LAB. 2: BIOS Interrupts (Int 10h)

Text and Pixel based Graphics

Objectives:

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

1.1 Introduction:

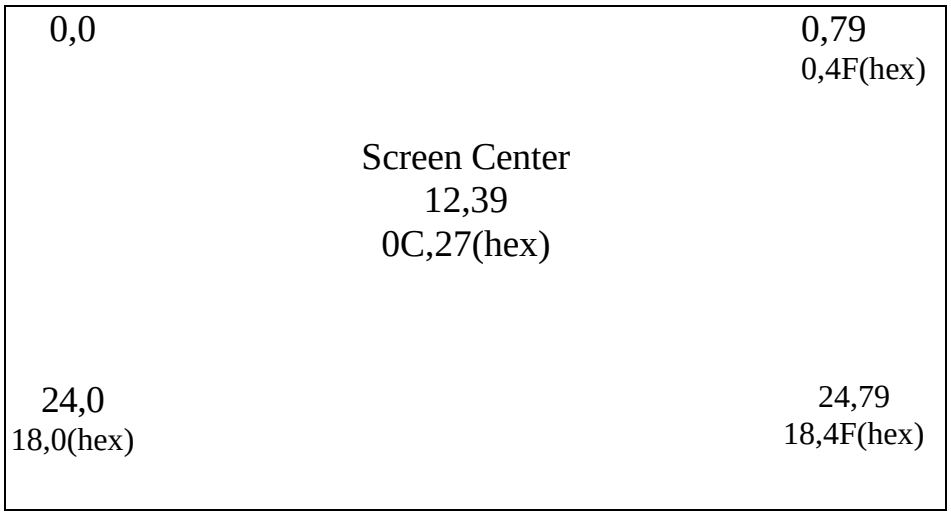
The Basic Input Output System (BIOS) is a set of x86 subroutines stored in Read-Only Memory (ROM) that can be used by any operating system (DOS, Windows, Linux, etc) for low-level input/output to various devices. Some of the services provided by BIOS are also provided by DOS. In fact, a large number of DOS services make use of BIOS services. There are different types of interrupts available which are divided into several categories as shown below:

Interrupt Types	Description
0h - 1Fh	BIOS Interrupts
20h - 3Fh	DOS Interrupts
40h - 7Fh	reserved
80h - F0h	ROM BASIC
F1h - FFh	not used

BIOS interrupt routines provide a number of services that can be used to write programs. These services include formatting disks, creating disk files, reading from or writing to files, reading from keyboard, writing to display monitor, etc. The software interrupt instruction INT is used for calling these services. 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.

1.2 Text Mode Programming



Positions on the screen are referenced using **(row, column)** coordinates. The upper left corner has coordinates (0,0). For an 80 x 25 display, the rows are 0-24 and the columns are 0-79.

The monitor screen in normal text mode is composed of 25 rows and 80 columns, and text mode is the default mode whenever a monitor is turned on.

There are several types of monitors including:

1- MDA (Monochrome Display Adapter)
2- MCGA (Multi-Color Graphics Array)
3- CGA (Color Graphics Adapter)
4- EGA (Enhanced Graphics Adapter)
5- VGA (Video Graphics Array)

In all these modes the text screen is 80X25 characters long. The text locations are numbered from 0 to 24 for the rows and 0 to 79 for the columns as shown in the diagram above.

Several functions are performed by INT 10H, therefore the programmer needs to identify which one is being used by storing an appropriate value in register AH.

For example:

AH = 00H; Selects the change video mode function

INT 10H; Executes BIOS interrupt 10H.

Depending on the function being used, other register may be used to pass information to the interrupt subroutine.

1.3 BIOS Video I/O Services

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

Set Video Mode (INT 10H, Function 00H):

Selects the video mode and clears the screen automatically.

Description: (INT 10H, Function 00H)	**Example** to set video mode to 80X25 CGA text

Invoked with: AH = 00H
AL = mode number to indicate the desired video mode
Returns: Nothing
	MOV AH, 00
	MOV AL, 03H ; text video mode
	INT 10H

03H – 80X25 CGA text

07H – 80X25 Monochrome text.

Set Cursor Position (INT 10H, Function 02H):

Sets the position of the display cursor by specifying the character coordinates.

Description: (INT 10H, Function 02H)	**Example**

Invoked with: AH = 2
BH = video page number (usually 0)
DH = row (0-24)
DL = column (0-79 for 80x25 display)
Returns: Nothing
	MOV AH, 02
	MOV BH, 0
	MOV DH, 12 ; row 12
	MOV DL, 40 ; column 40
	INT 10H

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

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**

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.

INT 10H Function 08H: Read character and attribute at cursor position

AH = 08H
BH = Display page
AH = Returned attribute byte
AL = Returned ASCII character code

INT 10H Function 09H: Write character and attribute at cursor position

AH = 09H
AL = ASCII character code
BH = Display page
BL = Attribute
CX = Number of characters to write

The character attribute is defined as shown in the following tables:

Monochrome display attributes

Blinking	Background		Intensity	Foreground		
D7	D6		D3	D2	D1	D0

D7 Non-blinking= 0
Blinking = 1
D3 Normal intensity = 0
Highlighted intensity = 1
D6 D5 D4 and D2 D1 D0 White = 0 0 0
Black = 1 1 1

CGA display attributes

Blinking	Background			Intensity	Foreground		
	R	G	B		R	G	B
D7	D6	D5	D4	D3	D2	D1	D0

D7 Non-blinking= 0
Blinking = 1
D3 Normal intensity = 0
Highlighted intensity = 1
Both blinking and intensity are applied to foreground only.
D6 D5 D4 and D2 D1 D0 Color as defined on the following table

Color Attributes

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Gray
1	0	0	1	Light blue
1	0	1	0	Light green
1	0	1	1	Light cyan
1	1	0	0	Light red
1	1	0	1	Light magenta
1	1	1	0	Yellow
1	1	1	1	High intensity white

16-Color Display

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

Pre-lab

1. The following program clears the screen and positions the cursor at a specified location on the screen using INT 10H functions. The program also displays a message string on the screen using function 09h of INT 21H.

```
LF      ; Line Feed character (0A in Hex)
CR      ; Carriage Return character (0D in Hex)

msg1        DB "WELCOME ! ", LF, CR, "$"
msg2        DB "To Microprocessor Lab ", LF, CR, "$"


MAIN PROC
    MOV AX,@DATA        ; get the address of data segment
    MOV DS, AX          ; and store it in register DS
    CALL CLEARSCREEN    ; clear the screen
    MOV DH, 10          ; row 10
     MOV DL, 13         ; column 13
    CALL SETCURSOR      ; set cursor position
    LEA DX, msg1        ; load the address offset of message to be displayed
    MOV AH, 09h         ; use DOS interrupt service for string display
     INT 21H            ; call the DOS interrupt
    MOV DH, 20          ; row 20
    MOV DL, 13          ; column 13
    CALL SETCURSOR      ; set cursor position
    LEA DX, msg2        ; load the address offset of message to be displayed
    MOV AH, 09h         ; use DOS interrupt service for string display
    INT 21H             ; call the DOS interrupt
```

```
    MOV AX, 4C00H          ; exit to DOS
    INT 21H

MAIN END PROGRAM

CLEARSCREEN PROC EDURE

    MOV AH, 00             ; set video mode
    MOV AL, 03             ; for text 80 x 25
    INT 10H RET            ; call the DOS interrupt
                           ; return to main procedure
CLEARSCREEN END PROCEDUR

SETCURSOR PROCEDURE

    MOV AH, 2              ; use DOS interrupt service for positioning screen
    MOV BH, 0              ; video page (usually 0)

    INT 10H                ; call the DOS interrupt

    RET                    ; return to main procedure

SETCURSOR END PROCEDURE

END MAIN
```

Notes:

1. The above program uses three procedures – MAIN, SETCURSOR, and CLEARSCREEN. The SETCURSOR and CLEARSCREEN procedures are called from the MAIN procedure using the CALL instruction.
2. The SETCURSOR procedure sets the cursor at a specified location on the screen whereas the CLEARSCREEN procedure uses the SET MODE function 00H of INT 10H to set the video mode to 80 x 25 text which automatically clears the screen.
3. You can display a string of characters on the screen, without using a loop, by using MOV AH, 09 with INT 21h. But the string must end with '$' character. You must also load the effective address of the string in register DX.
4. To display a string on a new line, you need to put CR after your string and LF and '$' at the end. CR stands for Carriage Return (or Enter key) and LF stands for Line Feed. You can also put 0Dh or 13 instead of CR (or cr), and 0Ah or 10 instead of LF (or lf).

2. 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.

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

MOV AH, 0Fh INT       ; get current video mode
10h PUSH AX
                      ; save current video mode
MOV AH, 00h
MOV AL, 12h           ; set video mode
INT 10h               ;graphics 640x480

; 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 MOV            ; retrieve original video mode
AH, 00h
                      ; restore original video mode
INT 10h
                      ; Exit to DOS function
MOV AX, 4C00H
INT 21H

END                   ; end of the program
```

3. 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.

```
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
        MOV DX, 240
        MOV AX, 0C02h          ;AH=0Ch and AL = pixel color (green) ;draw pixel
BACK: INT 10h
        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

                               ; retrieve original video mode
        POP AX

                               ;restore original video mode ;
        MOV AH, 00h

        INT 10h



        MOV AX, 4C00H          ;Exit to DOS function

        INT 21H

END                            ; end of the program
```

4. 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).

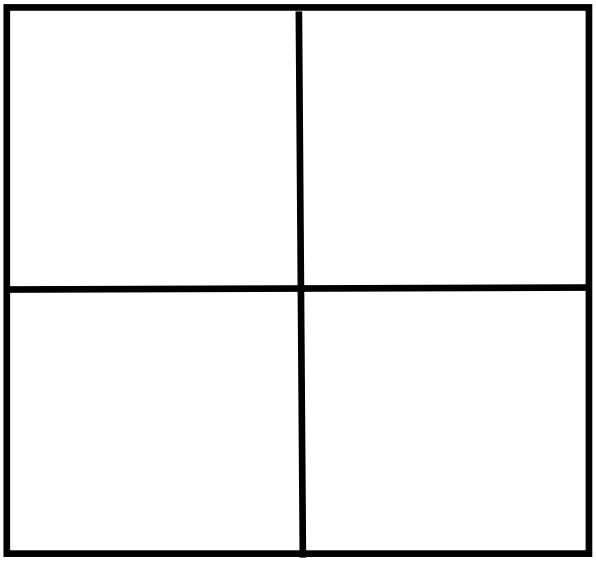
5. Drawing a plus (+) sign in the middle of the screen

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)].

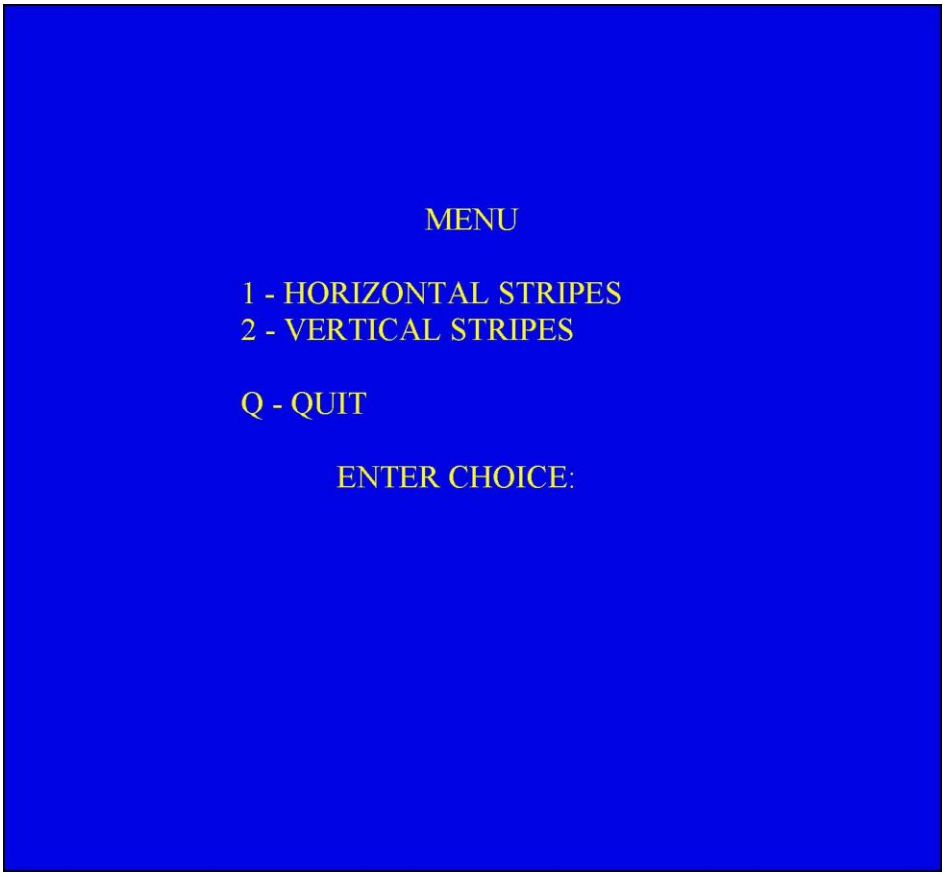
Exercises

1. Write a program that clears the screen and positions the cursor in the middle of the screen.

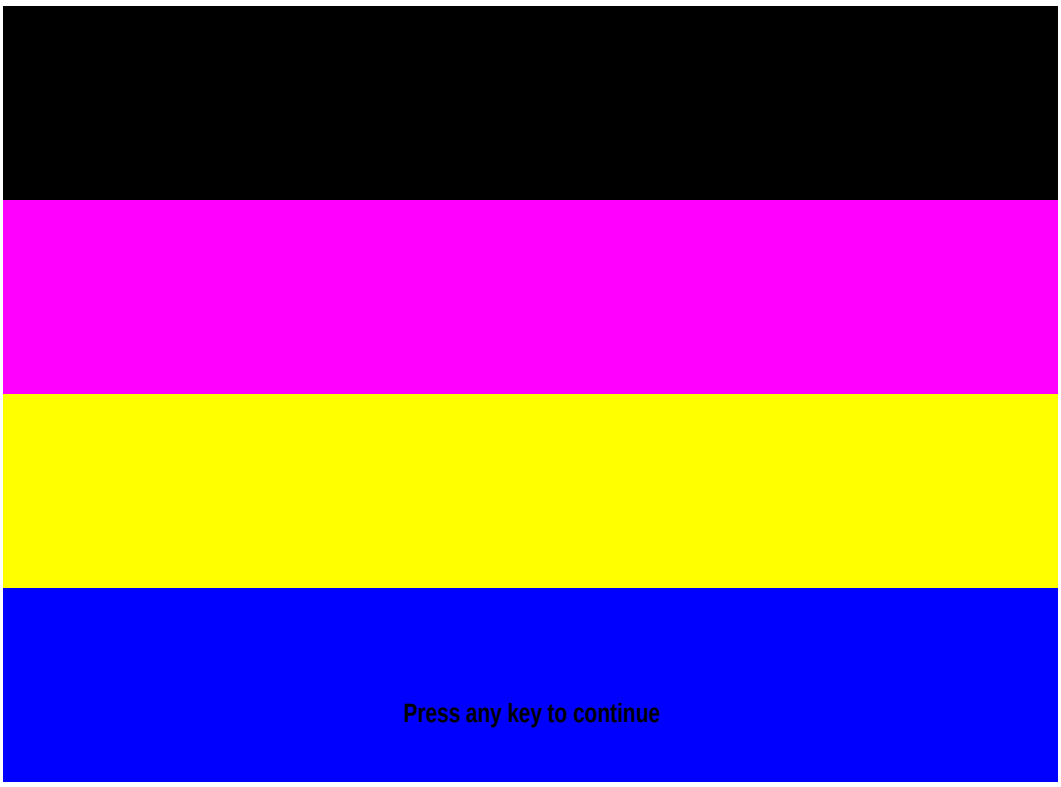
2. Draw the following figure on the screen using function 0Ch of INT 10h.



3. Using the interrupts described above, write a program to:
a- Clear the screen.
b- Create the following menu of choices:



The background color is blue and the foreground color for the letters is yellow.



4– For each choice generate the chosen pattern. Make sure the patterns have four differently colored stripes and the message: “Press any key to continue.” , is displayed. Display the new screen until any key is pressed on the keyboard then return to the main screen to display the menu of choices again.
Above is an example with 4 horizontal stripes.