Lab # 12
VIDEO Programming with INT 10h & Screen Buffer

April 22, 2013
Objective:
- To learn about Video Programming and Screen buffer.

Theory:
When an application program needs to write characters on the screen in text mode, it can choose between three types of video output:

- **MS-DOS-level access:**
  Any computer running or emulating MS-DOS can use INT 21h to write text to video display.

- **BIOS-level access:**
  Characters are output using INT 10h function, known as BIOS services. This executes more quickly than int 21h, and permits the control of text color.

- **Direct video access:**
  Characters are moved directly to video RAM (screen buffer), so the execution is instantaneous.

Screen Buffer:
- The screen buffer begins at B800:0000 and it has some rows and columns.
- Number of columns is usually 80.
- Number of rows can be 25 or 40 in most cases.
- Each line on screen takes 160 bytes (80 x 2 byte for CHARACTER + ATTRIBUTE).

VIDEO Programming with INT 10h:
There are two general types of video modes: text mode and graphics mode.

**Video Text Mode:**
**Fonts:** Characters are generated from a memory-resident table of character fonts. Originally, the table was in ROM, but later versions of the BIOS permitted programmers to rewrite the character tables at run time. This makes it possible to create custom fonts in text mode.

**Video Text Pages:** Text mode video memory is divided into multiple separate video pages, each with the ability to hold a full screen of text. Programs can display one page while writing text to other hidden pages, and they can rapidly flip back and forward between pages. In the days of high-performance MS-DOS applications, it was often necessary to keep several text screens in memory at the same time. With the current popularity of graphical interfaces, this text page feature is no longer so important. The default video page is **Page 0**.
Attributes: each screen character is assigned an attribute byte that controls both the color of the character (called the foreground) and the screen color behind the character (called the background). Each position on the video display holds a single character, along with its own attribute (color). The attribute is stored in a separate byte, following the character in memory.

Blinking: Characters on the video display can blink. The video controller does this by reversing the foreground and background colors of a character at a predetermined rate. By default, when a PC boots into MS-DOS mode, blinking is enabled. It is possible to turn blinking off using a video BIOS function. Also, blinking is off by default when you open up an MS-DOS emulation window under MS-Windows.

Controlling the Color:

Attribute Byte
In color text mode, each character is assigned an attribute byte, which consists of two 4-bit color codes: background and foreground:

![Attribute Byte Diagram]

Blinking There is one complication to this simple color scheme. If the video adapter currently has blinking enabled, the high bit of the background color controls the character blinking. When this bit is set, the character blinks:

![Blinking Diagram]

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Four-Bit Color Text Encoding.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRGB</td>
<td>Color</td>
</tr>
<tr>
<td>0000</td>
<td>black</td>
</tr>
<tr>
<td>0001</td>
<td>blue</td>
</tr>
<tr>
<td>0010</td>
<td>green</td>
</tr>
<tr>
<td>0011</td>
<td>cyan</td>
</tr>
<tr>
<td>0100</td>
<td>red</td>
</tr>
<tr>
<td>0101</td>
<td>magenta</td>
</tr>
<tr>
<td>0110</td>
<td>brown</td>
</tr>
<tr>
<td>0111</td>
<td>light gray</td>
</tr>
</tbody>
</table>
INT 10h Video Functions:

- AH register contains the function number
- 00h: Set video mode
  - Text modes listed in Table 2
  - Graphics modes listed in Table 3
- 01h: Set cursor lines
- 02h: Set cursor position
- 03h: Get cursor position and size
- 06h: Scroll window up
- 07h: Scroll window down
- 08h: Read character and attribute
- 09h: Write character and attribute
- 0Ah: Write character
- 10h (AL = 03h): Toggle blinking/intensity bit
- 0Fh: Get video mode

### Table 2
Video Text Modes Recognized by INT 10h.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Resolution (columns X rows)</th>
<th>Number of Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40 X 25</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>40 X 25</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>80 X 25</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>80 X 25</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>80 X 25</td>
<td>2</td>
</tr>
<tr>
<td>14h</td>
<td>132 X 25</td>
<td>16</td>
</tr>
</tbody>
</table>

### Table 3
Video Graphics Modes Recognized by INT 10h.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Resolution (columns X rows, in pixels)</th>
<th>Number of Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>640 X 200</td>
<td>2</td>
</tr>
<tr>
<td>0Dh</td>
<td>320 X 200</td>
<td>16</td>
</tr>
<tr>
<td>0Eh</td>
<td>640 X 200</td>
<td>16</td>
</tr>
<tr>
<td>0Fh</td>
<td>640 X 350</td>
<td>2</td>
</tr>
<tr>
<td>10h</td>
<td>640 X 350</td>
<td>16</td>
</tr>
<tr>
<td>11h</td>
<td>640 X 480</td>
<td>2</td>
</tr>
<tr>
<td>12h</td>
<td>640 X 480</td>
<td>16</td>
</tr>
<tr>
<td>13h</td>
<td>320 X 200</td>
<td>256</td>
</tr>
<tr>
<td>6Ah</td>
<td>800 X 600</td>
<td>16</td>
</tr>
</tbody>
</table>
### INT 10h Function 0

**Description**: Set the video mode

**Receives**
- AH = 0
- AL = video mode

**Returns**: nothing

**Sample Call**
- `mov ah, 0`
- `mov al, 3 ; video mode 3 (color text)`
- `int 10h`

**Notes**: The screen is cleared automatically unless the high bit in AL is set before calling this function.

### INT 10h Function 01h

**Description**: Set cursor lines

**Receives**
- AH = 01h
- CH = top line
- CL = bottom line

**Returns**: nothing

**Sample Call**
- `mov ah, 1`
- `mov ox, 0607h ; default color cursor size`
- `int 10h`

**Notes**: The monochrome display uses 12 lines for its cursor, while all other displays use 8 lines.

### INT 10h Function 02h

**Description**: Set cursor position

**Receives**
- AH = 2
- DH, DL = row, column values
- BH = video page

**Returns**: nothing

**Sample Call**
- `mov ah, 2`
- `mov dh, 10 ; row 10`
- `mov dl, 20 ; column 20`
- `mov bh, 0 ; video page 0`
- `int 10h`
### INT 10h Function 03h

**Description:** Get cursor position and size

**Receives:**
- AH = 3  
  BH = video page

**Returns:**
- CH, CL = starting, ending cursor scan lines  
- DH, DL = row, column of cursor’s location

**Sample Call:**
- mov ah, 3  
- mov bh, 0  
  ; video page 0
- int 10h
- mov cursor, CX  
- mov position, DX

### INT 10h Function 06h

**Description:** Scroll window up

**Receives:**
- AH = 6  
  AL = number of lines to scroll (0 = all)  
  BH = video attribute for blanked area  
  CH, CL = row, column of upper left window corner  
  DH, DL = row, column of lower right window corner

**Returns:**
- nothing

**Sample Call:**
- mov ah, 6  
  ; scroll window up
- mov al, 0  
  ; entire window
- mov ch, 0  
  ; upper left row
- mov cl, 0  
  ; upper left column
- mov dh, 24  
  ; lower right row
- mov dl, 79  
  ; lower right column
- mov bh, 7  
  ; attribute for blanked area
- int 10h  
  ; call BIOS

### INT 10h Function 08h

**Description:** Read character and attribute

**Receives:**
- AH = 8  
  BH = video page

**Returns:**
- AL = ASCII code of the character  
- AH = attribute of the character

**Sample Call:**
- mov ah, 8
- mov bh, 0  
  ; video page 0
- int 10h
- mov char, al  
  ; save the character
- mov attrib, ah  
  ; save the attribute
### INT 10h Function 09h

<table>
<thead>
<tr>
<th>Description</th>
<th>Write character and attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receives</td>
<td>AH = 9</td>
</tr>
<tr>
<td></td>
<td>AL = ASCII code of character</td>
</tr>
<tr>
<td></td>
<td>BH = video page</td>
</tr>
<tr>
<td></td>
<td>BL = attribute</td>
</tr>
<tr>
<td></td>
<td>CX = repetition count</td>
</tr>
<tr>
<td>Returns</td>
<td>nothing</td>
</tr>
<tr>
<td>Sample Call</td>
<td>mov ah, 9</td>
</tr>
<tr>
<td></td>
<td>mov al, 'A'</td>
</tr>
<tr>
<td></td>
<td>mov bh, 0</td>
</tr>
<tr>
<td></td>
<td>mov bl, 0F1h</td>
</tr>
<tr>
<td></td>
<td>mov cx, 1</td>
</tr>
<tr>
<td></td>
<td>int 10h</td>
</tr>
<tr>
<td>Notes</td>
<td>Does not advance the cursor after writing the character.</td>
</tr>
</tbody>
</table>

### INT 10h Function 0Ah

<table>
<thead>
<tr>
<th>Description</th>
<th>Write character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receives</td>
<td>AH = 0Ah</td>
</tr>
<tr>
<td></td>
<td>AL = character</td>
</tr>
<tr>
<td></td>
<td>BH = video page</td>
</tr>
<tr>
<td></td>
<td>CX = repetition count</td>
</tr>
<tr>
<td>Returns</td>
<td>nothing</td>
</tr>
<tr>
<td>Sample Call</td>
<td>mov ah, 0Ah</td>
</tr>
<tr>
<td></td>
<td>mov al, 'A'</td>
</tr>
<tr>
<td></td>
<td>mov bh, 0</td>
</tr>
<tr>
<td></td>
<td>mov cx, 1</td>
</tr>
<tr>
<td></td>
<td>int 10h</td>
</tr>
<tr>
<td>Notes</td>
<td>Does not advance the cursor.</td>
</tr>
</tbody>
</table>
Lab work:

**Exercise 1:**

Clean the screen using the INT 10h Video bios, then print the word "Hello" in the center of the screen.

```
.model small
.data
msg db 'Hello','$'
.code
main:
nou ax,0data
nou ds.ax
;;;;; CLEAR THE SCREEN;;;;;;;
nou ah,6
nou al,0
nou cl,0 ; start point of columns
nou ch,0 ; start point of Rows
nou d1.4fh ; end point of columns
nou dh,10h ; end point of Rows
nou bh,2 ; Back-Black,Fore-Green
int 10h
;;;;;POSITION THE CURSOR IN THE CENTER;;;;;
nou ah,2
nou bh,0
nou d1.27h ; mid-point of columns
nou dh,0ch ; mid-point of Rows
int 10h
;;;;;PRINT THE MSG;;;;;;;
nou ah,09h
nou dx.offset msg
int 21h
;;;;;END;;;;;;;;;;;;;;;;
nou ah,4ch
int 21h
end main
```
Activity:
Print the previous word at the origin location (0,0) with red foreground and green background.

Excercise2:
Write a program to clean the lower part of the screen with a green background and print your name in the middle of it with a light red background and white foreground.

Output:
Exercise 3:
Write a program to color the screen background with three colors vertically.

Output:
Homework:

Write a program to color the screen background with three colors horizontally, Like this: