# CHAPTER 7. THE DOS INTERFACE

## 7.1. BIOS BREAK INTERRUPT

DOS breaks must be setup and restored carefully if we want to move between DOS and LaForth conveniently. LaForth must install its own BIOS and DOS break interrupts; otherwise, you will be kicked back to DOS whenever an error is detected by DOS.

```
; Handle division by zero Interrupt
DIVBY0:    PUSH     AX                       ; Save some registers
           PUSH     DX
           MOV      DX,OFFSET DB0MSG
           MOV      AX,0900h
           STI
           INT      21h                      ; Print a message.
           CLI
           POP      DX                       ; Restore registers
           POP      AX
           IRET                              ; Return
;
DB0MSG     DB       0Dh,0Ah,07,"Division by Zero S"
;
; BREAK:   LJMP     WINIT
;
; BSTOM:   DB       'BAD STORE'
; BELLQ:   DB       7,63                     ; Bell Question Mark
; QRSTR:   DB       13,10,0,7Fh,7Fh,0
;
;
```

TIME (-- lo hi )
Read the time-of-day clock ticker. Returns the double precision "tick" from the IBM clock.

```
           HEADER   EMIT,T
TIME:      XOR      AH,AH
           INT      21h
           PUSH     DX
           PUSH     CX
           NEXT
```

BIOS Break Handler

```
BIOSBK:
           PUSH     AX
           PUSH     BX
           MOV      AX,CS:BRAN1
           SUB      AX,348Bh                 ; Did we break before?
           JNZ      RESTOR
           MOV      AX,00EBh+(100h*(ABNORM-BRAN-2))              ;
           MOV      CS:BRAN1,AX              ; Modify the code *********
RESTOR:
           POP      BX
           POP      AX
           IRET
```

DOS Break Handler

```
DOSBK:
        CLC                             ; Normal Break case
        DB      0CBh                    ; Far Return
```

## 7.2. THE DOS SERVICE CALL

(DOS ( DS DX CX BX AX -- AX' DS' DX' CX' 0 )
or, if error ( DS DX CX BX AX -- AX AX' -1 )

Primitive operator for DOS calls. If the top value rcturned is 0, there were no errors. It the top element is -1, then the requested operation results in an error specified by ax' .

```
        HEADER  SOD(,H
PDOS:   POP     AX
        POP     BX
        POP     CX
        POP     DX
        POP     DS
        PUSH    AX
        INT     21h                     ; The actual Interrupt.
        JC      PDOSC                   ; Check if there was an error
        ADD     SP,2                    ; No error. Drop old AX.
        PUSH    AX
        PUSH    DS
        PUSH    DX
        PUSH    CX
        XOR     AX,AX                   ; No error case - Push a zero.
        PUSH    AX
PDOS1:  MOV     AX,CS                   ; Restore DS from CS
        MOV     DS,AX
        NEXT
PDOSC:  PUSH    AX                      ; Push Error Return Code
        MOV     AX,-1                   ; Error flag
        PUSH    AX
        JMP     PDOS1
```

DOS ( DS DX CX BX AX -- AX' DS' DX' CX' )

Make a DOS interrupt call. The parameters are the values to be placed in the registers for the call. If an error occurs, the routine is aborted, with an error message.

```
        HEADER  SOD,D
DOS:    NEST
        DW      PDOS
        DW      ZBRAN
        DW      DOS1
        DW      CR
        DW      PTYPE
        DB      "DOS Error "
        DB      0
        DW      HPW
        DW      DROP
        DW      SPACE
        DW      HPW
        DW      SPSTO                   ; Abort
        DW      WINIT
DOS1:   DW      UNNEST
```

BYE ( -- )
Exit to Operating System. Leave LaForth and return to the operating system.

```
          HEADER    EYB,B
BYE:      MOV       DX,BBKIV
          MOV       AX,BBKIV+2
          MOV       DS,AX
          MOV       AX,251Bh
          INT       21h                 ; Restore Int 1B vector
          MOV       DX,DB0IV
          MOV       AX,DB0IV+2
          MOV       DS,AX
          MOV       AX,2500h
          INT       21h                 ; Restore Division by Zero Trap
          MOV       AX,4C00h
          INT       21h                 ; Advanced DOS Exit
```

## 7.3. FILE WORDS

OPEN ( seg offset -- handle )
Open a file for reading and writing. The segment and address point to an ASCIIZ string specifying the file. The "handle" is returned.

```
          HEADER    NEPO,O
OPEN:     NEST
          DW        ZERO
          DW        ZERO
          DW        LIT
          DW        3D02h
          DW        DOS
          DW        DROP
          DW        DROP2
          DW        UNNEST
```

READ ( seg addr count handle -- count' )
Read from the file whose handle is specified into the buffer at the segment and address is given.

```
          HEADER    DAER,R
READ:     NEST
          DW        LIT
          DW        3F00h
          DW        DOS
          DW        DROP
          DW        DROP2
          DW        UNNEST
```

TP ( – addr )
Text Pointer Returns the address of the Text Pointer of the current buffer in the buffer segment.

```
          HEADER    PT,T
TP:       LCALL     AT
          DW        TPTR
```

MT ( seg addr -- )
Purge text buffer down to the segment and address which is contained in top.

```
          HEADER    TM,M
MT:       NEST
          DW        TWODUP
```

```
            DW          ZERO
            DW          MROT
            DW          XCSTOR
            DW          TWODUP
            DW          TP
            DW          TWOSTO
            DW          LIT
            DW          LBUF
            DW          TWOSTO
            DW          UNNEST
```

LT ( seg addr -- )

Get address of Last Text Pushes the segment and address of the Last Text entered into the text buffer. It can be used with MT to discard the last text.

```
            HEADER      TL,L
LT:         NEST
            DW          LIT
            DW          LBUF
            DW          TWOAT
            DW          UNNEST
```

BT ( -- seg addr )

Gets the segment and address of the beginning of the text buffers.

```
            HEADER      TB,B
BT:         NEST
            DW          LIT
            DW          BOTB
            DW          TWOAT
            DW          UNNEST
```