Experiment 11

11 Accessing Video Memory

Introduction

This experiment introduces the use of the VGA controller and BIOS INT 10H functions to access video memory using the different graphics modes.

You will be provided with some routines that use the video modes. You should understand these routines before inserting them properly in your programs.

Objectives

1. Use of the 640x480 16-color graphics display mode.
2. Use mode 12H to divide the screen into a 53 line by 80 characters per line to display blocks of colors.
3. Display text on the 640x480 16-color graphics display without changing the background color.

11.1 Text Mode

In DOS mode, the videotext memory starts at address location **B800:0000** through **B800:FFFF** and contains ASCII data and attributes for display. In text mode, the following functions are used to display data on the screen.

Function 01H and **02H/06:** Used to read/display one character on the screen.

Function 09H/0AH: Used to display/read a string.

These functions have already been seen in previous experiments.

11.2 Graphics Mode

11.2.1 Video Adapters

The display on the monitor is controlled by a circuit known as the **video adapter**. This circuit, which is usually an add-in card, has two basic units:

- a ***display memory*** (also called a video buffer or video RAM) and
- a ***video controller***.

The display memory stores the information to be displayed. Although it is physically located in the video adapter, it is still part of the processor address space and may be manipulated by both normal 80x86 instructions or through the video controller. The display memory starts at physical address **A0000H** or above depending on the particular video adapter. The video controller translates data into an image on the screen.

11.2.2 Video Modes

The IBM PC provides two types of displays and associated adapters.

a. The monochrome adapter (MDA) is used to display black and white (or green) text characters.
b. The color/graphics adapters (CGA, VGA, MCGA, EGA, SVGA, EVGA, XVGA etc.) may operate in two modes: text or graphics.

In the alphanumeric mode (text mode), 80 columns by 25 rows, or 40 columns by 25 rows, color character displays can be generated. In the graphics mode, the screen may be viewed as consisting of a rectangular grid pattern consisting of dots or **pixels**. A picture can be displayed by specifying the color of each pixel on the screen. Depending on the kind of adapter present, a program can select text or graphics mode. Each mode is identified by a ***video mode number*** as shown in the following table.

Mode	Type	Rows	Cols	Resolution	Colors	MDA	CGA	EGA	MCGA	VGA
0	Text	25	40	320x200	16		ì	ì	ì	ì
1	Text	25	40	320x200	16		ì	ì	ì	ì
2	Text	25	80	640x200	16		ì	ì	ì	ì
3	Text	25	80	640x200	16		ì	ì	ì	ì
4	Graph	25	40	320x200	4		ì	ì	ì	ì
5	Graph	25	40	320x200	4		ì	ì	ì	ì
6	Graph	25	80	640x200	2		ì	ì	ì	ì
7	Text	25	80	720x350	Mono	ì		ì		ì

Table 11.1: Video Modes

Depending on the kind of adapter present, a program can select text or graphics mode. Each mode is identified by a video mode number as shown in the following table.

11.2.3 The VGA Mode

The 640x480 16-color graphics display mode uses memory location A000:0000 through A000:FFFF to access graphics data. In order to display 16 colors with a resolution of 640x 480 a memory greater than 64K bytes is required. Because 16 colors require 4 bits, and the resolution is 640 x 480 (i.e. 307,200 pixels), the memory system requires 640 x 480 x 4 (i.e. 1,228,800 bits) or 153,600 bytes of video memory in this display mode.

To allow access to such as amount of memory, mode 12H display is designed to be accessed in bit planes. A bit plane is a linear section of memory that contains one of the four bits to display the 16 colors. Each bit plane requires 307,200 bits of memory, stored in 38,400 bytes of memory. The 64K bytes at segment A000H are enough to only address a single bit plane at a time. The bit plane is addressed at memory locations A000:0000 through A000:95FF. In a 640x480 display, location A000:0000

represents the upper leftmost 8 pixels, and location A000:95FF represents the lower rightmost 8 pixels.

There are four planes, or banks of memory, that overlap this address range to represent the four bits or color for each pixel (

Figure **11. 1**). To change the color of one pixel on the video display four bits need to be changed, one in each bit plane. The color codes used for a standard VGA display are shown in Table 11.2. If all 4-bit planes are cleared, black is the pixel color.

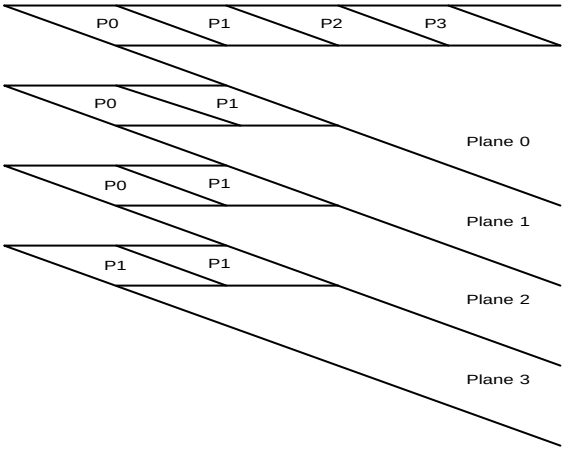


Figure 11. 1: The four bit-planes of the 640x480, 16-color VGA display

11.3 Accessing the Video Memory

Access to video memory in mode 12H is accomplished through the following steps:

Step 1:Read the byte of memory to be changed, to load the bit plane information into the video card.

Step 2:Select and address a single pixel (bit) through the graphics address register (GAR) and bit mask register (BMR). This is accomplished by sending an 8 out to I/O port 03CEH, which represents the GAR.

Steps 1 and 2 are done through the following set of instructions:

```
MOV DX, 03CEH        ; Select VGA address card
MOV AL, 08           ; Index of 8
OUT DX, AL           ; Select Index 8
```

Step 3:Load AL with the bits to be changed (a one bit represents a pixel to be changed), and send this out to the Bit Mask Register (BMR), or I/O port 03CFH.

```
MOV DX, 03CFH          ; Select BMR
MOV AL, 80H            ; Place mask in AL
OUT DX, AL             ; Select leftmost bit using 80H
```

Step 4:Set all mask bits to 1's (1111 or 0FH) in the Map Mask Register (MMR) at sequencer offset 2, and write color 0 to the VGA card (black) to the address containing the pixel, to clear the old color from the pixel.

Mask bits select the bit planes to be changed. If all are selected and a color 0 is written, all four-bit planes are cleared to zero. To do so, use the following code:

```
MOV DX, 03C4H          ; Select VGA sequencer register
MOV AL, 02             ; Index of 2
OUT DX, AL             ; Select Index 2
MOV DX, 03C5H          ; Address MMR
MOV AL, 0FH        ; Mask to 1111 binary
OUT DX, AL
```

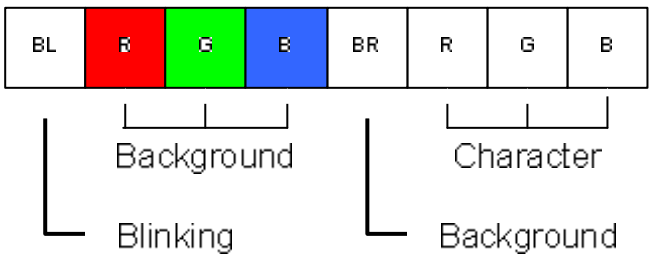
Step 5:Send the desired color number to the MMR and write an FFH to the video memory. This places a logic one in only the selected bit planes.

To write a new color to a pixel on the screen, use the following instructions:

```
MOV AL, Color          ; Choose color; e.g. 03 for cyan
OUT DX, AL             ; Select color
; Next write an FFH to the selected video memory location
```

Register	Meaning	Address
GAR	Graphics Address Register	03CEH
BMR	Bit Mask Register	03CFH
MMR	Map Mask Register	03C4H to access 03C5H to select bit planes

Table 11. 2: Registers used in Video Mode



Code				Color
br	R	G	B	
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	Grey
1	0	0	1	Bright Blue
1	0	1	0	Bright Green
1	0	1	1	Bright Cyan
1	1	0	0	Bright Red
1	1	0	1	Bright Magenta
1	1	1	0	Yellow
1	1	1	1	Bright White

Table 11.3: Colors Available to VGA, Mode 12H

11.4 Direct Video Access in Text Mode

The characters seen on the video monitor correspond directly to ASCII bytes stored in the video RAM. Thus to display a character, by direct video access, one need only place the ASCII code for that character into the correct video RAM location.

Example: The following program fills a screen with A's by direct video access. It uses the default text mode 3

```
,STACK 200
.CODE
.STARTUP
        MOV AX , 0B800H
                MOV DS , AX
                MOV CX , 2000                   ; 2000 words
                MOV DI , 0
FILL_PAGE:      MOV WORD PTR [DI], 7041h        ; black A on white
                ADD DI , 2
                LOOP FILL_PAGE
                MOV AH , 08H                    ; wait for a keystroke
                INT 21H
.EXIT
END
```

The formula for calculating a video memory offset address, in video page 0, given a screen row and column coordinate pair is:

Character offset = (row# * 80 + column#) * 2
= (row# * (64 + 16) + column#) * 2

Using the above formula the following procedure calculates an 80 * 25 text-mode memory address from a pair of row and column coordinates, contained in DH and DL respectively:

```
CALC_ADDRESS   PROC
        ; input: DH = row number (0 - 24) , DL = column number (0 - 79) ,
    VIDEO_SEG a constant which contains  ; either B000H or B800H
        ; output: ES:DI contains the required segment : offset address

        PUSH  AX
        MOV  AX ,  VIDEO_SEG
        MOV  ES ,  AX
        MOV  AH , 0
        MOV  AL , DH            ; AX := row#
        SHL  AX , 1             ;  AX := row# * 2
        SHL  AX , 1             ; AX := row# * 4
        SHL  AX , 1             ; AX := row# * 8
        SHL  AX , 1             ; AX := row# * 16
        MOV  DI , AX            ; DI := row# * 16
        SHL  AX , 1             ; AX := row# * 32
        SHL  AX , 1             ; AX := row# * 64
        ADD  DI , AX            ; DI := row# * 80
        MOV  AH , 0
        MOV  AL , DL            ; AX := column#
        ADD  DI ,  AX           ; DI := row# * 80  +  column#
        SHL  DI , 1             ; DI := ( row# * 80  +  column# ) * 2
        POP  AX
        RET
    CALC_ADDRESS  ENDP
```

Thus, for example, to display a yellow blinking T on a green background at row 6 and column 37, by direct video access, use :

```
        MOV  DH , 6             ; row#6
        MOV  DL , 37                    ; column#37
        CALL  CALC_ADDRESS
        MOV  AH , 10101110B     ; attribute: yellow on green
        MOV  AL , 'T'
        STOSW
```

Note: The effect of STOSW is:

```
        MOV  ES:[DI] , AL
```

```
MOV  ES:[DI + 1] , AH
```

11.5 Bios Video Functions

Another way of accessing video memory is through **INT 10H**. This method is recommended for most applications, since it frees the user from the burden of calculating video memory addresses. The following are most functions used with INT 10H, these allow most useful video tasks. Note that INT 10H preserves only the BX, CX, DX, and the segment registers

11.5.1 Accessing the Video Memory

Before accessing video, make sure that you save the current video mode so that you can restore it once your program is finished. This can be done using the following sequence of instructions: (INT 10H)

```
MOV AH, 0FH     ; Get current video mode
INT 10H
PUSH AX         ; Save Video mode in AL and Number of columns AH
…
POP AX          ; Restore Video mode AL and Number of columns AH
MOV AH, 00
INT 10H
```

11.5.2 Select Video Mode

```
MOV AH, 00
MOV AL, VIDEO_MODE
INT 10H
```

Function 00 automatically clears the screen. To preserve the screen while changing the mode set the most significant bit of AL to 1.

```
MOV   AH,  00
MOV   AL,  VIDEO_MODE
OR    AL,  80H
INT   10H
```

11.5.3 Get Current Video Mode

```
MOV   AH, 0FH
INT   10H
PUSH  AX            ; Or MOV Old_Video_Mode, AX
```

11.5.4 Restore Previous Video Mode

```
POP   AX            ; Or MOV AX, Old_Video_Mode
MOV   AH, 00H
```

```
INT     10H
```

11.5.5 Cursor Manipulation

If the row and column numbers are in Hexadecimal they can directly be assigned to the DX register. The cursor positioning on the current video page is independent of the other video pages.

1. Set Cursor Position

To set the cursor at a given position use:

```
MOV   AH, 02H
MOV   BH,  Current_Video_Page_Number   ;Usually 0
MOV   DH,  Row_Number
MOV   DL,  Column_Number
INT     10H
```

2. Get Cursor Position

To get the cursor position use:

```
MOV   AH, 03H
MOV   BH, Current_Video_Page_Number    ; Usually 0
INT     10H
MOV   Save_Cursor, CX
MOV   Current_Row, DH
MOV   Current_ Column, DL
```

3. Set Cursor Size

The cursor is displayed using starting and ending scan lines. In Mono mode the cursor uses 12 lines (0,1,2, .. 0BH,0CH), whereas in color mode it uses 8 lines (0,1, ..,6,7).

```
MOV   AH, 01H
MOV   CH, Start_Scan_Line#
MOV   CL, End_Scan_Line#
INT     10H
```

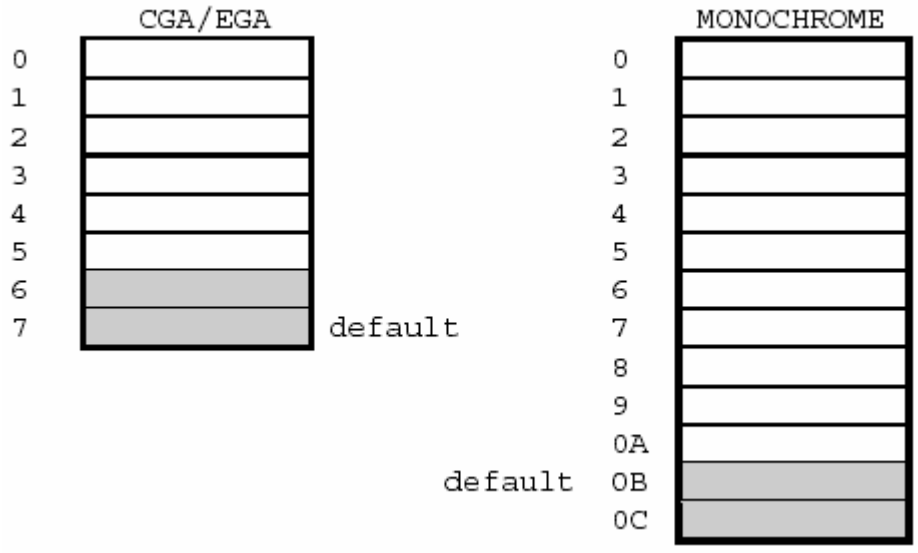


Figure 11. 3: Cursor Size

Example:

a. To set the cursor to its maximum size in color mode

```
MOV   AH, 01H
MOV   CX, 0007H
INT   10H
```

b. To set the cursor to its maximum size in monochrome mode

```
MOV   AH, 01H
MOV   CX, 000CH
INT   10H
```

4. Write Pixel

To put a pixel on the screen

```
MOV   AH, 0CH
INT   10H
```

5. Restore the current cursor size:

```
MOV   AH, 01H
MOV   CX, Cursor_Size      ; Cursor_Size previously saved
INT   10H
```

6. Make the Cursor Invisible

Set the starting scan line to an illegal value by setting bit 5 in CH to 1.

```
MOV   AH, 01H
OR    CH, 00100000B        ; Or MOV      CX, 2000H
INT   10H
```

Another way of hiding the cursor is to place it in the undisplayed portion of the video page, e.g. row #25 column # 0.

```
MOV   DH, 25               ;Row number
MOV   DL, 00               ;Column number
MOV   AH, 02H
MOV   BH, 00               ;Video page # 0
INT   10H
```

11.5.6 Set Border Color

To set the border to a given color use the following sequence:

```
MOV   AH, 0BH
MOV   BH, 00H
```

```
MOV   BL, 04H
INT   10H
```

11.6 Lab Work

11.6.1 Pre-Lab

1- Write two Macros: one to get the current video mode, and the other restore the video mode.
2- Write and run programs 11.1 and 11.2. Write your programs using macros and procedures.
3- Prepare all programs in this experiment by writing them using macros and procedures.

11.6.2 Lab Work

1- Show programs 11.1 and 11.2 to your lab instructor.
2- Write and run programs 11.3 and 11.4.
3- Write a program that displays the time on the top right hand corner of the display. Use INT 11H function 02, which inputs the column and row numbers in DL and DH respectively, and Video page (usually 0) in BH.

11.6.3 Lab Assignment

Rewrite the program that displays the time on the screen using graphics mode only. Review the part that shows how to display text in video mode.

Title 'Program 11-1'

```
; A program that blanks the test mode screen and makes it red.
; It then displays the message This is a test line. before
; returning to DOS.
;
        .MODEL SMALL
        .DATA
        MES    DB      'This is a test line.$'
        .CODE
        .STARTUP
        MOV    AX,0B800H     ;address text segment
        MOV    ES,AX
        CLD                  ;select increment
        MOV    DI,0          ;address text offset
        MOV    AH,40H                ;attribute black on red
        MOV    AL,20H                ;character is space
        MOV    CX,25*80      ;set count
        REP    STOSW                 ;clear screen and change attributes

        MOV    AH,2          ;home cursor
        MOV    BH,0          ;page 0
        MOV    DX,0          ;row 0, char 0
        INT    10H

        MOV    DX,OFFSET MES       ;display "This is a test line."
        MOV    AH,9
        INT    21H
        .EXIT
        END
```

