Experiment #6

Using BIOS Services and DOS functions
Part 1: Pixel-based Graphics

6.0 Objectives:

The objective of this experiment is to introduce BIOS and DOS interrupt service routines to write assembly language programs for pixel-based graphics.

In this experiment, you will use BIOS and DOS services to write programs that can do the following:

- Set graphics video mode
- Write a pixel on the screen
- Draw a line on the screen
- Draw a rectangle on the screen

6.1 Introduction:

In text mode, the cursor is always displayed on the screen and the resolution is indicated as number of characters per line and number of lines per screen.
In graphics mode, the cursor will not appear on the screen and the resolution is specified as number of pixels per line and number of lines per screen. Text can be used as usual in graphics mode.

6.1.1 BIOS Video I/O Services

The BIOS function requests in this category are used to control graphics on the PC's display screen. The function request is chosen by setting the AH register to the appropriate value and issuing and interrupt 10H.

<u>Set Video Mode (INT 10H, Function 00H):</u>

Selects the video mode and clears the screen automatically.

Description: (INT 10H, Function 00H)	**Example**
Invoked with: AH = 00H AL = mode number to indicate the desired video mode Returns: Nothing	MOV AH, 00 MOV AL, 12H ; graphics video mode INT 10H

Table: Possible video mode settings.

Mode	Type	Max. Colors	Size	Resolution
00	Text	16	40 x 25	- -
01	Text	16	40 x 25	- -
02	Text	16	80 x 25	- -
03	Text	16	80 x 25	- -
04	Graphics	4	40 x 25	320 x 200
05	Graphics	4	40 x 25	320 x 200
06	Graphics	2	80 x 25	640 x 200
07	Text	Mono	80 x 25	
08	Graphics	16	20 x 25	
09	Graphics	16	40 x 25	
0A	Graphics	4	80 x 25	
0B	- -	-		
0C	- -	-		
0D	Graphics	16	40 x 25	320 x 200
0E	Graphics	16	80 x 25	640 x 200
0F	Graphics	Mono	80 x 25	640 x 350
10	Graphics	16	80 x 25	640 x 350
11	Graphics	2	80 x 25	640 x 480
12	Graphics	16	80 x 25	640 x 480
13	Graphics	256	40 x 25	320 x 200

Get Video Mode (INT 10H, Function 0FH):

Gets the current video mode.

Description: (INT 10H, Function 0FH)	**Example**
Invoked with: AH = 0FH Returns: current mode number in AL	MOV AH, 0FH INT 10H

Scroll the Screen or a Window Up (INT 10H, Function 06H)**:**

Input:

AH = 6

AL = number of lines to scroll (0 => whole screen)

BH = attribute for blank lines

CH, CL = row, column for upper left corner

DH, DL = row, column for lower right window

Returns: Nothing

Scrolling the **screen up one line** means to move each display line UP one row and insert a blank line at the bottom of the screen. The previous top row disappears from the screen.

The whole screen or any rectangular area (window) may be scrolled. AL contains the number of lines to scroll. If AL = 0, all the lines are scrolled and this clears the screen or window.

Example: Clear the screen to black for the 80x25 display.

```
MOV AH, 6          ; scroll up function
XOR AL, AL         ; clear entire screen
XOR CX, CX         ; upper left corner is (0,0)
MOV DX, 184FH      ; lower right corner is (4Fh, 18H)
MOV BH, 7          ; normal video attribute
INT 10H            ; clear screen
```

Scroll the Screen/Window down (INT 10H, Function 07H)**:**

Input:

AH = 7

AL = number of lines to scroll (0 => whole screen)

BH = attribute for blank lines

CH, CL = row, column for upper left corner

DH, DL = row, column for lower right corner

Returns: Nothing

Same as function 6, but lines are scrolled down instead of up.

16-Color Display

Attribute Byte:

Bit#	7	6	5	4	3	2	1	0
Attr	BL	R	G	B	IN	R	G	B

Attributes:

Bit #	Attribute
0-2	character color (**foreground color**)
3	intensity
4-6	**background** color
7	blinking

E.g., to display a red character on a blue background, the attribute byte would be:

0001 0100 = 14h

If the attribute byte is: 0011 0101 = 35h

Uses blue + green (cyan) in the background and red + blue (magenta) in the foreground, so the character displayed would be magenta on a cyan background.

