP      $11             ; is subsequent character a ')' ?
        JR      NZ,L0FFF        ; forward if not to S-RPT-C
```

```
        CALL    L0049           ; routine CH-ADD+1  advances.
        JR      L0FF8           ; relative jump to S-JP-CONT3 and
then S-CONT3

; ---

; consider a quoted string e.g. PRINT "Hooray!"
; Note. quotes are not allowed within a string.

;; S-QUOTE
L0FD6:  CP      $0B             ; is character a quote (") ?
        JR      NZ,L1002        ; forward, if not, to S-FUNCTION

        CALL    L0049           ; routine CH-ADD+1 advances
        PUSH    HL              ; * save start of string.
        JR      L0FE3           ; forward to S-QUOTE-S

; ---


;; S-Q-AGAIN
L0FE0:  CALL    L0049           ; routine CH-ADD+1

;; S-QUOTE-S
L0FE3:  CP      $0B             ; is character a '"' ?
        JR      NZ,L0FFB        ; forward if not to S-Q-NL

        POP     DE              ; * retrieve start of string
        AND     A               ; prepare to subtract.
        SBC     HL,DE           ; subtract start from current
position.
        LD      B,H             ; transfer this length
        LD      C,L             ; to the BC register pair.


;; S-STRING
L0FED:  LD      HL,$C001        ; address system variable FLAGS
        RES     6,(HL)          ; signal string result
        BIT     7,(HL)          ; test if checking syntax.
     res 7,h
        CALL    NZ,L12C3        ; in run-time routine STK-STO-$
stacks the
                                ; string descriptor - start DE,
length BC.

        call L0020             ; NEXT-CHAR advances pointer.

;; S-J-CONT-3
L0FF8:  JP      L1088           ; jump to S-CONT-3

;; S-Q-NL
L0FFB:  CP      $76             ; compare to NEWLINE
        JR      NZ,L0FE0        ; loop back if not to S-Q-AGAIN
```

```
;; S-RPT-C
L0FFF:  JP      L0D9A           ; to REPORT-C

;; S-FUNCTION
L1002:  SUB     $C4             ; subtract 'CODE' reducing codes
                                ; CODE thru '<>' to range $00 -
$XX
        JR      C,L0FFF         ; back, if less, to S-RPT-C

; test for NOT the last function in character set.

        LD      BC,$04EC        ; prepare priority $04, operation
'not'
        CP      $13             ; compare to 'NOT'  ( - CODE)
        JR      Z,L1020         ; forward, if so, to S-PUSH-PO

        JR      NC,L0FFF        ; back with anything higher to S-
RPT-C

; else is a function 'CODE' thru 'CHR$'

        LD      B,$10           ; priority sixteen binds all
functions to
                                ; arguments removing the need for
brackets.

        ADD     A,$D9           ; add $D9 to give range $D9 thru
$EB
                                ; bit 6 is set to show numeric
argument.
                                ; bit 7 is set to show numeric
result.

; now adjust these default argument/result indicators.

        LD      C,A             ; save code in C

        CP      $DC             ; separate 'CODE', 'VAL', 'LEN'
        JR      NC,L101A        ; skip forward if string operand
to S-NO-TO-$

        RES     6,C             ; signal string operand.

;; S-NO-TO-$
L101A:  CP      $EA             ; isolate top of range 'STR$' and
'CHR$'
        JR      C,L1020         ; skip forward with others to S-
PUSH-PO

        RES     7,C             ; signal string result.
```

```
;; S-PUSH-PO
L1020:  PUSH    BC              ; push the priority/operation

        call   L0020             ; NEXT-CHAR
        JP      L0F59           ; jump back to S-LOOP-1


;; S-LTR-DGT
L1025:  exx             ; emulnow
     ld c,a
     ld de,#9025
     jp i00+1

;; S-DECIMAL
L1047:  CALL    L0DA6           ; routine SYNTAX-Z
        JR      NZ,L106F        ; forward in run-time to S-STK-DEC

     call emulnow
        CALL    #14D9           ; routine DEC-TO-FP
     db #ed,#1b

        call   L0018             ; GET-CHAR advances HL past digits
        LD      BC,$0006        ; six locations are required.
        CALL    L099E           ; routine MAKE-ROOM
        INC     HL              ; point to first new location
     set 7,h
        LD      (HL),$7E        ; insert the number marker 126
decimal.
     res 7,h
        INC     HL              ; increment
        EX      DE,HL           ; transfer destination to DE.
        LD      HL,($C01C)      ; set HL from STKEND which points
to the
                                ; first location after the 'last
value'
        LD      C,$05           ; five bytes to move.
        AND     A               ; clear carry.
        SBC     HL,BC           ; subtract five pointing to 'last
value'.
        LD      ($C01C),HL      ; update STKEND thereby 'deleting
the value.

     set 7,h
     set 7,d
        LDIR                    ; copy the five value bytes.
     res 7,h
     res 7,d
        EX      DE,HL           ; basic pointer to HL which may be
white-space
                                ; following the number.
        DEC     HL              ; now points to last of five
bytes.
```

```
        CALL    L004C           ; routine TEMP-PTR1 advances the
character
                                ; address skipping any white-
space.
        JR      L1083           ; forward to S-NUMERIC
                                ; to signal a numeric result.

; ---

; In run-time the branch is here when a digit or point is
encountered.

;; S-STK-DEC
L106F:  call   L0020             ; NEXT-CHAR
        CP      $7E             ; compare to 'number marker'
        JR      NZ,L106F        ; loop back until found to S-STK-
DEC
                                ; skipping all the digits.

        INC     HL              ; point to first of five hidden
bytes.
        LD      DE,($C01C)      ; set destination from STKEND
system variable
        CALL    L19F6           ; routine MOVE-FP stacks the
number.
        LD      ($C01C),DE      ; update system variable STKEND.
        LD      ($C016),HL      ; update system variable CH_ADD.

;; S-NUMERIC
L1083:  SET     6,(IY+$01)      ; update FLAGS  - Signal numeric
result

;; S-CONT-2
L1087:  CALL    L0018             ; GET-CHAR

;; S-CONT-3
L1088:  CP      $10             ; compare to opening bracket '('
        JR      NZ,L1098        ; forward if not to S-OPERTR

        BIT     6,(IY+$01)      ; test FLAGS  - Numeric or string
result?
        JR      NZ,L10BC        ; forward if numeric to S-LOOP

; else is a string
        CALL    L1263           ; routine SLICING

        CALL    L0020             ; NEXT-CHAR
        JR      L1088           ; back to S-CONT-3

; ---

; the character is now manipulated to form an equivalent in the
table of
```

```
; calculator literals. This is quite cumbersome and in the ZX
Spectrum a
; simple look-up table was introduced at this point.

;; S-OPERTR
L1098:  LD      BC,$00C3        ; prepare operator 'subtract' as
default.
                                ; also set B to zero for later
indexing.

        CP      $12             ; is character '>' ?
        JR      C,L10BC         ; forward if less to S-LOOP as
                                ; we have reached end of
meaningful expression

        SUB     $16             ; is character '-' ?
        JR      NC,L10A7        ; forward with - * / and '**' '<>'
to SUBMLTDIV

        ADD     A,$0D           ; increase others by thirteen
                                ; $09 '>' thru $0C '+'
        JR      L10B5           ; forward to GET-PRIO

; ---

;; SUBMLTDIV
L10A7:  CP      $03             ; isolate $00 '-', $01 '*', $02
'/'
        JR      C,L10B5         ; forward if so to GET-PRIO

; else possibly originally $D8 '**' thru $DD '<>' already reduced
by $16

        SUB     $C2             ; giving range $00 to $05
        JR      C,L10BC         ; forward if less to S-LOOP

        CP      $06             ; test the upper limit for
nonsense also
        JR      NC,L10BC        ; forward if so to S-LOOP

        ADD     A,$03           ; increase by 3 to give combined
operators of

;; GET-PRIO
L10B5:  ADD     A,C             ; add to default operation 'sub'
($C3)
        LD      C,A             ; and place in operator byte - C.

        LD      HL,#910F - $C3  ; theoretical base of the
priorities table.
        ADD     HL,BC           ; add C ( B is zero)
     set 7,h
        LD      B,(HL)          ; pick up the priority in B
```

```
     res 7,h
;; S-LOOP
L10BC:  call emulnow
     jp #10BC


;; REPORT-3
L1231:  call emulnow
     jp #1231

; ------------------------
; THE 'SLICING' SUBROUTINE
; ------------------------
;
;

;; SLICING
L1263:  CALL    L0DA6           ; routine SYNTAX-Z
        CALL    NZ,L13F8        ; routine STK-FETCH

        call   L0020            ; NEXT-CHAR
        CP      $11             ; is it ')' ?
        JR      Z,L12BE         ; forward if so to SL-STORE

        PUSH    DE              ;
        XOR     A               ;
        PUSH    AF              ;
        PUSH    BC              ;
        LD      DE,$0001        ;

        call   L0018            ; GET-CHAR
        POP     HL              ;
        CP      $DF             ; is it 'TO' ?
        JR      Z,L1292         ; forward if so to SL-SECOND

        POP     AF              ;
        CALL    L12DE           ; routine INT-EXP2
        PUSH    AF              ;
        LD      D,B             ;
        LD      E,C             ;
        PUSH    HL              ;

        call   L0018            ; GET-CHAR
        POP     HL              ;
        CP      $DF             ; is it 'TO' ?
        JR      Z,L1292         ; forward if so to SL-SECOND

        CP      $11             ;

;; SL-RPT-C
L128B:  JP      NZ,L0D9A        ; to REPORT-C

        LD      H,D             ;
```

```
        LD      L,E             ;
        JR      L12A5           ; forward to SL-DEFINE

; ---

;; SL-SECOND
L1292:  PUSH    HL              ;

        CALL    L0020            ; NEXT-CHAR
        POP     HL              ;
        CP      $11             ; is it ')' ?
        JR      Z,L12A5         ; forward if so to SL-DEFINE

        POP     AF              ;
        CALL    L12DE           ; routine INT-EXP2
        PUSH    AF              ;

        call    L0018            ; GET-CHAR
        LD      H,B             ;
        LD      L,C             ;
        CP      $11             ; is it ')' ?
        JR      NZ,L128B        ; back if not to SL-RPT-C

;; SL-DEFINE
L12A5:  POP     AF              ;
        EX      (SP),HL         ;
        ADD     HL,DE           ;
        DEC     HL              ;
        EX      (SP),HL         ;
        AND     A               ;
        SBC     HL,DE           ;
        LD      BC,$0000        ;
        JR      C,L12B9         ; forward to SL-OVER

        INC     HL              ;
        AND     A               ;
        JP      M,L1231         ; jump back to REPORT-3

        LD      B,H             ;
        LD      C,L             ;

;; SL-OVER
L12B9:  POP     DE              ;
        RES     6,(IY+$01)      ; sv FLAGS  - Signal string result

;; SL-STORE
L12BE:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     Z               ; return if checking syntax.



;; STK-ST-0
L12C2:  XOR     A               ;
```

```
;; STK-STO-$
L12C3:  PUSH    BC              ;
        CALL    L19EB           ; routine TEST-5-SP
        POP     BC              ;
        LD      HL,($C01C)      ; sv STKEND
     set 7,h
        LD      (HL),A          ;
        INC     HL              ;
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;
        INC     HL              ;
        LD      (HL),C          ;
        INC     HL              ;
        LD      (HL),B          ;
        INC     HL              ;
     res 7,h
        LD      ($C01C),HL      ; sv STKEND
        RES     6,(IY+$01)      ; update FLAGS - signal string
result
        RET                     ; return.


; -------------------------
; THE 'INT EXP' SUBROUTINES
; -------------------------
;
;

;; INT-EXP1
L12DD:  XOR     A               ;

;; INT-EXP2
L12DE:  PUSH    DE              ;
        PUSH    HL              ;
        PUSH    AF              ;
        CALL    L0D92           ; routine CLASS-6
        POP     AF              ;
        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L12FC         ; forward if checking syntax to I-
RESTORE

        PUSH    AF              ;
        CALL    L0EA7           ; routine FIND-INT
        POP     DE              ;
        LD      A,B             ;
        OR      C               ;
        SCF                     ; Set Carry Flag
        JR      Z,L12F9         ; forward to I-CARRY

        POP     HL              ;
        PUSH    HL              ;
```

```
        AND     A               ;
        SBC     HL,BC           ;

;; I-CARRY
L12F9:  LD      A,D             ;
        SBC     A,$00           ;

;; I-RESTORE
L12FC:  POP     HL              ;
        POP     DE              ;
        RET                     ;


; --------------------------
; THE 'STK-FETCH' SUBROUTINE
; --------------------------
; This routine fetches a five-byte value from the calculator stack
; reducing the pointer to the end of the stack by five.
; For a floating-point number the exponent is in A and the
mantissa
; is the thirty-two bits EDCB.
; For strings, the start of the string is in DE and the length in
BC.
; A is unused.

;; STK-FETCH
L13F8:  LD      HL,($C01C)      ; load HL from system variable
STKEND

        DEC     HL              ;
     set 7,h
        LD      B,(HL)          ;
        DEC     HL              ;
        LD      C,(HL)          ;
        DEC     HL              ;
        LD      D,(HL)          ;
        DEC     HL              ;
        LD      E,(HL)          ;
        DEC     HL              ;
        LD      A,(HL)          ;
     res 7,h
        LD      ($C01C),HL      ; set system variable STKEND to
lower value.
        RET                     ; return.

;; X-TEMP
L14A3:  LD      HL,($C014)      ; sv E_LINE_lo

; ----------------------
; THE 'SET-STK' ROUTINES
; ----------------------
;
;
```

```
;; SET-STK-B
L14A6:  LD      ($C01A),HL      ; sv STKBOT

;

;; SET-STK-E
L14A9:  LD      ($C01C),HL      ; sv STKEND
        RET                     ;

; -----------------------
; THE 'CURSOR-IN' ROUTINE
; -----------------------
; This routine is called to set the edit line to the minimum
cursor/newline
; and to set STKEND, the start of free space, at the next
position.

;; CURSOR-IN
L14AD:  LD      HL,($C014)      ; fetch start of edit line from
E_LINE
     set 7,h
        LD      (HL),$7F        ; insert cursor character

        INC     HL              ; point to next location.
        LD      (HL),$76        ; insert NEWLINE character
        INC     HL              ; point to next free location.
     res 7,h

        LD      (IY+$22),$02    ; set lower screen display file
size DF_SZ

        JR      L14A6           ; exit via SET-STK-B above

; ------------------------
; THE 'SET-MIN' SUBROUTINE
; ------------------------
;
;

;; SET-MIN
L14BC:  LD      HL,$405D        ; normal location of calculator's
memory area
        LD      ($C01F),HL      ; update system variable MEM
        LD      HL,($C01A)      ; fetch STKBOT
        JR      L14A9           ; back to SET-STK-E



; ------------------------
; THE 'STACK-A' SUBROUTINE
; ------------------------
;
```

```
;; STACK-A
L151D:  LD      C,A             ;
        LD      B,$00           ;

; -------------------------
; THE 'STACK-BC' SUBROUTINE
; -------------------------
; The ZX81 does not have an integer number format so the BC
register contents
; must be converted to their full floating-point form.

;; STACK-BC
L1520:  LD      IY,$C000        ; re-initialize the system
variables pointer.
        PUSH    BC              ; save the integer value.

; now stack zero, five zero bytes as a starting point.

     call    emulnow
        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero                    0.
        DEFB    $34             ;;end-calc
     db #ed,#1b

        POP     BC              ; restore integer value.

     SET 7,H
        LD      (HL),$91        ; place $91 in exponent
65536.
                                ; this is the maximum possible
value

        LD      A,B             ; fetch hi-byte.
        AND     A               ; test for zero.
        JR      NZ,L1536        ; forward if not zero to STK-BC-2

        LD      (HL),A          ; else make exponent zero again
     RES 7,H
        OR      C               ; test lo-byte
        RET     Z               ; return if BC was zero - done.

; else  there has to be a set bit if only the value one.

        LD      B,C             ; save C in B.
     set 7,h
        LD      C,(HL)          ; fetch zero to C
        LD      (HL),$89        ; make exponent $89
256.

;; STK-BC-2
L1536:  DEC     (HL)            ; decrement exponent - halving
```

```
number
        SLA     C               ;  C<-76543210<-0
        RL      B               ;  C<-76543210<-C
        JR      NC,L1536        ; loop back if no carry to STK-BC-
2

        SRL     B               ;  0->76543210->C
        RR      C               ;  C->76543210->C

        INC     HL              ; address first byte of mantissa
        LD      (HL),B          ; insert B
        INC     HL              ; address second byte of mantissa
        LD      (HL),C          ; insert C
     res 7,h
        DEC     HL              ; point to the
        DEC     HL              ; exponent again
        RET                     ; return.

; ----------------------------------------

; -------------------------------------
; THE 'FLOATING-POINT TO BC' SUBROUTINE
; -------------------------------------
; The floating-point form on the calculator stack is compressed
directly into
; the BC register rounding up if necessary.
; Valid range is 0 to 65535.4999

;; FP-TO-BC

L158A:  CALL    L13F8           ; routine STK-FETCH - exponent to
A
                                ; mantissa to EDCB.
        AND     A               ; test for value zero.
        JR      NZ,L1595        ; forward if not to FPBC-NZRO

; else value is zero

        LD      B,A             ; zero to B
        LD      C,A             ; also to C
        PUSH    AF              ; save the flags on machine stack
        JR      L15C6           ; forward to FPBC-END

; ---

; EDCB  =>  BCE

;; FPBC-NZRO
L1595:  LD      B,E             ; transfer the mantissa from EDCB
        LD      E,C             ; to BCE. Bit 7 of E is the 17th
bit which
        LD      C,D             ; will be significant for rounding
if the
```

```
                                ; number is already normalized.

        SUB     $91             ; subtract 65536
        CCF                     ; complement carry flag
        BIT     7,B             ; test sign bit
        PUSH    AF              ; push the result

        SET     7,B             ; set the implied bit
        JR      C,L15C6         ; forward with carry from SUB/CCF
to FPBC-END
                                ; number is too big.

        INC     A               ; increment the exponent and
        NEG                     ; negate to make range $00 - $0F

        CP      $08             ; test if one or two bytes
        JR      C,L15AF         ; forward with two to BIG-INT

        LD      E,C             ; shift mantissa
        LD      C,B             ; 8 places right
        LD      B,$00           ; insert a zero in B
        SUB     $08             ; reduce exponent by eight

;; BIG-INT
L15AF:  AND     A               ; test the exponent
        LD      D,A             ; save exponent in D.

        LD      A,E             ; fractional bits to A
        RLCA                    ; rotate most significant bit to
carry for
                                ; rounding of an already normal
number.

        JR      Z,L15BC         ; forward if exponent zero to EXP-
ZERO
                                ; the number is normalized

;; FPBC-NORM
L15B5:  SRL     B               ;   0->76543210->C
        RR      C               ;   C->76543210->C

        DEC     D               ; decrement exponent

        JR      NZ,L15B5        ; loop back till zero to FPBC-NORM

;; EXP-ZERO
L15BC:  JR      NC,L15C6        ; forward without carry to NO-
ROUND

        INC     BC              ; round up.
        LD      A,B             ; test result
        OR      C               ; for zero
        JR      NZ,L15C6        ; forward if not to GRE-ZERO
```

```
        POP     AF              ; restore sign flag
        SCF                     ; set carry flag to indicate
overflow
        PUSH    AF              ; save combined flags again

;; FPBC-END
L15C6:  PUSH    BC              ; save BC value

; set HL and DE to calculator stack pointers.

     call emulnow
        RST     28H             ;; FP-CALC
        DEFB    $34             ;;end-calc
     db #ed,#1b


        POP     BC              ; restore BC value
        POP     AF              ; restore flags
        LD      A,C             ; copy low byte to A also.
        RET                     ; return


calc4     ld a,c
     call getreg
     exx            ; L19AE
     jp L19AE

calc3     ld a,c
     call getreg
     exx            ; L19A7
     jp L19A7

calc2     ld a,c
     call getreg
     exx            ; L19A0
     jp L19A0

calc1     ld a,c
     call getreg
     exx            ; L199D


;; CALCULATE
; get reg
L199D:  CALL    L1B85           ; routine STK-PNTRS is called to
set up the
                                ; calculator stack pointers for a
default
                                ; unary operation. HL = last value
on stack.
                                ; DE = STKEND first location after
stack.
```

```
; the calculate routine is called at this point by the series
generator...

;; GEN-ENT-1
L19A0:  LD      A,B             ; fetch the Z80 B register to A
        LD      ($C01E),A       ; and store value in system
variable BREG.
                                ; this will be the counter for
dec-jr-nz
                                ; or if used from fp-calc2 the
calculator
                                ; instruction.

; ... and again later at this point

;; GEN-ENT-2
L19A4:  EXX                     ; switch sets
        EX      (SP),HL         ; and store the address of next
instruction,
                                ; the return address, in H'L'.
                                ; If this is a recursive call then
the H'L'
                                ; of the previous invocation goes
on stack.
                                ; c.f. end-calc.
        EXX                     ; switch back to main set.

; this is the re-entry looping point when handling a string of
literals.

;; RE-ENTRY
L19A7:  LD      ($C01C),DE      ; save end of stack in system
variable STKEND
        EXX                     ; switch to alt
     set 7,h
        LD      A,(HL)          ; get next literal
     res 7,h
        INC     HL              ; increase pointer'

; single operation jumps back to here

;; SCAN-ENT
L19AE:  PUSH    HL              ; save pointer on stack   *
        AND     A               ; now test the literal
        JP      P,L19C2         ; forward to FIRST-3D if in range
$00 - $3D
                                ; anything with bit 7 set will be
one of
                                ; 128 compound literals.

; compound literals have the following format.
; bit 7 set indicates compound.
```

```
; bits 6-5 the subgroup 0-3.
; bits 4-0 the embedded parameter $00 - $1F.
; The subgroup 0-3 needs to be manipulated to form the next
available four
; address places after the simple literals in the address table.

        LD      D,A             ; save literal in D
        AND     $60             ; and with 01100000 to isolate
subgroup
        RRCA                    ; rotate bits
        RRCA                    ; 4 places to right
        RRCA                    ; not five as we need offset * 2
        RRCA                    ; 00000xx0
        ADD     A,$72           ; add ($39 * 2) to give correct
offset.
                                ; alter above if you add more
literals.
        LD      L,A             ; store in L for later indexing.
        LD      A,D             ; bring back compound literal
        AND     $1F             ; use mask to isolate parameter
bits
        JR      L19D0           ; forward to ENT-TABLE

; ---

; the branch was here with simple literals.

;; FIRST-3D
L19C2:  CP      $18             ; compare with first unary
operations.
        JR      NC,L19CE        ; to DOUBLE-A with unary
operations

; it is binary so adjust pointers.

        EXX                     ;
        LD      BC,$FFFB        ; the value -5
        LD      D,H             ; transfer HL, the last value, to
DE.
        LD      E,L             ;
        ADD     HL,BC           ; subtract 5 making HL point to
second
                                ; value.
        EXX                     ;

;; DOUBLE-A
L19CE:  RLCA                    ; double the literal
        LD      L,A             ; and store in L for indexing

;; ENT-TABLE
L19D0:  LD      DE,$1923        ; Address: tbl-addrs
        LD      H,$00           ; prepare to index
        ADD     HL,DE           ; add to get address of routine
```

```
     set 7,h
        LD      E,(HL)          ; low byte to E
        INC     HL              ;
        LD      D,(HL)          ; high byte to D
        LD      HL,$19A7        ; Address: RE-ENTRY
        EX      (SP),HL         ; goes on machine stack
                                ; address of next literal goes to
HL. *


        PUSH    DE              ; now the address of routine is
stacked.
        EXX                     ; back to main set
                                ; avoid using IY register.
        LD      BC,($C01D)      ; STKEND_hi
                                ; nothing much goes to C but BREG
to B
                                ; and continue into next ret
instruction
                                ; which has a dual identity
; The RET in the recursion CALCULATOR.
; We keep execution through emulation or it goes wrong.
savreg    exx
     ld (bca+1),bc
     ld (dea+1),de
     ld (hla+1),hl
     ld b,tab1      ; repair B
     ld c,a
     jp ic9              ; use RET to get DE


L19EB:  PUSH    DE              ; save
        PUSH    HL              ; registers

        LD      BC,$0005        ; an overhead of five bytes
        CALL    L0EC5           ; routine TEST-ROOM tests free RAM
raising
                                ; an error if not.
        POP     HL              ; else restore
        POP     DE              ; registers.
        RET                     ; return with BC set at 5.

L19F6:  CALL    L19EB           ; routine TEST-5-SP test free
memory
                                ; and sets BC to 5.
     set 7,h
     set 7,d
        LDIR                    ; copy the five bytes.
     res 7,d
     res 7,h
        RET                     ; return with DE addressing new
STKEND
                                ; and HL addressing new last
```

value.

```
;; STK-PNTRS
L1B85:  LD      HL,($C01C)      ; fetch STKEND value from system
variable.
        LD      DE,$FFFB        ; the value -5
        PUSH    HL              ; push STKEND value.

        ADD     HL,DE           ; subtract 5 from HL.

        POP     DE              ; pop STKEND to DE.
        RET                     ; return.