Title 'Program 11-2'

```
;a program that displays all of 256 colors available to the
;320 x 200 video display mode (13H)
;***uses***
;the BAND procedure to display 64 colors at a time in a band
;on the display.
;
        .MODEL TINY
        .CODE
        .STARTUP
        MOV    AX,13H                ;select mode 13H
        INT    10H

        MOV    AX,0A000H     ;address segment A000 with ES
        MOV    ES,AX
        CLD                  ;select increment
        MOV    DI,0          ;address offset 0000

        MOV    AL,0          ;load starting test color of 00H
        CALL   BAND          ;display one band of 64 colors

        MOV    AL,64         ;load starting color of 40H
        CALL   BAND          ;display one band of 64 colors
```

```
        MOV   AL,128        ;load starting color of 80H
        CALL  BAND          ;display one band of 64 colors

        MOV   AL,192        ;load starting color of C0H
        CALL  BAND          ;display one band of 64 colors

        MOV   AH,1          ;wait for any key
        INT   21H

        MOV   AX,3          ;switch back to DOS video mode
        INT   10H
        .EXIT
;
;the BAND procedure displays a color band of 64 colors
;***input parameter***
;AL = starting color number
;ES = A000H
;DI = starting offset address for display
;
BAND PROC  NEAR
        MOV   BH,40         ;load line count
BAND1:
        PUSH  AX            ;save starting color
        MOV   CX,64         ;load color across line count
BAND2:
        MOV   BL,5          ;load times color is displayed
BAND3:
        STOSB               ;store color
        DEC   BL
        JNZ   BAND3             ;repeat 5 times
        INC   AL            ;change to next color
        LOOP  BAND2             ;repeat for 64 colors
        POP   AX            ;restore starting color
        DEC   BH
        JNZ   BAND1             ;repeat for 40 lines
        ADD   DI,320*10     ;skip 10 lines
        RET

BAND ENDP
        END
```

Title 'Program 11-3'

```
 ;a program that displays all the possible brightness levels of the
;color red for the 320 x 200, 256-color mode (13H)
;
.MODEL TINY
.CODE
.STARTUP

        MOV   AX,13H            ;switch to mode 13H
        INT   10H

        MOV   AX,0A000H   ;address segment A000 with ES
        MOV   ES,AX
        CLD               ;select increment

        MOV   CH,0        ;green value
        MOV   CL,0        ;blue value
        MOV   DH,0        ;red value
        MOV   BX,80H            ;color register number 80H
        MOV   AX,1010H    ;change palette color function
        MOV   DL,64       ;count to change colors 80H to BFH
PROG1:
        INT   10H         ;change a color value
        INC   DH          ;next color of red
        INC   BX          ;next color palette register
        DEC   DL
        JNZ   PROG1       ;repeat for 64 colors

        MOV   DI,0        ;address offset 0000
        MOV   AL,80H            ;starting color number
        CALL  BAND        ;display 64 colors

        MOV   AH,1        ;wait for any key
        INT   21H

        MOV   AX,3        ;switch back to DOS video mode
        INT   10H
.EXIT

;the BAND procedure displays a color band of 64 colors
;***input parameter***
;AL = starting color number
;ES = A000H
;DI = starting offset address for display
;
BAND PROC  NEAR

        MOV   BH,40       ;line count of 40
BAND1:
        PUSH  AX          ;save starting color number
        MOV   CX,64       ;color count of 64
BAND2:
        MOV   BL,5        ;load times color is displayed
BAND3:
        STOSB    ;store color
```

```
        DEC   BL
        JNZ   BAND3             ;repeat 5 times
        INC   AL          ;get next color number
        LOOP  BAND2             ;repeat for all 64 colors
        POP   AX          ;restore original color number
        DEC   BH
        JNZ   BAND1             ;repeat for 40 raster lines
        ADD   DI,320*10   ;skip 10 raster lines
        RET

BAND ENDP
        END
```

Title 'Program 11-4'

```
;a program that displays a green box on the video screen using
;video mode 13H.
;
.MODEL TINY
.CODE
.STARTUP
        CLD               ;select auto-increment

        MOV   AX,13H            ;select mode 13H
        INT   10H         ;this also clears the screen

        MOV   AL,2        ;use color 02H (green)
        MOV   CX,100      ;starting column number
        MOV   SI,10       ;starting row number
        MOV   BP,75       ;size
        CALL  BOX         ;display box

        MOV   AH,1        ;wait for any key
        INT   21H

        MOV   AX,3        ;switch to DOS video mode
        INT   10H
.EXIT

;the BOX procedure displays a box on the mode 13H display.
;***input parameters***
;AL = color number (0-255)
;CX = starting column number (0-319)
;SI = starting row number (0-199)
;BP = size of box
;
BOX   PROC  NEAR
        MOV   BX,0A000H   ;address segment A000 with ES
        MOV   ES,BX
        PUSH  AX          ;save color
        MOV   AX,320      ;find starting PEL
        MUL   SI
        MOV   DI,AX       ;address start of BOX
        ADD   DI,CX
        POP   AX
```

```
        PUSH  DI            ;save starting offset address
        MOV   CX,BP         ;save size in BP
BOX1:   REP   STOSB         ;draw top line
        MOV   CX,BP
        SUB   CX,2          ;adjust CX

BOX2:   POP   DI
        ADD   DI,320        ;address next row
        PUSH  DI
        STOSB               ;draw PEL
        ADD   DI,BP
        SUB   DI,2
        STOSB               ;draw PEL
        LOOP  BOX2

        POP   DI
        ADD   DI,320        ;address last row
        MOV   CX,BP
        REP   STOSB
        RET
BOX     ENDP
        END
```

Title 'Program 11-5'

```
;a program that displays a short cyan line that is 10 Pixels wide
;with a red dot below and to the right of the cyan line.
.MODEL TINY
.CODE
.STARTUP
                MOV   AX,0A000H     ;address video RAM at segment A000
                MOV   DS,AX
                CLD                 ;select increment

                MOV   AX,12H        ;set mode to 12H
                INT   10H           ;and clear screen

                MOV   CX,10         ;set dot count to 10
                MOV   BX,10         ;row address
                MOV   SI,100        ;column address
                MOV   DL,3          ;color 3 (cyan)
MAIN1:          CALL  DOT           ;plot 10 dots
                INC   SI                    ;display one dot
                LOOP  MAIN1         ;repeat 10 times

                MOV   BX,40         ;row address
                MOV   SI,200        ;column address
                MOV   DL,4          ;color 4 (red)
                CALL  DOT           ;display one red dot

                MOV   AH,1          ;wait for key
                INT   21H
                MOV   AX 0003
                INT   10H           ;return to DOS video mode
.EXIT
```

```
;the DOT procedure displays one dot or PEL on the video display.
;BX = row address (0 to 479)
;SI = column address (0 to 639)
;DL = color (0 to 15)
;
DOT   PROC NEAR
      PUSH CX
      PUSH DX           ;save color
      MOV  AX,80        ;find row address byte
      MUL  BX
      MOV  DI,AX        ;save it
      MOV  AX,SI        ;find column address byte
      MOV  DH,8
      DIV  DH
      MOV  CL,AH        ;get shift count
      MOV  AH,0
      ADD  DI,AX        ;form address of PEL byte

      MOV  AL,80H
      SHR  AL,CL        ;find bit in bit mask register
      PUSH AX           ;save bit mask

      MOV  DX,3CEH            ;graphics address register
      MOV  AL,8         ;select bit mask register
      OUT  DX,AL

      MOV  DX,3CFH            ;bit mask register
      POP  AX           ;get bit mask
      OUT  DX,AL

      MOV  DX,3C4H            ;sequence address register
      MOV  AL,2         ;select map mask register
      OUT  DX,AL

      MOV  DX,3C5H            ;map mask register
      MOV  AL,0FH             ;enable all planes
      OUT  DX,AL

      MOV  AL,[DI]            ;must read first
      MOV  BYTE PTR [DI],0    ;clear old color
      POP  AX           ;get color from stack
      PUSH AX
      OUT  DX,AL
      MOV  BYTE PTR [DI],0FFH;write memory

      POP  DX           ;restore registers
      POP  CX
      RET

DOT   ENDP

END
```

Title 'Program 11-6'

```
;a program that display a cyan bar across the top of a white
;screen.
;
        .MODEL TINY
        .CODE
        .STARTUP
        MOV   AX,0A000H     ;address video RAM at segment A000
        MOV   DS,AX
        CLD                 ;select increment

        MOV   AX,12H                ;set mode to 12H
        INT   10H           ;and clear screen

        MOV   CX,80         ;block count
        MOV   BX,0          ;row address
        MOV   SI,0          ;column address
        MOV   DL,3          ;color 3 (cyan)
MAIN1:                              ;plot 80 blocks
        CALL  BLOCK                 ;display a block
        INC   SI            ;address next column
        LOOP  MAIN1         ;repeat 80 times

        MOV   BX,1          ;row address
        MOV   DL,7          ;color 7 (white)
        MOV   DH,52         ;row count
MAIN2:
        MOV   SI,0          ;column address
        MOV   CX,80         ;column count
MAIN3:
        CALL  BLOCK                 ;display a block
        INC   SI            ;address next column
        LOOP  MAIN3         ;repeat 80 times
        INC   BX            ;increment row address
        DEC   DH
        JNZ   MAIN2         ;repeat 52 times

        MOV   AH,1          ;wait for key
        INT   21H

        MOV   AX,3
        INT   10H           ;return to DOS video mode
        .EXIT
;
;The BLOCK procedure displays one block that is 8 pixels
;wide by 9 pixels high.
;BX = row address (0 to 52)
;SI = column address (0 to 79)
;DL = block color (0 to 15)
;
BLOCK       PROC  NEAR

        PUSH  CX
        PUSH  DX            ;save color
```

```
        MOV   DX,3CEH           ;graphics address register
        MOV   AL,8          ;select bit mask register
        OUT   DX,AL
        MOV   DX,3CFH           ;bit mask register
        MOV   AL,0FFH           ;enable all 8 bits
        OUT   DX,AL

        MOV   DX,3C4H           ;sequence address register
        MOV   AL,2          ;select map mask register
        OUT   DX,AL

        MOV   AX,80*9           ;find row address byte
        MUL   BX
        MOV   DI,AX         ;save it
        ADD   DI,SI         ;form address of PEL byte

        MOV   CX,9          ;byte count
        MOV   DX,3C5H           ;map mask register
        POP   AX            ;get color
        PUSH  AX
        MOV   AH,AL
BLOCK1:
        MOV   AL,0FH            ;enable all planes
        OUT   DX,AL
        MOV   AL,[DI]           ;must read first
        MOV   BYTE PTR [DI],0   ;clear old color
        MOV   AL,AH
        OUT   DX,AL
        MOV   BYTE PTR [DI],0FFH  ;write memory
        ADD   DI,80
        LOOP  BLOCK1

        POP   DX
        POP   CX
        RET

BLOCK       ENDP
        END
```

Title 'Program 11-7'

```
;program that display a bright red B at row 0, column 0, and a
;cyan A at row 5, column 20.
        .MODEL TINY
        .CODE
        .STARTUP
        MOV    AX,0A000H      ;address video RAM at segment A000
        MOV    DS,AX
        CLD                ;select increment

        MOV    AX,12H          ;set mode to 12H
        INT    10H           ;and clear screen

        MOV    AL,'A'        ;display 'A'
```

```
        MOV    DL,3          ;cyan
        MOV    BX,5          ;row 5
        MOV    SI,20         ;column 0
        CALL   CHAR            ;display cyan 'A'

        MOV    AL,'B'        ;display 'B'
        MOV    DL,12          ;bright red
        MOV    BX,0          ;row 0
        MOV    SI,0         ;column 0
        CALL   CHAR            ;display bright red 'B'

        MOV    AH,1          ;wait for key
        INT    21H

        MOV    AX,3
        INT    10H          ;return to DOS video mode
        .EXIT
;
;The CHAR procedure displays a character (8 x 8) on the
;mode 12H display without changing the background color.
;AL = ASCII code
;DL = color (0 to 15)
;BX = row (0 to 52)
;SI = column (0 to 79)
;
CHAR   PROC   NEAR

        PUSH   CX
        PUSH   DX
        PUSH   BX             ;save row address
        PUSH   AX             ;save ASCII
        MOV    AX,1130H       ;get 8 x 8 set
        MOV    BH,3
        INT    10H          ;segment is in ES
        POP    AX            ;get ASCII code
        MOV    AH,0
        SHL    AX,1          ;multiply by 8
        SHL    AX,1
        SHL    AX,1
        ADD    BP,AX          ;index character in ROM
        POP    BX           ;get row address
                MOV    AX,80*9        ;find row address
        MUL    BX
        MOV    DI,AX
        ADD    DI,SI         ;add in column address
        MOV    CX,8          ;set count to 8 rows
C1:
        MOV   DX,3CEH        ;address bit mask register
        MOV    AL,8          ;load index 8
        MOV    AH,ES:[BP]     ;get character row
        INC    BP            ;point to next row
        OUT    DX,AX          ;modify bit mask register
        MOV    DX,3C4H        ;address map mask register
        MOV    AX,0F02H
        OUT    DX,AX          ;select all planes
```

```
        INC    DX
        MOV    AL,[DI]        ;read data
        MOV    BYTE PTR [DI],0 ;write black
        POP    AX             ;get color
        PUSH   AX
        OUT    DX,AL          ;write color
        MOV    BYTE PTR [DI],0FFH
        ADD    DI,80          ;address next raster row
        LOOP   C1             ;repeat 8 times

        POP    DX
        POP    CX
        RET

CHAR   ENDP
        END
```

Title 'Program 11-8'

```
;a program that displays two test lines of text on a cyan graphics
;background screen.
;
        .MODEL SMALL
        .DATA
MES1    DB      'This is test line 1.',0
MES2    DB      'This is test line 2.',0

        .CODE
        .STARTUP
        MOV     AX,0A000H       ;address video RAM
        MOV     DS,AX
        CLD                     ;select increment

        MOV     AX,12H                  ;set mode to 12H
        INT     10H             ;and clear screen

        MOV     DL,3            ;color cyan
        MOV     DH,53           ;row counter
        MOV     BX,0            ;row 0
MAIN1:
        MOV     CX,80           ;column counter
        MOV     SI,0            ;column 0
MAIN2:
        CALL    BLOCK                   ;display a cyan block
        INC     SI              ;address next column
        LOOP    MAIN2           ;repeat 80 times
        INC     BX              ;address next row
        DEC     DH              ;decrement row counter
        JNZ     MAIN1           ;repeat for 53 rows

        MOV     AX,@DATA        ;address data segment
        MOV     ES,AX           ;with ES

        MOV     DL,9            ;bright blue text
```

```
        MOV   BX,5              ;row 5
        MOV   SI,0              ;column 0
        MOV   DI,OFFSET MES1    ;address MES1
        CALL  LINE              ;display bright blue MES1

        MOV   DL,12             ;bright red
        MOV   BX,15             ;row 15
        MOV   SI,0              ;column 0
        MOV   DI,OFFSET MES2    ;address MES2
        CALL  LINE              ;display bright red MES2

        MOV   AH,1              ;wait for key
        INT   21H

        MOV   AX,3
        INT   10H               ;return to DOS video mode
        .EXIT
;
;The line procedure displays the line of text addressed by ES:DI
;DL = color of text (0 to 15).
;The text must be stored as a null string
;BX = row
;SI = column
;
LINE    PROC  NEAR

        MOV   AL,ES:[DI]        ;get character
        OR    AL,AL             ;test for null
        JZ    LINE1             ;if null
        PUSH  ES                ;save registers
        PUSH  DI
        PUSH  SI
        CALL  CHAR              ;display characters
        POP   SI                ;restore registers
        POP   DI
        POP   ES
        INC   SI                ;address next column
        INC   DI                ;address next character
        JMP   LINE              ;repeat until null
LINE1:
        RET

LINE    ENDP
;
;The CHAR procedure displays a character (8 x 8) on the
;mode 12H display without changing the background color.
;AL = ASCII code
;DL = color (0 to 15)
;BX = row (0 to 52)
;SI = column (0 to 79)
;
CHAR  PROC  NEAR

        PUSH  CX
        PUSH  DX
```

```
        PUSH  BX            ;save row address
        PUSH  AX            ;save ASCII
        MOV   AX,1130H      ;get 8 x 8 set
        MOV   BH,3
        INT   10H
        POP   AX            ;get ASCII code
        MOV   AH,0
        SHL   AX,1          ;multiply by 8
        SHL   AX,1
        SHL   AX,1
        ADD   BP,AX         ;index character in ROM
        POP   BX            ;get row address
        MOV   AX,80*9               ;find row address
        MUL   BX
        MOV   DI,AX
        ADD   DI,SI         ;add in column address
        MOV   CX,8          ;set count to 8 rows
C1:
        MOV   DX,3CEH               ;address bit mask register
        MOV   AL,8          ;load index 8
        MOV   AH,ES:[BP]    ;get character row
        INC   BP            ;point to next row
        OUT   DX,AX
        MOV   DX,3C4H               ;address map mask register
        MOV   AX,0F02H
        OUT   DX,AX         ;select all planes
        INC   DX
        MOV   AL,[DI]               ;read data
        MOV   BYTE PTR [DI],0       ;write black
        POP   AX            ;get color
        PUSH  AX
        OUT   DX,AL         ;write color
        MOV   BYTE PTR [DI],0FFH
        ADD   DI,80         ;address next raster row
        LOOP  C1            ;repeat 8 times
        POP   DX
        POP   CX
        RET

CHAR  ENDP
;
;The BLOCK procedure displays one block that is 8 pixels
;wide by 9 pixels high.
;BX = row address (0 to 52)
;SI = column address (0 to 79)
;DL = block color (0 to 15)
;
BLOCK       PROC   NEAR
        PUSH  CX
        PUSH  DX            ;save color

        MOV   DX,3CEH               ;graphics address register
        MOV   AL,8          ;select bit mask register
        OUT   DX,AL
        MOV   DX,3CFH               ;bit mask register
```

```
        MOV   AL,0FFH               ;enable all 8 bits
        OUT   DX,AL

        MOV   DX,3C4H               ;sequence address register
        MOV   AL,2          ;select map mask register
        OUT   DX,AL

        MOV   AX,80*9               ;find row address byte
        MUL   BX
        MOV   DI,AX         ;save it
        ADD   DI,SI         ;form address of PEL byte

        MOV   CX,9          ;byte count
        MOV   DX,3C5H               ;map mask register
        POP   AX            ;get color
        PUSH  AX
        MOV   AH,AL
BLOCK1:
        MOV   AL,0FH                ;enable all planes
        OUT   DX,AL
        MOV   AL,[DI]               ;must read first
        MOV   BYTE PTR [DI],0       ;clear old color
        MOV   AL,AH
        OUT   DX,AL
        MOV   BYTE PTR [DI],0FFH    ;write memory
        ADD   DI,80
        LOOP  BLOCK1

        POP   DX
        POP   CX
        RET

BLOCK       ENDP
END
```