Assembly Language for x86 Processors 7th Edition

Kip R. Irvine

Chapter 16: BIOS-Level Programming

Slide show prepared by the author

Revision date: 1/15/2014

Chapter Overview

- Introduction
- Keyboard Input with INT 16h
- VIDEO Programming with INT 10h
- Drawing Graphics Using INT 10h
- Memory-Mapped Graphics
- Mouse Programming

Personalities

- Bill Gates: co-authored QBASIC interpreter
- Gary Kildall: creator of CP/M-86 operating system
  - multitasking capabilities
- Peter Norton:
  - Inside the IBM-PC first book to thoroughly explore IBM-PC software and hardware
  - created the **Norton Utilities** software
- Michael Abrash: columnist, expert programmer
  - worked on Quake and Doom computer games
  - optimized graphics code in Windows NT
  - book: The Zen of Code Optimization

PC-BIOS

- The BIOS (Basic Input-Output System) provides low-level hardware drivers for the operating system.
  - accessible to 16-bit applications
  - written in assembly language, of course
  - source code published by IBM in early 1980's
- Advantages over MS-DOS:
  - permits graphics and color programming
  - faster I/O speeds
  - read mouse, serial port, parallel port
  - low-level disk access

BIOS Data Area

- Fixed-location data area at address 00400h
  - this area is also used by MS-DOS
  - this area is accessible under Windows 98 & Windows Me, but not under Windows NT, 2000, or XP.
- Contents:
  - Serial and parallel port addresses
  - Hardware list, memory size
  - Keyboard status flags, keyboard buffer pointers, keyboard buffer data
  - Video hardware configuration
  - Timer data

What's Next

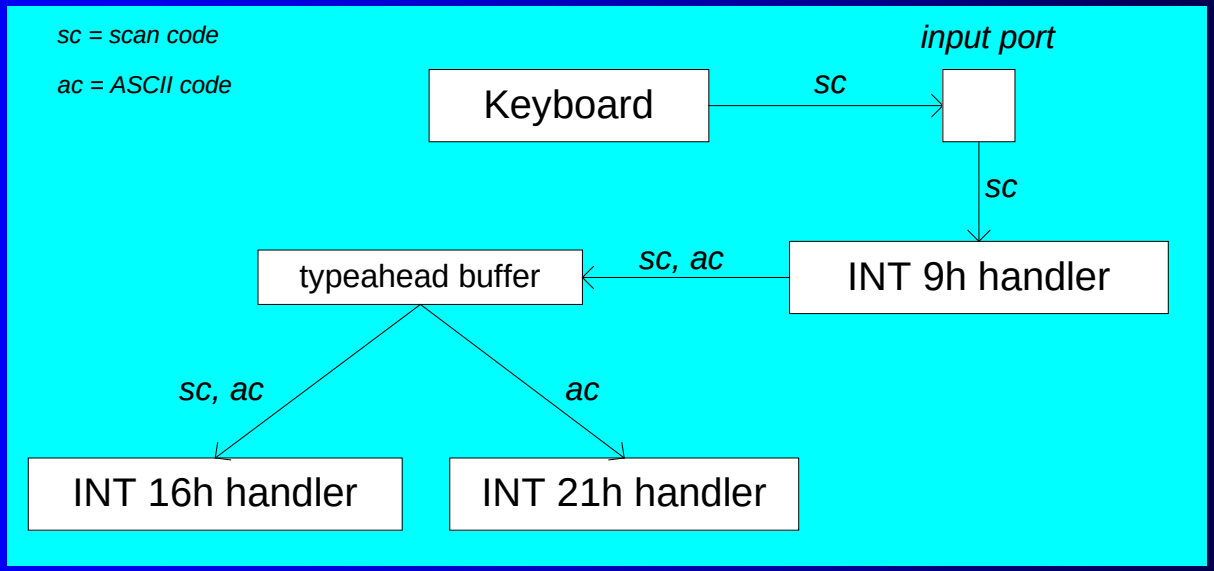
- Introduction
- **Keyboard Input with INT 16h**
- VIDEO Programming with INT 10h
- Drawing Graphics Using INT 10h
- Memory-Mapped Graphics
- Mouse Programming

Keyboard Input with INT 16h

- How the Keyboard Works
- INT 16h Functions

How the Keyboard Works

- Keystroke sends a scan code to the keyboard serial input port
- Interrupt triggered: INT 9h service routine executes
- Scan code and ASCII code inserted into keyboard typeahead buffer



Keyboard Flags

16-bits, located at 0040:0017h – 0018h.