If the ***intensity bit*** **(bit 3)** is 1, the foreground color is lightened (brightened). If the ***blinking bit*** **(bit 7)** is 1, the character turns on and off.

Write Pixel (INT 10h Function 0Ch):

Draws the smallest unit of graphics display, also called a dot, a point or a pixel (picture element) on the display at specified graphics coordinates. This function operates only in graphics modes.

Input

AH = 0Ch

AL = pixel value

(if bit 7 is 1, the new pixel color bits will be EX-ORed with the color bits of the current pixel.

BH = video display page

CX = column (graphics x coordinate)

DX = row (graphics y coordinate)

Returns: Nothing

6.2 Pre-lab:

1. Drawing a Pixel

The following program draws a pixel on the screen at location (320, 240) using the “write pixel” function (AH=0Ch) of INT 10h. Run the program after assembling and linking it.

```
TITLE  "Program to draw a pixel on the screen"
.MODEL SMALL                    ; this defines the memory model
.STACK 100                      ; define a stack segment of 100 bytes
.DATA                           ; this is the data segment
.CODE                           ; this is the code segment

        MOV AX,@DATA            ; get the address of the data segment
        MOV DS, AX              ; and store it in DS register

        MOV AH, 0Fh             ; get current video mode
        INT 10h
        PUSH AX                 ; save current video mode

        MOV AH, 00h             ; set video mode
        MOV AL, 12h             ; graphics 640x480
        INT 10h

        ; draw a green color pixel at location (320, 240)
        MOV AH, 0Ch             ; Function 0Ch: Write pixel dot
        MOV AL, 02              ; specify green color
        MOV CX, 320             ; column 320
        MOV DX, 240             ; row 240
        MOV BH, 0               ; page 0
        INT 10h

        MOV AH, 07h             ; wait for key press to exit program
        INT 21h

        POP AX                  ; retrieve original video mode
        MOV AH, 00h
        INT 10h                 ; restore original video mode

        MOV AX, 4C00H           ; Exit to DOS function
        INT 21H

END                             ; end of the program
```

2. Drawing a horizontal line

The following program draws a horizontal line on the screen from location (170, 240) to (470, 240) by writing pixels on the screen using function (AH=0Ch) of INT 10h. Run the program after assembling and linking it.

```
TITLE  "Program to draw a horizontal line on the screen"
.MODEL SMALL                    ; this defines the memory model
.STACK 100                      ; define a stack segment of 100 bytes
.DATA                           ; this is the data segment
.CODE                           ; this is the code segment

        MOV AX,@DATA            ; get the address of the data segment
        MOV DS, AX              ; and store it in DS register


        MOV AH, 0Fh             ; get current video mode
        INT 10h
        PUSH AX                 ; save current video mode

        MOV AH, 00h             ; set video mode
        MOV AL, 12h             ; graphics 640x480
        INT 10h

        ; draw a green color line from (170, 240) to (470, 240)
        MOV CX, 170             ; start from column 170
        MOV DX, 240             ; and row 240
        MOV AX, 0C02h           ; AH=0Ch and AL = pixel color (green)
BACK: INT 10h                   ; draw pixel
        INC CX                  ; go to next column
        CMP CX, 470             ; check if column=470
        JB BACK                 ; if not reached column=470, then continue

        MOV AH, 07h             ; wait for key press to exit program
        INT 21h

        POP AX                  ; retrieve original video mode
        MOV AH, 00h
        INT 10h                 ; restore original video mode

        MOV AX, 4C00H           ; Exit to DOS function
        INT 21H

END                             ; end of the program
```

3. Drawing a vertical line

Using the procedure followed in part 2 (drawing a horizontal line), draw a vertical line on the screen from location (320, 90) to (320, 390). Run the program after assembling and linking it.

4. Drawing a plus (+) sign in the middle of the screen

Combine the programs written for parts 2 and 3 above to draw a plus sign. All you have to do is to insert the code for drawing the vertical line [from location (320, 90) to (320, 390)] right after the code for drawing the horizontal line [from location (170, 240) to (470, 240)]. Run the program after assembling and linking it.

6.3 Lab Work:

Draw the following figure on the screen using function 0Ch of INT 10h. Assemble, link, and run it and show it to your lab instructor for credit.