; In the ROM teh LOAD-routine is altered to come here.
; LOADING goes from BASIC so you can alter to you own special
hardware
loader  di
     ld b,4              ; 1024 and more
     call bk2bas         ; use BASIC as loader
     im 2

     ld hl,(frames)      ; after loading simulate 1 intrupt only
     dec hl
     set 7,h
     ld (frames),hl

     ld hl,isr2          ; skip normal intrupt for >1/2 sec after
loading
     ld (isrmod+1),hl    ; to create time to run special code in
some games
     ld b,tab1
     ld de,#8676         ; start after loading a game
     ei
     jp i00+1

dofd3     ld l,a
     inc de
     ld a,(de)
     ld h,a
     ld (opix3+1),hl
     ld a,c
     exx
opix3   db #fd,0,0
     exx
     ld c,a
     b1main

ddtab     equ $ / 256 +1

     block ddtab *256 -$,0
; from here we use prefix IX followed by a number as extra command
on the ZX81
```

```
; prefix IX on a ZX81 is used for display. Only the unsed
combinations
; Display hires is not possible on t his emulator so I use this to
speed up.
; the extra commands are used to quickrun calculator options
d00  jp  jumptr     ; DD 00 v
d01  jp  exchan     ; DD 03 v
d02  jp  subtrc     ; DD 06 v
     db 0,0,0  ; DD 09 = ADD IX,BC
d04  jp  multpl     ; DD 0C v
d05  jp  divisn     ; DD 0F v
d06     jp  or      ; DD 12 v
d07  jp  nono  ; DD 15 v
d08     jp  noeql   ; DD 18
d09  jp  testnz     ; DD 1B  speed up routine
d0a  jp  testz ; DD 1E  speed up routine

jumptr  call getreg
     ld a,c
     exx
     call L1C2F     ; dd 00
     jp savreg

exchan  call getreg
     exx
     ex af,af'
     call L1A72     ; dd 03
     jp savreg

subtrc  call getreg
     exx
     ex af,af'
     call L174C     ; dd 06
     jp savreg

multpl  call getreg
     exx
     ex af,af'
     call L17C6     ; dd 0c
     jp savreg

divisn  call getreg
     exx
     ex af,af'
     call L1882     ; dd 0f
     jp savreg

or   ld a,c
     exx
     ex af,af'
     call L1AED     ; dd 12
     exx
     ld c,a
```

```
     jp ic9

nono      ld a,c
     exx
     call L1AF3     ; dd 15
     exx
     ld c,a
     jp ic9

noeql   call getreg
     ld a,c
     exx
     call L1B03     ; dd 18
     jp savreg

testz   ld a,#28
     db 33          ; hide testnz
testnz    ld a,#20
     exx
     push hl
     exx
     pop hl
     push de
     ld de,frames-#8000
     and a          ; reset C
     sbc hl,de ; test frames used
     pop hl
     ld e,a
     dec hl
     jr nz,repair   ; no frames, repair original HL
     dec hl
     dec hl
     ld a,(hl)
     cp #86         ; is it ADD A,(HL)?
     jr z,setqd     ; If so use special repaircode
     dec hl
     ld a,(hl)
     cp #D6         ; is it SUB N?
     inc hl         ; point to N
     jr nz,spup     ; no, repair find

setsp     dec (hl)  ; do the speed up
     jr nz,showsp   ; but
     inc (hl)  ; keep a small delay

showsp    call shdel     ; show speedup had effect

spup inc hl
     inc hl
repair    ld (hl),e
     inc hl
     ld (hl),253
     ex de,hl
```

```
     jp i00

setqd    ld (hl),0      ; keep A
     inc hl
     ld (hl),#cd        ; over CP (HL)
     inc hl
     ld (hl),qdel mod 256    ; over JR
     inc hl
     ld (hl),qdel/256- $80   ; over DISPLACE
     push hl
     call shdel
     pop de
     jp i00

;; PREP-ADD
L16D8:  set 7,h
     LD      A,(HL)          ; fetch exponent.
        LD      (HL),$00        ; make this byte zero to take any
overflow and
                                ; default to positive.
     res 7,h
        AND     A               ; test stored exponent for zero.
        RET     Z               ; return with zero flag set if
number is zero.

        INC     HL              ; point to first byte of mantissa.
     set 7,h
        BIT     7,(HL)          ; test the sign bit.
        SET     7,(HL)          ; set it to its implied state.
     res 7,h
        DEC     HL              ; set pointer to first byte again.

        RET     Z               ; return if bit indicated number
is positive.>>

; if negative then all five bytes are twos complemented starting
at LSB.

        PUSH    BC              ; save B register contents.
        LD      BC,$0005        ; set BC to five.
        ADD     HL,BC           ; point to location after 5th
byte.
        LD      B,C             ; set the B counter to five.
        LD      C,A             ; store original exponent in C.
        SCF                     ; set carry flag so that one is
added.

; now enter a loop to twos-complement the number.
; The first of the five bytes becomes $FF to denote a negative
number.

;; NEG-BYTE
L16EC:  DEC     HL              ; point to first or more
```

```
significant byte.
     set 7,h
        LD      A,(HL)          ; fetch to accumulator.
        CPL                     ; complement.
        ADC     A,$00           ; add in initial carry or any
subsequent carry.
        LD      (HL),A          ; place number back.
     res 7,h
        DJNZ    L16EC           ; loop back five times to NEG-BYTE

        LD      A,C             ; restore the exponent to
accumulator.
        POP     BC              ; restore B register contents.

        RET                     ; return.

;; FETCH-TWO
L16F7:  PUSH    HL              ; save HL
        PUSH    AF              ; save A - result sign when used
from division.

     set 7,h
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        LD      (HL),A          ; insert sign when used from
multiplication.
        INC     HL              ;
        LD      A,C             ; m1
        LD      C,(HL)          ;
        PUSH    BC              ; PUSH m2 m3

        INC     HL              ;
        LD      C,(HL)          ; m4
        INC     HL              ;
        LD      B,(HL)          ; m5  BC holds m5 m4
     res 7,h
        EX      DE,HL           ; make HL point to start of second
number.

        LD      D,A             ; m1
     set 7,h
        LD      E,(HL)          ;
        PUSH    DE              ; PUSH m1 n1

        INC     HL              ;
        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
     res 7,h
        PUSH    DE              ; PUSH n2 n3

        EXX                     ; - - - - - - -
```

```
        POP     DE              ; POP n2 n3
        POP     HL              ; POP m1 n1
        POP     BC              ; POP m2 m3

        EXX                     ; - - - - - - -

        INC     HL              ;
     set 7,h
        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ; DE holds n4 n5

        POP     AF              ; restore saved
        POP     HL              ; registers.
        RET                     ; return.

;; SHIFT-FP
L171A:  AND     A               ; test difference between
exponents.
        RET     Z               ; return if zero. both normal.

        CP      $21             ; compare with 33 bits.
        JR      NC,L1736        ; forward if greater than 32 to
ADDEND-0

        PUSH    BC              ; preserve BC - part
        LD      B,A             ; shift counter to B.

; Now perform B right shifts on the addend  L'D'E'D E
; to bring it into line with the augend     H'B'C'C B

;; ONE-SHIFT
L1722:  EXX                     ; - - -
        SRA     L               ;    76543210->C    bit 7
unchanged.
        RR      D               ; C->76543210->C
        RR      E               ; C->76543210->C
        EXX                     ; - - -
        RR      D               ; C->76543210->C
        RR      E               ; C->76543210->C
        DJNZ    L1722           ; loop back B times to ONE-SHIFT

        POP     BC              ; restore BC
        RET     NC              ; return if last shift produced no
carry.   >>

; if carry flag was set then accuracy is being lost so round up
the addend.

        CALL    L1741           ; routine ADD-BACK
        RET     NZ              ; return if not FF 00 00 00 00
```

```
; this branch makes all five bytes of the addend zero and is made
during
; addition when the exponents are too far apart for the addend
bits to
; affect the result.

;; ADDEND-0
L1736:  EXX                     ; select alternate set for more
significant
                                ; bytes.
        XOR     A               ; clear accumulator.


; this entry point (from multiplication) sets four of the bytes to
zero or if
; continuing from above, during addition, then all five bytes are
set to zero.

;; ZEROS-4/5
L1738:  LD      L,$00           ; set byte 1 to zero.
        LD      D,A             ; set byte 2 to A.
        LD      E,L             ; set byte 3 to zero.
        EXX                     ; select main set
        LD      DE,$0000        ; set lower bytes 4 and 5 to zero.
        RET                     ; return.

;; ADD-BACK
L1741:  INC     E               ;
        RET     NZ              ;

        INC     D               ;
        RET     NZ              ;

        EXX                     ;
        INC     E               ;
        JR      NZ,L174A        ; forward if no overflow to ALL-
ADDED

        INC     D               ;

;; ALL-ADDED
L174A:  EXX                     ;
        RET                     ; return with zero flag set for
zero mantissa.

;; subtract
L174C:  set 7,d
     LD      A,(DE)          ; fetch exponent byte of second number
the
                                ; subtrahend.
        AND     A               ; test for zero
     res 7,d
        RET     Z               ; return if zero - first number is
```

```
result.

        INC     DE              ; address the first mantissa byte.
     set 7,d
        LD      A,(DE)          ; fetch to accumulator.
        XOR     $80             ; toggle the sign bit.
        LD      (DE),A          ; place back on calculator stack.
     res 7,d
        DEC     DE              ; point to exponent byte.
                                ; continue into addition routine.

; ------------------------
; THE 'ADDITION' OPERATION
; ------------------------
; The addition operation pulls out all the stops and uses most of
the Z80's
; registers to add two floating-point numbers.
; This is a binary operation and on entry, HL points to the first
number
; and DE to the second.

;; addition
L1755:  EXX                     ; - - -
        PUSH    HL              ; save the pointer to the next
literal.
        EXX                     ; - - -

        PUSH    DE              ; save pointer to second number
        PUSH    HL              ; save pointer to first number -
will be the
                                ; result pointer on calculator
stack.