Bit	Description
0	Right Shift key is down
1	Left Shift key is down
2	Either Ctrl key is down
3	Either Alt key is down
4	Scroll Lock toggle is on
5	Num Lock toggle is on
6	Caps Lock toggle is on
7	Insert toggle is on
8	Left Ctrl key is down

Bit	Description
9	Left Alt key is down
10	Right Ctrl key is down
11	Right Alt key is down
12	Scroll key is down
13	Num Lock key is down
14	Caps Lock key is down
15	SysReq key is down

INT 16h Functions

- Provide low-level access to the keyboard, more so than MS-DOS.
- Input-output cannot be redirected at the command prompt.
- Function number is always in the AH register
- Important functions:
  - set typematic rate
  - push key into buffer
  - wait for key
  - check keyboard buffer
  - get keyboard flags

Function 10h: Wait for Key

If a key is waiting in the buffer, the function returns it immediately. If no key is waiting, the program pauses (blocks), waiting for user input.

```
.data
scanCode  BYTE ?
ASCIICode BYTE ?

.code
mov ah,10h
int 16h
mov scanCode,ah
mov ASCIICode,al
```

Function 12h: Get Keyboard Flags

Retrieves a copy of the keyboard status flags from the BIOS data area.

```
.data
keyFlags WORD ?

.code
mov ah,12h
int 16h
mov keyFlags,ax
```

Clearing the Keyboard Buffer

Function 11h clears the Zero flag if a key is waiting in the keyboard typeahead buffer.

```
L1: mov ah,11h              ; check keyboard buffer
    int 16h                 ; any key pressed?
    jz  noKey               ; no: exit now
    mov ah,10h              ; yes: remove from buffer
    int 16h
    cmp ah,scanCode         ; was it the exit key?
    je  quit                ; yes: exit now (ZF=1)
    jmp L1                  ; no: check buffer again

noKey:                      ; no key pressed
    or  al,1                ; clear zero flag
quit:
```

What's Next

- Introduction
- Keyboard Input with INT 16h
- **VIDEO Programming with INT 10h**
- Drawing Graphics Using INT 10h
- Memory-Mapped Graphics
- Mouse Programming

VIDEO Programming with INT 10h

- Basic Background
- Controlling the Color
- INT 10h Video Functions
- Library Procedure Examples

Video Modes

- Graphics video modes
  - draw pixel by pixel
  - multiple colors
- Text video modes
  - character output, using hardware or software-based font table
  - mode 3 (color text) is the default
  - default range of 80 columns by 25 rows.
  - color attribute byte contains foreground and background colors

Three Levels of Video Access

- MS-DOS function calls
  - slow, but they work on any MS-DOS machine
  - I/O can be redirected
- BIOS function calls
  - medium-fast, work on nearly all MS-DOS-based machines
  - I/O cannot be redirected
- Direct memory-mapped video
  - fast – works only on 100% IBM-compatible computers
  - cannot be redirected
  - does not work under Windows NT, 2000, or XP

Controlling the Color

- Mix primary colors: red, yellow, blue
  - called subtractive mixing
  - add the intensity bit for 4th channel
- Examples:
  - red + green + blue = light gray (0111)
  - intensity + green + blue = white (1111)
  - green + blue = cyan (0011)
  - red + blue = magenta (0101)
- Attribute byte:
  - 4 MSB bits = background
  - 4 LSB bits = foreground

Constructing Attribute Bytes

- Color constants defined in Irvine32.inc and Irvine16.inc:

- Examples:
  - Light gray text on a blue background:
    - (blue SHL 4) OR lightGray
  - White text on a red background:
    - (red SHL 4) OR white

INT 10h Video Functions

- AH register contains the function number
- 00h: Set video mode
  - text modes listed in Table 15-5
  - graphics modes listed in Table 15-6
- 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

INT 10h Video Functions *(cont)*

- 09h: Write character and attribute
- 0Ah: Write character
- 10h (AL = 03h): Toggle blinking/intensity bit
- 0Fh: Get video mode
- 13h: Write string in teletype mode

Displaying a Color String

Write one character and attribute:

```
mov  si,OFFSET string
. . .
mov  ah,9                  ; write character/attribute
mov  al,[si]               ; character to display
mov  bh,0                  ; video page 0
mov  bl,color              ; attribute
or   bl,10000000b          ; set blink/intensity bit
mov  cx,1                  ; display it one time
int  10h
```

Gotoxy Procedure

```
;-----------------------------------------
Gotoxy PROC
;
; Sets the cursor position on video page 0.
; Receives: DH,DL = row, column
; Returns: nothing
;-----------------------------------------
   pusha
   mov ah,2
   mov bh,0
   int 10h
   popa
   ret
Gotoxy ENDP
```

Clrscr Procedure

```
Clrscr PROC
   pusha
   mov     ax,0600h          ; scroll window up
   mov     cx,0              ; upper left corner (0,0)
   mov     dx,184Fh          ; lower right corner (24,79)
   mov     bh,7              ; normal attribute
   int     10h               ; call BIOS
   mov     ah,2              ; locate cursor at 0,0
   mov     bh,0              ; video page 0
   mov     dx,0              ; row 0, column 0
   int     10h
   popa
   ret
Clrscr ENDP
```

What's Next

- Introduction
- Keyboard Input with INT 16h
- VIDEO Programming with INT 10h
- **Drawing Graphics Using INT 10h**
- Memory-Mapped Graphics
- Mouse Programming

