

the short list of supported interrupts with descriptions:

INT 10h / AH = 0 - set video mode.

input:

AL = desired video mode.

these video modes are supported:

00h - text mode. 40x25. 16 colors. 8 pages.

03h - text mode. 80x25. 16 colors. 8 pages.

13h - graphical mode. 40x25. 256 colors. 320x200 pixels. 1 page.

example:

```
mov al, 13h
mov ah, 0
int 10h
```

INT 10h / AH = 01h - set text-mode cursor shape.

input:

CH = cursor start line (bits 0-4) and options (bits 5-7).

CL = bottom cursor line (bits 0-4).

when bit 5 of CH is set to **0**, the cursor is visible. when bit 5 is **1**, the cursor is not visible.

```
; hide blinking text cursor:
    mov ch, 32
    mov ah, 1
    int 10h

; show standard blinking text cursor:
    mov ch, 6
    mov cl, 7
    mov ah, 1
    int 10h

; show box-shaped blinking text cursor:
    mov ch, 0
    mov cl, 7
    mov ah, 1
    int 10h

; note: some bioses required CL to be >=7,
; otherwise wrong cursor shapes are displayed.
```

INT 10h / AH = 2 - set cursor position.

input:

DH = row.

DL = column.

BH = page number (0..7).

example:

```
mov dh, 10
mov dl, 20
mov bh, 0
mov ah, 2
int 10h
```

INT 10h / AH = 03h - get cursor position and size.

input:

BH = page number.

return:

DH = row.

DL = column.

CH = cursor start line.

CL = cursor bottom line.

INT 10h / AH = 05h - select active video page.

input:

AL = new page number (0..7).

the activated page is displayed.

INT 10h / AH = 06h - scroll up window.

INT 10h / AH = 07h - scroll down window.

input:

AL = number of lines by which to scroll (00h = clear entire window).

BH = [attribute](#) used to write blank lines at bottom of window.

CH, CL = row, column of window's upper left corner.

DH, DL = row, column of window's lower right corner.

INT 10h / AH = 08h - read character and [attribute](#) at cursor position.

input:

BH = page number.

return:

AH = [attribute](#).

AL = character.

INT 10h / AH = 09h - write character and [attribute](#) at cursor position.

input:

AL = character to display.

BH = page number.

BL = [attribute](#).

CX = number of times to write character.

INT 10h / AH = 0Ah - write character only at cursor position.

input:

AL = character to display.

BH = page number.

CX = number of times to write character.

INT 10h / AH = 0Ch - change color for a single pixel.

input:

AL = pixel color

CX = column.

DX = row.

example:

```
mov al, 13h
mov ah, 0
int 10h      ; set graphics video mode.
mov al, 1100b
mov cx, 10
mov dx, 20
mov ah, 0ch
int 10h      ; set pixel.
```

INT 10h / AH = 0Dh - get color of a single pixel.

input:

CX = column.

DX = row.

output:

AL = pixel color

INT 10h / AH = 0Eh - teletype output.

input:

AL = character to write.

this functions displays a character on the screen, advancing the cursor and scrolling the screen as necessary. the printing is always done to current active page.

example:

```
mov al, 'a'
mov ah, 0eh
int 10h

; note: on specific systems this
; function may not be supported in graphics mode.
```

INT 10h / AH = 13h - write string.

input:

AL = write mode:

bit 0: update cursor after writing;

bit 1: string contains [attributes](#).

BH = page number.

BL = [attribute](#) if string contains only characters (bit 1 of AL is zero).

CX = number of characters in string (attributes are not counted).

DL,DH = column, row at which to start writing.

ES:BP points to string to be printed.

example:

```
mov al, 1
mov bh, 0
mov bl, 0011_1011b
mov cx, msglend - offset msg1 ; calculate message size.
mov dl, 10
mov dh, 7
push cs
pop es
mov bp, offset msg1
mov ah, 13h
int 10h
jmp msglend
msg1 db " hello, world! "
msglend:
```

INT 10h / AX = 1003h - toggle intensity/blinking.

input:

BL = write mode:

0: enable intensive colors.

1: enable blinking (not supported by the emulator and windows command prompt).

BH = 0 (to avoid problems on some adapters).

example:

```
mov ax, 1003h
mov bx, 0
int 10h
```

bit color table:

character attribute is 8 bit value, low 4 bits set fore color, high 4 bits set background color.

note: the emulator and windows command line prompt do not support background blinking, however to make colors look the same in dos and in full screen mode it is required to turn off the background blinking.

HEX	BIN	COLOR
0	0000	black
1	0001	blue
2	0010	green
3	0011	cyan
4	0100	red
5	0101	magenta
6	0110	brown
7	0111	light gray
8	1000	dark gray
9	1001	light blue
A	1010	light green
B	1011	light cyan
C	1100	light red
D	1101	light magenta
E	1110	yellow
F	1111	white

note:

; use this code for compatibility with dos/cmd prompt full screen mode:

```
mov     ax, 1003h
mov     bx, 0    ; disable blinking.
int     10h
```

INT 11h - get BIOS equipment list.

return:

AX = BIOS equipment list word, actually this call returns the contents of the word at 0040h:0010h.

Currently this function can be used to determine the number of installed number of floppy disk drives.

Bit fields for BIOS-detected installed hardware:

bit(s)	Description
15-14	Number of parallel devices.
13	Reserved.
12	Game port installed.
11-9	Number of serial devices.
8	Reserved.
7-6	Number of floppy disk drives (minus 1):
00	single floppy disk;
01	two floppy disks;
10	three floppy disks;
11	four floppy disks.
5-4	Initial video mode:
00	EGA,VGA,PGA, or other with on-board video BIOS;
01	40x25 CGA color.
10	80x25 CGA color (emulator default).
11	80x25 mono text.
3	Reserved.
2	PS/2 mouse is installed.

- 1 Math coprocessor installed.
 - 0 Set when booted from floppy.
-

INT 12h - get memory size.

return:

AX = kilobytes of contiguous memory starting at absolute address 00000h, this call returns the contents of the word at 0040h:0013h.

Floppy drives are emulated using FLOPPY_0(..3) files.

INT 13h / AH = 00h - reset disk system.

INT 13h / AH = 02h - read disk sectors into memory.

INT 13h / AH = 03h - write disk sectors.

input:

AL = number of sectors to read/write (must be nonzero)

CH = cylinder number (0..79).

CL = sector number (1..18).

DH = head number (0..1).

DL = drive number (0..3 , for the emulator it depends on quantity of FLOPPY_ files).

ES:BX points to data buffer.

return:

CF set on error.

CF clear if successful.

AH = status (0 - if successful).

AL = number of sectors transferred.

Note: each sector has **512** bytes.

INT 15h / AH = 86h - BIOS wait function.

input:

CX:DX = interval in microseconds

return:

CF clear if successful (wait interval elapsed),

CF set on error or when wait function is already in progress.

Note:

the resolution of the wait period is 977 microseconds on many systems (1 million microseconds - 1 second).

Windows XP does not support this interrupt (always sets CF=1).

INT 16h / AH = 00h - get keystroke from keyboard (no echo).

return:

AH = BIOS scan code.

AL = ASCII character.

(if a keystroke is present, it is removed from the keyboard buffer).

INT 16h / AH = 01h - check for keystroke in the keyboard buffer.

return:

ZF = 1 if keystroke is not available.

ZF = 0 if keystroke available.

AH = BIOS scan code.

AL = ASCII character.

(if a keystroke is present, it is not removed from the keyboard buffer).

INT 19h - system reboot.

Usually, the BIOS will try to read sector 1, head 0, track 0 from drive **A:** to 0000h:7C00h. The emulator just stops the execution, to boot from floppy drive select from the menu: **'virtual drive' - > 'boot from floppy'**

INT 1Ah / AH = 00h - get system time.

return:

CX:DX = number of clock ticks since midnight.

AL = midnight counter, advanced each time midnight passes.

notes:

there are approximately **18.20648** clock ticks per second,

and **1800B0h** per 24 hours.

AL is not set by the emulator.

INT 20h - exit to operating system.

The short list of emulated MS-DOS interrupts -- INT 21h

DOS file system is emulated in **C:\emu8086\vdribe\x** (x is a drive letter)

If no drive letter is specified and current directory is not set, then **C:\emu8086\MyBuild** path is used by default. **FLOPPY_0,1,2,3** files are emulated independently from DOS file system.

For the emulator physical drive **A:** is this file **c:\emu8086\FLOPPY_0** (for BIOS interrupts: **INT 13h** and boot).

For DOS interrupts (**INT 21h**) drive **A:** is emulated in this subdirectory: **C:\emu8086\vdribe\A**

Note: DOS file system limits the file and directory names to 8 characters, extension is limited to 3 characters; example of a valid file name: **myfile.txt** (file name = 6 chars, extension = 3 chars). extension is written after the dot, no other dots are allowed.

INT 21h / AH=1 - read character from standard input, with echo, result is stored in **AL**.
if there is no character in the keyboard buffer, the function waits until any key is pressed.

example:

```
mov ah, 1
int 21h
```

INT 21h / AH=2 - write character to standard output.
entry: **DL** = character to write, after execution **AL = DL**.

example:

```
mov ah, 2
mov dl, 'a'
int 21h
```

INT 21h / AH=5 - output character to printer.
entry: **DL** = character to print, after execution **AL = DL**.

example:

```
mov ah, 5
mov dl, 'a'
int 21h
```

INT 21h / AH=6 - direct console input or output.

parameters for output: **DL** = 0..254 (ascii code)

parameters for input: **DL** = 255

for output returns: **AL** = **DL**

for input returns: **ZF** set if no character available and **AL = 00h**, **ZF** clear if character available.

AL = character read; buffer is cleared.

example:

```
mov ah, 6
mov dl, 'a'
int 21h          ; output character.

mov ah, 6
mov dl, 255
int 21h          ; get character from keyboard buffer (if any) or set ZF=1.
```

INT 21h / AH=7 - character input without echo to **AL**.

if there is no character in the keyboard buffer, the function waits until any key is pressed.

example: