```
mov ah, 7
int 21h
```

INT 21h / AH=9 - output of a string at **DS:DX**. String must be terminated by '**$**'.

example:

```
org 100h
mov dx, offset msg
mov ah, 9
int 21h
ret
msg db "hello world $"
```

INT 21h / AH=0Ah - input of a string to **DS:DX**, fist byte is buffer size, second byte is number of chars actually read. this function does **not** add '$' in the end of string. to print using **INT 21h / AH=9** you must set dollar character at the end of it and start printing from address **DS:DX + 2**.

example:

```
org 100h
mov dx, offset buffer
mov ah, 0ah
int 21h
jmp print
buffer db 10,?, 10 dup(' ')
print:
xor bx, bx
mov bl, buffer[1]
mov buffer[bx+2], '$'
mov dx, offset buffer + 2
mov ah, 9
int 21h
ret
```

the function does not allow to enter more characters than the specified buffer size.
see also **int21.asm** in c:\emu8086\examples

INT 21h / AH=0Bh - get input status;
returns: **AL = 00h** if no character available, **AL = 0FFh** if character is available.

INT 21h / AH=0Ch - flush keyboard buffer and read standard input.
entry: **AL** = number of input function to execute after flushing buffer (can be 01h,06h,07h,08h, or 0Ah - for other values the buffer is flushed but no input is attempted); other registers as appropriate for the selected input function.

INT 21h / AH= 0Eh - select default drive.

Entry: **DL** = new default drive (0=A:, 1=B:, etc)

Return: **AL** = number of potentially valid drive letters

Notes: the return value is the highest drive present.

INT 21h / AH= 19h - get current default drive.

Return: AL = drive (0=A:, 1=B:, etc)

INT 21h / AH=25h - set interrupt vector;
input: **AL** = interrupt number. **DS:DX** -> new interrupt handler.

INT 21h / AH=2Ah - get system date;
return: **CX** = year (1980-2099). **DH** = month. **DL** = day. **AL** = day of week (00h=Sunday)

INT 21h / AH=2Ch - get system time;
return: **CH** = hour. **CL** = minute. **DH** = second. **DL** = 1/100 seconds.

INT 21h / AH=35h - get interrupt vector;
entry: **AL** = interrupt number;
return: **ES:BX** -> current interrupt handler.

INT 21h / AH= 39h - make directory.
entry: **DS:DX** -> ASCIZ pathname; zero terminated string, for example:

```
org 100h
mov dx, offset filepath
mov ah, 39h
int 21h

ret

filepath DB "C:\mydir", 0       ; path to be created.
end
```

the above code creates **c:\emu8086\vdrive\C\mydir** directory if run by the emulator.

Return: **CF** clear if successful **AX** destroyed. **CF** set on error **AX** = error code.
Note: all directories in the given path must exist except the last one.

INT 21h / AH= 3Ah - remove directory.

Entry: **DS:DX** -> ASCIZ pathname of directory to be removed.

Return:

CF is clear if successful, **AX** destroyed **CF** is set on error **AX** = error code.

Notes: directory must be empty (there should be no files inside of it).

INT 21h / AH= 3Bh - set current directory.

Entry: **DS:DX** -> ASCIZ pathname to become current directory (max 64 bytes).

Return:

Carry Flag is clear if successful, **AX** destroyed.
Carry Flag is set on error **AX** = error code.
Notes: even if new directory name includes a drive letter, the default drive is not changed, only the current directory on that drive.

INT 21h / AH= 3Ch - create or truncate file.

entry:

CX = file attributes:

```
mov cx, 0       ;  normal - no attributes.
mov cx, 1       ;  read-only.
mov cx, 2       ;  hidden.
mov cx, 4       ;  system
mov cx, 7       ;  hidden, system and read-only!
mov cx, 16      ;  archive
```

DS:DX -> ASCIZ filename.

returns:

CF clear if successful, **AX** = file handle.
CF set on error **AX** = error code.

note: if specified file exists it is deleted without a warning.

example:

```
        org 100h
        mov ah, 3ch
        mov cx, 0
        mov dx, offset filename
```

```
mov ah, 3ch
int 21h
jc err
mov handle, ax
jmp k
filename db "myfile.txt", 0
handle dw ?
err:
; ....
k:
ret
```

INT 21h / AH= 3Dh - open existing file.

Entry:

AL = access and sharing modes:

```
mov al, 0    ; read
mov al, 1    ; write
mov al, 2    ; read/write
```

DS:DX -> ASCIZ filename.