        CALL    L16D8           ; routine PREP-ADD
        LD      B,A             ; save first exponent byte in B.
        EX      DE,HL           ; switch number pointers.
        CALL    L16D8           ; routine PREP-ADD
        LD      C,A             ; save second exponent byte in C.
        CP      B               ; compare the exponent bytes.
        JR      NC,L1769        ; forward if second higher to
SHIFT-LEN

        LD      A,B             ; else higher exponent to A
        LD      B,C             ; lower exponent to B
        EX      DE,HL           ; switch the number pointers.

;; SHIFT-LEN
L1769:  PUSH    AF              ; save higher exponent
        SUB     B               ; subtract lower exponent

        CALL    L16F7           ; routine FETCH-TWO
        CALL    L171A           ; routine SHIFT-FP
```

```
        POP     AF              ; restore higher exponent.
        POP     HL              ; restore result pointer.
     set 7,h
        LD      (HL),A          ; insert exponent byte.
     res 7,h
        PUSH    HL              ; save result pointer again.

; now perform the 32-bit addition using two 16-bit Z80 add
instructions.

        LD      L,B             ; transfer low bytes of mantissa
individually
        LD      H,C             ; to HL register

        ADD     HL,DE           ; the actual binary addition of
lower bytes

; now the two higher byte pairs that are in the alternate register
sets.

        EXX                     ; switch in set
        EX      DE,HL           ; transfer high mantissa bytes to
HL register.

        ADC     HL,BC           ; the actual addition of higher
bytes with
                                ; any carry from first stage.

        EX      DE,HL           ; result in DE, sign bytes ($FF or
$00) to HL

; now consider the two sign bytes

        LD      A,H             ; fetch sign byte of num1

        ADC     A,L             ; add including any carry from
mantissa
                                ; addition. 00 or 01 or FE or FF

        LD      L,A             ; result in L.

        RRA                     ; C->76543210->C
        XOR     L               ; set bit 0 if shifting required.

        EXX                     ; switch back to main set
        EX      DE,HL           ; full mantissa result now in
D'E'D E registers.
        POP     HL              ; restore pointer to result
exponent on
                                ; the calculator stack.

        RRA                     ; has overflow occurred ?
        JR      NC,L1790        ; skip forward if not to TEST-NEG
```

```
; if the addition of two positive mantissas produced overflow or
if the
; addition of two negative mantissas did not then the result
exponent has to
; be incremented and the mantissa shifted one place to the right.

        LD      A,$01           ; one shift required.
        CALL    L171A           ; routine SHIFT-FP performs a
single shift
                                ; rounding any lost bit
     set 7,h
        INC     (HL)            ; increment the exponent.
     res 7,h
        JR      Z,L17B3         ; forward to ADD-REP-6 if the
exponent
                                ; wraps round from FF to zero as
number is too
                                ; big for the system.

; at this stage the exponent on the calculator stack is correct.

;; TEST-NEG
L1790:  EXX                     ; switch in the alternate set.
        LD      A,L             ; load result sign to accumulator.
        AND     $80             ; isolate bit 7 from sign byte
setting zero
                                ; flag if positive.
        EXX                     ; back to main set.

        INC     HL              ; point to first byte of mantissa
     set 7,h
        LD      (HL),A          ; insert $00 positive or $80
negative at
     res 7,h
                                ; position on calculator stack.

        DEC     HL              ; point to exponent again.
        JR      Z,L17B9         ; forward if positive to GO-NC-MLT

; a negative number has to be twos-complemented before being
placed on stack.

        LD      A,E             ; fetch lowest (rightmost)
mantissa byte.
        NEG                     ; Negate
        CCF                     ; Complement Carry Flag
        LD      E,A             ; place back in register

        LD      A,D             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        LD      D,A             ;
```

```
        EXX                     ; switch to higher (leftmost) 16
bits.

        LD      A,E             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        LD      E,A             ;

        LD      A,D             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        JR      NC,L17B7        ; forward without overflow to END-
COMPL

; else entire mantissa is now zero.  00 00 00 00

        RRA                     ; set mantissa to 80 00 00 00
        EXX                     ; switch.
     set 7,h
        INC     (HL)            ; increment the exponent.
     res 7,h
;; ADD-REP-6
L17B3:  JP      Z,L1880         ; jump forward if exponent now
zero to REPORT-6
                                ; 'Number too big'

        EXX                     ; switch back to alternate set.

;; END-COMPL
L17B7:  LD      D,A             ; put first byte of mantissa back
in DE.
        EXX                     ; switch to main set.

;; GO-NC-MLT
L17B9:  XOR     A               ; clear carry flag and
                                ; clear accumulator so no extra
bits carried
                                ; forward as occurs in
multiplication.
        JR      L1828           ; forward to common code at TEST-
NORM
                                ; but should go straight to
NORMALIZE.
;; PREP-M/D
L17BC:  SCF                     ; set carry flag to signal number
is zero.
     set 7,h
        DEC     (HL)            ; test exponent
        INC     (HL)            ; for zero.
     res 7,h
        RET     Z               ; return if zero with carry flag
set.
```

```
        INC     HL              ; address first mantissa byte.
     set 7,h
        XOR     (HL)            ; exclusive or the running sign
bit.
        SET     7,(HL)          ; set the implied bit.
     res 7,h
        DEC     HL              ; point to exponent byte.
        RET                     ; return.

; ------------------------------
; THE 'MULTIPLICATION' OPERATION
; ------------------------------
;
;

;; multiply
L17C6:  XOR     A               ; reset bit 7 of running sign
flag.
        CALL    L17BC           ; routine PREP-M/D
        RET     C               ; return if number is zero.
                                ; zero * anything = zero.

        EXX                     ; - - -
        PUSH    HL              ; save pointer to 'next literal'
        EXX                     ; - - -

        PUSH    DE              ; save pointer to second number

        EX      DE,HL           ; make HL address second number.

        CALL    L17BC           ; routine PREP-M/D

        EX      DE,HL           ; HL first number, DE - second
number
        JR      C,L1830         ; forward with carry to ZERO-RSLT
                                ; anything * zero = zero.

        PUSH    HL              ; save pointer to first number.

        CALL    L16F7           ; routine FETCH-TWO fetches two
mantissas from
                                ; calc stack to B'C'C,B  D'E'D E
                                ; (HL will be overwritten but the
result sign
                                ; in A is inserted on the
calculator stack)

        LD      A,B             ; transfer low mantissa byte of
first number
        AND     A               ; clear carry.
        SBC     HL,HL           ; a short form of LD HL,$0000 to
take lower
```

```
                                ; two bytes of result. (2 program
bytes)
        EXX                     ; switch in alternate set
        PUSH    HL              ; preserve HL
        SBC     HL,HL           ; set HL to zero also to take
higher two bytes
                                ; of the result and clear carry.
        EXX                     ; switch back.

        LD      B,$21           ; register B can now be used to
count thirty
                                ; three shifts.
        JR      L17F8           ; forward to loop entry point
STRT-MLT

; ---

; The multiplication loop is entered at  STRT-LOOP.

;; MLT-LOOP
L17E7:  JR      NC,L17EE        ; forward if no carry to NO-ADD

                                ; else add in the multiplicand.

        ADD     HL,DE           ; add the two low bytes to result
        EXX                     ; switch to more significant
bytes.
        ADC     HL,DE           ; add high bytes of multiplicand
and any carry.
        EXX                     ; switch to main set.

; in either case shift result right into B'C'C A

;; NO-ADD
L17EE:  EXX                     ; switch to alternate set
        RR      H               ; C > 76543210 > C
        RR      L               ; C > 76543210 > C
        EXX                     ;
        RR      H               ; C > 76543210 > C
        RR      L               ; C > 76543210 > C

;; STRT-MLT
L17F8:  EXX                     ; switch in alternate set.
        RR      B               ; C > 76543210 > C
        RR      C               ; C > 76543210 > C
        EXX                     ; now main set
        RR      C               ; C > 76543210 > C
        RRA                     ; C > 76543210 > C
        DJNZ    L17E7           ; loop back 33 times to MLT-LOOP

;

        EX      DE,HL           ;
```

```
        EXX                     ;
        EX      DE,HL           ;
        EXX                     ;
        POP     BC              ;
        POP     HL              ;
        LD      A,B             ;
        ADD     A,C             ;
        JR      NZ,L180E        ; forward to MAKE-EXPT

        AND     A               ;

;; MAKE-EXPT
L180E:  DEC     A               ;
        CCF                     ; Complement Carry Flag

;; DIVN-EXPT
L1810:  RLA                     ;
        CCF                     ; Complement Carry Flag
        RRA                     ;
        JP      P,L1819         ; forward to OFLW1-CLR

        JP      NC,L1880        ; forward to REPORT-6

        AND     A               ;

;; OFLW1-CLR
L1819:  INC     A               ;
        JR      NZ,L1824        ; forward to OFLW2-CLR

        JR      C,L1824         ; forward to OFLW2-CLR

        EXX                     ;
        BIT     7,D             ;
        EXX                     ;
        JR      NZ,L1880        ; forward to REPORT-6

;; OFLW2-CLR
L1824:  set 7,h
     LD      (HL),A          ;
     res 7,h
        EXX                     ;
        LD      A,B             ;
        EXX                     ;


;; TEST-NORM
L1828:  JR      NC,L183F        ; forward to NORMALIZE

     set 7,h
        LD      A,(HL)          ;
     res 7,h
        AND     A               ;
```

```
;; NEAR-ZERO
L182C:  LD      A,$80           ; prepare to rescue the most
significant bit
                                ; of the mantissa if it is set.
        JR      Z,L1831         ; skip forward to SKIP-ZERO

;; ZERO-RSLT
L1830:  XOR     A               ; make mask byte zero signaling
set five
                                ; bytes to zero.

;; SKIP-ZERO
L1831:  EXX                     ; switch in alternate set
        AND     D               ; isolate most significant bit (if
A is $80).

        CALL    L1738           ; routine ZEROS-4/5 sets mantissa
without
                                ; affecting any flags.

        RLCA                    ; test if MSB set. bit 7 goes to
bit 0.
                                ; either $00 -> $00 or $80 -> $01
     set 7,h
        LD      (HL),A          ; make exponent $01 (lowest) or
$00 zero
     res 7,h
        JR      C,L1868         ; forward if first case to OFLOW-
CLR

        INC     HL              ; address first mantissa byte on
the
                                ; calculator stack.
     set 7,h
        LD      (HL),A          ; insert a zero for the sign bit.
     res 7,h
        DEC     HL              ; point to zero exponent
        JR      L1868           ; forward to OFLOW-CLR

; ---

; this branch is common to addition and multiplication with the
mantissa
; result still in registers D'E'D E .

;; NORMALIZE
L183F:  LD      B,$20           ; a maximum of thirty-two left
shifts will be
                                ; needed.

;; SHIFT-ONE
L1841:  EXX                     ; address higher 16 bits.
        BIT     7,D             ; test the leftmost bit
```

```
        EXX                     ; address lower 16 bits.

        JR      NZ,L1859        ; forward if leftmost bit was set
to NORML-NOW

        RLCA                    ; this holds zero from addition,
33rd bit
                                ; from multiplication.

        RL      E               ; C < 76543210 < C
        RL      D               ; C < 76543210 < C

        EXX                     ; address higher 16 bits.

        RL      E               ; C < 76543210 < C
        RL      D               ; C < 76543210 < C

        EXX                     ; switch to main set.

     set 7,h
        DEC     (HL)            ; decrement the exponent byte on
the calculator
                                ; stack.
     res 7,h
        JR      Z,L182C         ; back if exponent becomes zero to
NEAR-ZERO
                                ; it's just possible that the last
rotation
                                ; set bit 7 of D. We shall see.

        DJNZ    L1841           ; loop back to SHIFT-ONE

; if thirty-two left shifts were performed without setting the
most significant
; bit then the result is zero.

        JR      L1830           ; back to ZERO-RSLT

; ---

;; NORML-NOW
L1859:  RLA                     ; for the addition path, A is
always zero.
                                ; for the mult path, ...

        JR      NC,L1868        ; forward to OFLOW-CLR

; this branch is taken only with multiplication.

        CALL    L1741           ; routine ADD-BACK

        JR      NZ,L1868        ; forward to OFLOW-CLR
```

```
        EXX                     ;
        LD      D,$80           ;
        EXX                     ;
     set 7,h
        INC     (HL)            ;
     res 7,h
        JR      Z,L1880         ; forward to REPORT-6

; now transfer the mantissa from the register sets to the
calculator stack
; incorporating the sign bit already there.

;; OFLOW-CLR
L1868:  PUSH    HL              ; save pointer to exponent on
stack.
        INC     HL              ; address first byte of mantissa
which was
                                ; previously loaded with sign bit
$00 or $80.

        EXX                     ; - - -
        PUSH    DE              ; push the most significant two
bytes.
        EXX                     ; - - -

        POP     BC              ; pop - true mantissa is now BCDE.

; now pick up the sign bit.

        LD      A,B             ; first mantissa byte to A
        RLA                     ; rotate out bit 7 which is set
     set 7,h
        RL      (HL)            ; rotate sign bit on stack into
carry.
        RRA                     ; rotate sign bit into bit 7 of
mantissa.

; and transfer mantissa from main registers to calculator stack.

        LD      (HL),A          ;
        INC     HL              ;
        LD      (HL),C          ;
        INC     HL              ;
        LD      (HL),D          ;
        INC     HL              ;
        LD      (HL),E          ;

        POP     HL              ; restore pointer to num1 now
result.
        POP     DE              ; restore pointer to num2 now
STKEND.

        EXX                     ; - - -
```

```
        POP     HL              ; restore pointer to next
calculator literal.
        EXX                     ; - - -

        RET                     ; return.

;; REPORT-6
L1880:  call emulnow
     RST     08H             ; ERROR-1
        DEFB    $05             ; Error Report: Arithmetic
overflow.

;; division
L1882:  EX      DE,HL           ; consider the second number
first.
        XOR     A               ; set the running sign flag.
        CALL    L17BC           ; routine PREP-M/D
        JR      C,L1880         ; back if zero to REPORT-6
                                ; 'Arithmetic overflow'

        EX      DE,HL           ; now prepare first number and
check for zero.
        CALL    L17BC           ; routine PREP-M/D
        RET     C               ; return if zero, 0/anything is
zero.

        EXX                     ; - - -
        PUSH    HL              ; save pointer to the next
calculator literal.
        EXX                     ; - - -

        PUSH    DE              ; save pointer to divisor - will
be STKEND.
        PUSH    HL              ; save pointer to dividend - will
be result.

        CALL    L16F7           ; routine FETCH-TWO fetches the
two numbers
                                ; into the registers H'B'C'C B
                                ;                    L'D'E'D E
        EXX                     ; - - -
        PUSH    HL              ; save the two exponents.

        LD      H,B             ; transfer the dividend to H'L'H L
        LD      L,C             ;
        EXX                     ;
        LD      H,C             ;
        LD      L,B             ;

        XOR     A               ; clear carry bit and accumulator.
        LD      B,$DF           ; count upwards from -33 decimal
        JR      L18B2           ; forward to mid-loop entry point
DIV-START
```

```
; ---

;; DIV-LOOP
L18A2:  RLA                     ; multiply partial quotient by two
        RL      C               ; setting result bit from carry.
        EXX                     ;
        RL      C               ;
        RL      B               ;
        EXX                     ;

;; div-34th
L18AB:  ADD     HL,HL           ;
        EXX                     ;
        ADC     HL,HL           ;
        EXX                     ;
        JR      C,L18C2         ; forward to SUBN-ONLY

;; DIV-START
L18B2:  SBC     HL,DE           ; subtract divisor part.
        EXX                     ;
        SBC     HL,DE           ;
        EXX                     ;
        JR      NC,L18C9        ; forward if subtraction goes to
NO-RSTORE

        ADD     HL,DE           ; else restore
        EXX                     ;
        ADC     HL,DE           ;
        EXX                     ;
        AND     A               ; clear carry
        JR      L18CA           ; forward to COUNT-ONE

; ---

;; SUBN-ONLY
L18C2:  AND     A               ;
        SBC     HL,DE           ;
        EXX                     ;
        SBC     HL,DE           ;
        EXX                     ;

;; NO-RSTORE
L18C9:  SCF                     ; set carry flag

;; COUNT-ONE
L18CA:  INC     B               ; increment the counter
        JP      M,L18A2         ; back while still minus to DIV-
LOOP

        PUSH    AF              ;
        JR      Z,L18B2         ; back to DIV-START
```

```
; (Note. on the ZX81 this would be JR Z,L18AB)
;
; However if you make this change, then while (1/2=.5) will now
evaluate as
; true, (.25=1/4), which did evaluate as true, no longer does.

        LD      E,A             ;
        LD      D,C             ;
        EXX                     ;
        LD      E,C             ;
        LD      D,B             ;

        POP     AF              ;
        RR      B               ;
        POP     AF              ;
        RR      B               ;

        EXX                     ;
        POP     BC              ;
        POP     HL              ;
        LD      A,B             ;
        SUB     C               ;
        JP      L1810           ; jump back to DIVN-EXPT


L1A72   ld b,5
L1A74     set 7,d
     ld a,(de)
     set 7,h
     ld c,(hl)
     ex de,hl
     ld (de),a
     ld (hl),c
     res 7,h
     res 7,d
     inc hl
     inc de
     djnz L1A74
     ex de,hl
     ret

;; greater-0
L1ACE:  set 7,h
     LD      A,(HL)          ; fetch exponent.
     res 7,h
        AND     A               ; test it for zero.
        RET     Z               ; return if so.