Return:

CF clear if successful, **AX** = file handle.
CF set on error **AX** = error code.

note 1: file pointer is set to start of file.
note 2: file must exist.

example:

```
org 100h
mov al, 2
mov dx, offset filename
mov ah, 3dh
int 21h
jc err
mov handle, ax
jmp k
filename db "myfile.txt", 0
handle dw ?
err:
; ....
k:
ret
```

INT 21h / AH= 3Eh - close file.

Entry: **BX** = file handle

Return:

CF clear if successful, **AX** destroyed.
CF set on error, **AX** = error code (06h).

INT 21h / AH= 3Fh - read from file.

Entry:

BX = file handle.
CX = number of bytes to read.
DS:DX -> buffer for data.

Return:

CF is clear if successful - **AX** = number of bytes actually read; 0 if at EOF (end of file) before call.
CF is set on error **AX** = error code.

Note: data is read beginning at current file position, and the file position is updated after a successful read the returned **AX** may be smaller than the request in **CX** if a partial read occurred.

INT 21h / AH= 40h - write to file.

entry:

BX = file handle.
CX = number of bytes to write.
DS:DX -> data to write.

return:

CF clear if successful; **AX** = number of bytes actually written.
CF set on error; **AX** = error code.

note: if **CX** is zero, no data is written, and the file is truncated or extended to the current position data is written beginning at the current file position, and the file position is updated after a successful write the usual cause for **AX** < **CX** on return is a full disk.

INT 21h / AH= 41h - delete file (unlink).

Entry:

DS:DX -> ASCIZ filename (no wildcards, but see notes).

return:

CF clear if successful, **AX** destroyed. **AL** is the drive of deleted file (undocumented).
CF set on error **AX** = error code.

Note: DOS does not erase the file's data; it merely becomes inaccessible because the FAT chain for the file is cleared deleting a file which is currently open may lead to filesystem corruption.

INT 21h / AH= 42h - SEEK - set current file position.

Entry:

AL = origin of move: **0** - start of file. **1** - current file position. **2** - end of file.
BX = file handle.
CX:DX = offset from origin of new file position.

Return:

CF clear if successful, **DX:AX** = new file position in bytes from start of file.
CF set on error, AX = error code.

Notes:

for origins **1** and **2**, the pointer may be positioned before the start of the file; no error is returned in that case, but subsequent attempts to read or write the file will produce errors. If the new position is beyond the current end of file, the file will be extended by the next write (see AH=40h).

example:

```
org 100h
mov ah, 3ch
mov cx, 0
mov dx, offset filename
mov ah, 3ch
int 21h  ; create file...
mov handle, ax

mov bx, handle
mov dx, offset data
mov cx, data_size
mov ah, 40h
int 21h ; write to file...

mov al, 0
mov bx, handle
mov cx, 0
mov dx, 7
mov ah, 42h
int 21h ; seek...

mov bx, handle
mov dx, offset buffer
mov cx, 4
mov ah, 3fh
int 21h ; read from file...

mov bx, handle
mov ah, 3eh
int 21h ; close file...
ret

filename db "myfile.txt", 0
handle dw ?
data db " hello files! "
data_size=$-offset data
buffer db 4 dup(' ')
```

INT 21h / AH= 47h - get current directory.

Entry:

DL = drive number (00h = default, 01h = A:, etc)
DS:SI -> 64-byte buffer for ASCIZ pathname.

Return:

Carry is clear if successful
Carry is set on error, **AX** = error code (0Fh)

Notes:

the returned path does not include a drive and the initial backslash.

INT 21h / AH=4Ch - return control to the operating system (stop program).

INT 21h / AH= 56h - rename file / move file.

Entry:

DS:DX -> ASCIZ filename of existing file.
ES:DI -> ASCIZ new filename.

Return:

CF clear if successful.
CF set on error, **AX** = error code.

Note: allows move between directories on same logical drive only; open files should not be renamed!

---

mouse driver interrupts -- INT 33h

INT 33h / AX=0000 - mouse ininialization. any previous mouse pointer is hidden.

returns:

if successful: **AX**=0FFFFh and **BX**=number of mouse buttons.
if failed: **AX**=0

example:

```
mov ax, 0
int 33h
```

see also: mouse.asm in examples.

INT 33h / AX=0001 - show mouse pointer.

example:

```
mov ax, 1
int 33h
```

INT 33h / AX=0002 - hide visible mouse pointer.

example:

```
mov ax, 2
int 33h
```

INT 33h / AX=0003 - get mouse position and status of its buttons.

returns:

if left button is down: **BX**=1
if right button is down: **BX**=2
if both buttons are down: **BX**=3
CX = x
DX = y

example:

```
mov ax, 3
int 33h

; note: in graphical 320x200 mode the value of CX is doubled.
; see mouse2.asm in examples.
```