        LD      A,$FF           ; prepare XOR mask for sign bit
        JR      L1ADC           ; forward to SIGN-TO-C
                                ; to put sign in carry
                                ; (carry will become set if sign
```

```
is positive)
                                ; and then overwrite location with
1 or 0
                                ; as appropriate.



;; NOT
;; not
L1AD5:  set 7,h
     LD      A,(HL)          ; get exponent byte.
     res 7,h
        NEG                     ; negate - sets carry if non-zero.
        CCF                     ; complement so carry set if zero,
else reset.
        JR      L1AE0           ; forward to FP-0/1.

; -------------------
; Less than zero (32)
; -------------------
; Destructively test if last value on calculator stack is less
than zero.
; Bit 7 of second byte will be set if so.

;; less-0
L1ADB:  XOR     A               ; set xor mask to zero
                                ; (carry will become set if sign
is negative).

; transfer sign of mantissa to Carry Flag.

;; SIGN-TO-C
L1ADC:  INC     HL              ; address 2nd byte.
     set 7,h
        XOR     (HL)            ; bit 7 of HL will be set if
number is negative.
     res 7,h
        DEC     HL              ; address 1st byte again.
        RLCA                    ; rotate bit 7 of A to carry.


; -----------
; Zero or one
; -----------
; This routine places an integer value zero or one at the
addressed location
; of calculator stack or MEM area. The value one is written if
carry is set on
; entry else zero.

;; FP-0/1
L1AE0:  PUSH    HL              ; save pointer to the first byte
        LD      B,$05           ; five bytes to do.
```

```
;; FP-loop
L1AE3:  set 7,h
     LD      (HL),$00        ; insert a zero.
        INC     HL              ;
        DJNZ    L1AE3           ; repeat.

        POP     HL              ;
        RET     NC              ;
     set 7,h
        LD      (HL),$81        ; make value 1
     res 7,h
        RET                     ; return.



;; or
L1AED:  set 7,d
     LD      A,(DE)          ; fetch exponent of second number
     res 7,d
        AND     A               ; test it.
        RET     Z               ; return if zero.

        SCF                     ; set carry flag
        JR      L1AE0           ; back to FP-0/1 to overwrite the
first operand
                                ; with the value 1.

;; no-&-no
L1AF3:  set 7,d
     LD      A,(DE)          ; fetch exponent of second number.
     res 7,d
        AND     A               ; test it.
        RET     NZ              ; return if not zero.

        JR      L1AE0           ; back to FP-0/1 to overwrite the
first operand
                                ; with zero for return value.

;; no-l-eql,etc.
L1B03:  LD      A,B             ; transfer literal to accumulator.
        SUB     $08             ; subtract eight - which is not
useful.

        BIT     2,A             ; isolate '>', '<', '='.

        JR      NZ,L1B0B        ; skip to EX-OR-NOT with these.

        DEC     A               ; else make $00-$02, $08-$0A to
match bits 0-2.

;; EX-OR-NOT
L1B0B:  RRCA                    ; the first RRCA sets carry for a
```

```
swap.
        JR      NC,L1B16        ; forward to NU-OR-STR with other
8 cases

; for the other 4 cases the two values on the calculator stack are
exchanged.

        PUSH    AF              ; save A and carry.
        PUSH    HL              ; save HL - pointer to first
operand.
                                ; (DE points to second operand).

        CALL    L1A72           ; routine exchange swaps the two
values.
                                ; (HL = second operand, DE =
STKEND)

        POP     DE              ; DE = first operand
        EX      DE,HL           ; as we were.
        POP     AF              ; restore A and carry.

; Note. it would be better if the 2nd RRCA preceded the string
test.
; It would save two duplicate bytes and if we also got rid of that
sub 8
; at the beginning we wouldn't have to alter which bit we test.

;; NU-OR-STR
L1B16:  BIT     2,A             ; test if a string comparison.
        JR      NZ,L1B21        ; forward to STRINGS if so.

; continue with numeric comparisons.

        RRCA                    ; 2nd RRCA causes eql/neql to set
carry.
        PUSH    AF              ; save A and carry

        CALL    L174C           ; routine subtract leaves result
on stack.
        JR      L1B54           ; forward to END-TESTS

; ---

;; STRINGS
L1B21:  RRCA                    ; 2nd RRCA causes eql/neql to set
carry.
        PUSH    AF              ; save A and carry.

        CALL    L13F8           ; routine STK-FETCH gets 2nd
string params
        PUSH    DE              ; save start2 *.
        PUSH    BC              ; and the length.
```

```
        CALL    L13F8           ; routine STK-FETCH gets 1st
string
                                ; parameters - start in DE, length
in BC.
        POP     HL              ; restore length of second to HL.

; A loop is now entered to compare, by subtraction, each
corresponding character
; of the strings. For each successful match, the pointers are
incremented and
; the lengths decreased and the branch taken back to here. If both
string
; remainders become null at the same time, then an exact match
exists.

;; BYTE-COMP
L1B2C:  LD      A,H             ; test if the second string
        OR      L               ; is the null string and hold
flags.

        EX      (SP),HL         ; put length2 on stack, bring
start2 to HL *.
        LD      A,B             ; hi byte of length1 to A

        JR      NZ,L1B3D        ; forward to SEC-PLUS if second
not null.

        OR      C               ; test length of first string.

;; SECND-LOW
L1B33:  POP     BC              ; pop the second length off stack.
        JR      Z,L1B3A         ; forward to BOTH-NULL if first
string is also
                                ; of zero length.

; the true condition - first is longer than second (SECND-LESS)

        POP     AF              ; restore carry (set if eql/neql)
        CCF                     ; complement carry flag.
                                ; Note. equality becomes false.
                                ; Inequality is true. By swapping
or applying
                                ; a terminal 'not', all
comparisons have been
                                ; manipulated so that this is
success path.
        JR      L1B50           ; forward to leave via STR-TEST

; ---
; the branch was here with a match

;; BOTH-NULL
L1B3A:  POP     AF              ; restore carry - set for eql/neql
```

```
        JR      L1B50           ; forward to STR-TEST

; ---
; the branch was here when 2nd string not null and low byte of
first is yet
; to be tested.


;; SEC-PLUS
L1B3D:  OR      C               ; test the length of first string.
        JR      Z,L1B4D         ; forward to FRST-LESS if length
is zero.

; both strings have at least one character left.

     set 7,d
     set 7,h
        LD      A,(DE)          ; fetch character of first string.
        SUB     (HL)            ; subtract with that of 2nd
string.
     res 7,h
     res 7,d
        JR      C,L1B4D         ; forward to FRST-LESS if carry
set

        JR      NZ,L1B33        ; back to SECND-LOW and then STR-
TEST
                                ; if not exact match.

        DEC     BC              ; decrease length of 1st string.
        INC     DE              ; increment 1st string pointer.

        INC     HL              ; increment 2nd string pointer.
        EX      (SP),HL         ; swap with length on stack
        DEC     HL              ; decrement 2nd string length
        JR      L1B2C           ; back to BYTE-COMP

; ---
;   the false condition.

;; FRST-LESS
L1B4D:  POP     BC              ; discard length
        POP     AF              ; pop A
        AND     A               ; clear the carry for false
result.

; ---
;   exact match and x$>y$ rejoin here

;; STR-TEST
L1B50:  PUSH    AF              ; save A and carry
     call svreg2
     call emulnow
```

```
        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero      an initial false
value.
        DEFB    $34             ;;end-calc

     db #ed,#1b

     exx
     call getreg
     EXX

;   both numeric and string paths converge here.

;; END-TESTS
L1B54:  POP     AF              ; pop carry  - will be set if
eql/neql
        PUSH    AF              ; save it again.

        CALL    C,L1AD5         ; routine NOT sets true(1) if
equal(0)
                                ; or, for strings, applies true
result.
        CALL    L1ACE           ; greater-0  ??????????


        POP     AF              ; pop A
        RRCA                    ; the third RRCA - test for '<=',
'>=' or '<>'.
        CALL    NC,L1AD5        ; apply a terminal NOT if so.
        RET                     ; return.




L1C23:  EXX                     ;switch in pointer set

;; JUMP-2
L1C24:  set 7,h
     LD      E,(HL)          ; the jump byte 0-127 forward, 128-255
back.
     res 7,h
        XOR     A               ; clear accumulator.
        BIT     7,E             ; test if negative jump
        JR      Z,L1C2B         ; skip, if positive, to JUMP-3.

        CPL                     ; else change to $FF.

;; JUMP-3
L1C2B:  LD      D,A             ; transfer to high byte.
        ADD     HL,DE           ; advance calculator pointer
forward or back.
```

```
        EXX                     ; switch out pointer set.
        RET                     ; return.

; ----------------------------
; THE 'JUMP ON TRUE' SUBROUTINE
; ----------------------------
; (Offset $00; 'jump-true')
;   This enables the calculator to perform conditional relative
jumps
;   dependent on whether the last test gave a true result
;   On the ZX81, the exponent will be zero for zero or else $81
for one.

;; jump-true
L1C2F:  set 7,d
     LD      A,(DE)          ; collect exponent byte
     res 7,d

        AND     A               ; is result 0 or 1 ?
        JR      NZ,L1C23        ; back to JUMP if true (1).

        EXX                     ; else switch in the pointer set.
        INC     HL              ; step past the jump length.
        EXX                     ; switch in the main set.
        RET                     ; return.

svreg2    exx
     ld (bca+1),bc
     ld (dea+1),de
     ld (hla+1),hl
     exx
     ret


; intrupt service routine
dbfreq    equ 10         ; debounce frequency

lastk     equ #8000+16421
debounc equ #8000+16423
dispfrq equ 2

isr1 push af
     push bc
     push de            ; PC, BC can be set
     push hl

     ld de,#4000        ; preset for screen
     ld b,8

     exx
     push bc
     push de
```

```
     push hl

     ld hl,frames        ; quicker than LD HL,(FRAMES) DEC HL SET
7,H LD (FRAMES),HL
     dec (hl)
     jp nz,read15
     inc hl
     dec (hl)
     set 7,(hl)

read15    ld a,%11110111      ; read 1-5
     in a,(254)
     rra
     jp nc,test0         ; 1 pressed, now test 0 pressed

nomenu  xor a            ; read entire keyboard
     in a,(254)
     cpl
     and 31

nokeyhl   ld hl,#ffff
     call nz,#82be       ; find key only when a key is pressed
     jr z,nokey          ; But also when key released before found
     ld a,h
     inc a
     and l
     inc a
     jr nz,setkey        ; Not shift only
     inc h               ; Make Shift only into nokey
nokey     ld  b,h
     ld c,l
setkey    ld (lastk),hl

dbtest  ld a,h
     and l
     inc a
     jr z,dbnk      ; nokey, reset debounce
     sbc hl,bc
dbval     ld a,dbfreq
     sub 1
     ld (dbval+1),a
     sbc a,a
     or h
     or l
dbnk ld hl,cdflag
     res 0,(hl)
     jr  z,enddb         ; nokey, NO DEBOUNCE
     ld a,dbfreq
     ld (dbval+1),a
     ld (hl),#41         ; signal SLOW and keypressed

enddb     bit 6,(hl)     ; test fast mode
     jr z,exit ; skip display in fast
```

```
     ld hl,intcnt
     dec (hl)  ; only screenupdate when counter reaches 0
     jp nz,exit

setfr     ld (hl),dispfrq     ; reset counter that reached 0

; screendisplay
; d end of screen
; e test for NL
; bc zx81 screen
; hl shadow screen

; hl' ROM pointer
; de' zx screen
; c' topline char

scrdisp   ld de,#5876         ; end of screen and end of line
     ld bc,(#c00c)       ; get ZX81 screen
     set 7,b             ; Placed #8000 further
     ld hl,#bd00-1       ; get ZX81 copyscreen, but WITHOUT
linefeed!

     ld a,(bc)      ; get current screen character
     inc bc
NLin cp #e9              ; test shortest screen after a newline
     jp z,exit      ; end of screen, no further changes

fldtst    inc hl
     ld a,(bc)
     cp e           ; compressed screen, a NL can appear
     jp nz,dochar        ; end of line, no further line changes
     xor a               ; preset for a space
     dec bc              ; next test is still NL, compressed
dochar  cp (hl)              ; test current position in shadow
     jp z,same      ; no display for same character needed
     ld (hl),a      ; store changed character
nl2  exx
     ld l,a
     ld h,14             ; we copied full charset
     add hl,hl
     inc h               ; we needed 1/2 so now we add 1
     add hl,hl
     add hl,hl
     ld c,d              ; save topline
     ld a,(hl)      ; get characterdata
     ld (de),a      ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)      ; get characterdata
     ld (de),a      ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
```

```
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen
     inc l               ; next ROM-pointer (never >256)
     inc d               ; next line on screen
     ld a,(hl)       ; get characterdata
     ld (de),a       ; write to screen

     ld d,c              ; set D back to top
     db 62               ; we have EXX, LD A,N 1 tstate quicker
than 2x EXX

same exx
     inc e               ; goto next field on screen
     ld a,e
     and 31              ; test end of line
     exx
     inc bc              ; next screencharacter, except on
compressed NL
     jp nz,fldtst        ; not yet end of line, tested on displayed
screen
     inc bc              ; skip full line NL
     ld a,l
     inc a               ; end of block?
     ld a,(bc)
     jp nz,NLin          ; 1/3 of screen ready?, no, cont.
     exx
     ld a,d              ; get current screenblock
     add a,b             ; calculate next screenblock
     ld d,a              ; store new screenblock
     exx             ; swap to screenregisters
     cp d            ; test end of screen reached
     jp nz,fldtst        ; do next screenblock

exit pop hl
     pop de
     pop bc
     exx
```

```
     pop hl
     pop de
     pop bc
     pop af
     ei
     ret

intcnt    db 8           ; can be changed by menu

test0     ld a,%11101111      ; read 6-0
     in a,(254)
     rra
     jp c,nomenu         ; no 0 pressed
     ld b,2              ; line 512 to 1023 reserved in BASIC
     call bk2bas         ; back 2 basic for menu
     xor a               ; back from menu, signal no key
     ld de,#4000
;    ld b,8              ; placed at bk2bas. Here over shadowscreen
     exx
     jp setkey      ; exit without keypress
; End of code, but now we have 768 bytes of copyscreen
```

Speedup-routine in contended memory:

```
; Machinecode to find delayloops in a game and speed it up

     org 32000
; find speedup code in your game
     ld hl,#c000
floop   ld a,(hl)
     cp #cd         ; is it call?
     jr nz,testcp   ; if not, test others
     inc hl
     inc hl
     ld a,(hl)
     cp #AF         ; is it call #AFxx (only possible from earlier
speedup)
     dec hl
     jr nz,fnext+1  ; if not, then no speedincrease
     ld a,(hl)
     sub 3
     cp #2B         ; only 2 speed up allowed
     jr c,fnext
     ld (hl),a
     jr fnext
```

```
testcp    cp #be          ; is it CP (HL)
     jr nz,fnext    ; no, test next
     inc hl
     ld a,(hl)
     ld b,#1b
     cp #20         ; is it JR NZ
     jr z,jrnzf
     ld b,#1e
     cp #28         ; is it JR Z
     jr nz,fnext+1  ; test if this was CP (HL)
jrnzf     inc hl          ; Both JR (N)Z found
     ld a,(hl)
     dec hl
     cp 256-3  ; test JR (N)Z,-3
     jr nz,fnext+1  ; if not, find next
     ld (hl),#dd    ; write special routine
     inc hl
     ld (hl),b ; to check if test involves frames
fnext     inc hl
     ld a,h
     or a
     jr nz,floop

     ld hl,(#c00c)  ; get start of screen
     set 7,h        ; relocated 32K up.
     ld b,25        ; 25 newlines on a screen
     dec hl
escr    inc hl
     ld a,(hl)
     cp #e9
     ret z          ; already a JP (HL), then nothing to win
     ld a,#76
     cp (hl)
     jr nz,escr
     djnz escr

fend dec hl         ; we are at end of screen,
                    ; but are there unused lines?
     cp (hl)
     jr z,fend

     inc hl         ; skip last NL
     inc hl         ;
     cp (hl)
     ret nz         ; last NL on screen passed
     ld (hl),#e9    ; we have unused lines and
                    ; we can speed screen up
     ret
```

```
     block 32300-$,0
; write CHROMA-characters over ROM-characterset
     ld de,60000
nchar     ld a,(de)      ; get character
     ld l,a              ; set to begin of pointer
     inc a               ; test on end
     ret z
     ld h,14
     add hl,hl
     inc h
     add hl,hl
     add hl,hl           ; ROM-position calculated
     inc de
     ld b,8
wrchar    ld a,(de)      ; read 8 bytes of data
     ld (hl),a
     inc hl
     inc de
     djnz wrchar         ; write CHROMA-data over "ROM"
     jr nchar            ; fetch next character

     block 32400-$,0
; unpack compressed ZX81 keyboardscreen
     ld hl,26300    ; start of compressed screen
     ld de,16384    ; start of visible screen
decomp    ld a,(hl) ; get data from compressed screen
     cp 11          ; is it compresscode
                    (value 11 not used on screen)
     ld b,1         ; preset 1 copy
     jr nz,wr       ; if not compresscode write value
     inc hl         ; go to next field
     ld b,(hl)      ; get repeatnumber
     inc hl         ; go to next field
     ld a,(hl)      ; get displayvalue
wr   ld (de),a      ; write value to screen
     inc de         ; go to next screenfield
     djnz wr        ; repeat all displays
     inc hl         ; go to next datafield
     ld a,d         ; test 2/3 screen reached
     cp 80
     jr c,decomp    ; display full keyboard
     ret

end
```