Drawing Graphics Using INT 10h

- INT 10h Pixel-Related Functions
- DrawLine Program
- Cartesian Coordinates Program
- Converting Cartesian Coordinates to Screen Coordinates

INT 10h Pixel-Related Functions

- Slow performance
- Easy to program
- 0Ch: Write graphics pixel
- 0Dh: Read graphics pixel

DrawLine Program

- Draws a straight line, using INT 10h function calls
- Saves and restores current video mode
- Excerpt from the *DrawLine* program (DrawLine.asm):

```
mov ah,0Ch          ; write pixel
mov al,color        ; pixel color
mov bh,0            ; video page 0
mov cx,currentX
int 10h
```

Cartesian Coordinates Program

- Draws the X and Y axes of a Cartesian coordinate system
- Uses video mode 6A (800 x 600, 16 colors)
- Name: Pixel2.asm
- Important procedures:
  - DrawHorizLine
  - DrawVerticalLine

Converting Cartesian Coordinates to Screen Coordinates

- Screen coordinates place the origin (0,0) at the upper-left corner of the screen
- Graphing functions often need to display negative values
  - move origin point to the middle of the screen
- For Cartesian coordinates X, Y and origin points *sOrigX* and *sOrigY*, screen X and screen Y are calculated as:
  - *sx* = (*sOrigX* + X)
  - *sy* = (*sOrigY* – Y)

What's Next

- Introduction
- Keyboard Input with INT 16h
- VIDEO Programming with INT 10h
- Drawing Graphics Using INT 10h
- **Memory-Mapped Graphics**
- Mouse Programming

Memory-Mapped Graphics

- Binary values are written to video RAM
  - video adapter must use standard address
- Very fast performance
  - no BIOS or DOS routines to get in the way

Mode 13h: 320 X 200, 256 Colors

- Mode 13h graphics (320 X 200, 256 colors)
  - Fairly easy to program
  - read and write video adapter via IN and OUT instructions
  - pixel-mapping scheme (1 byte per pixel)

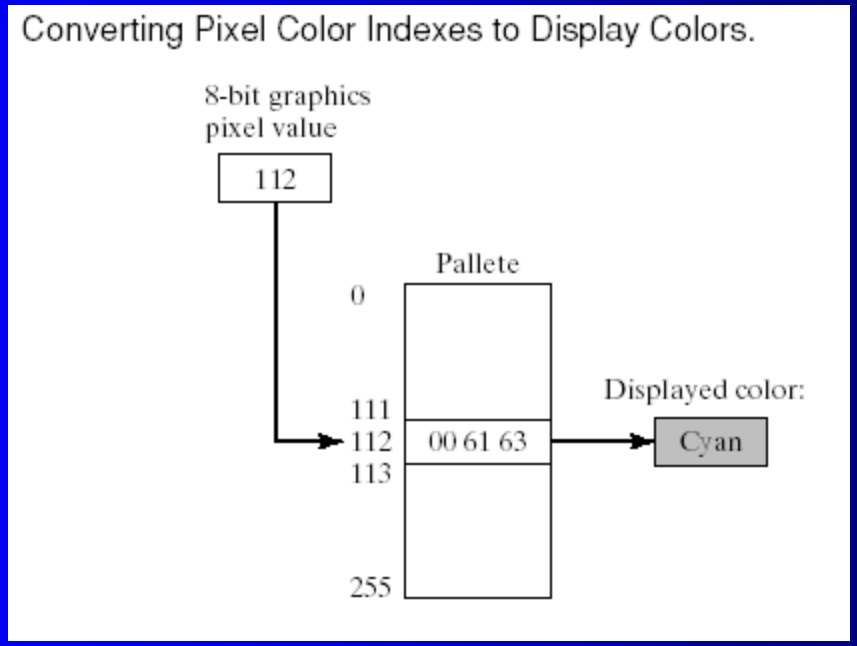
Mode 13h Details

- OUT Instruction
  - 16-bit port address assigned to DX register
  - output value in AL, AX, or EAX
  - Example:

```
mov  dx,3c8h          ; port address
mov  al,20h           ; value to be sent
out  dx,al            ; send to the port
```

- Color Indexes
  - color integer value is an index into a table of colors called a palette

Color Indexes in Mode 13h



RGB Colors

Additive mixing of light (red, green, blue). Intensities vary from 0 to 255.

Examples:

Red	Green	Blue	Color
0	30	30	cyan
30	30	0	yellow
30	0	30	magenta
40	0	63	lavender

Red	Green	Blue	Color
0	0	0	black
20	20	20	dark gray
35	35	35	medium gray
50	50	50	light gray
63	63	63	white

Red	Green	Blue	Color
63	0	0	bright red
10	0	0	dark red
30	0	0	medium red
63	40	40	pink

What's Next

- Introduction
- Keyboard Input with INT 16h
- VIDEO Programming with INT 10h
- Drawing Graphics Using INT 10h
- Memory-Mapped Graphics
- **Mouse Programming**

Mouse Programming

- MS-DOS functions for reading the mouse
- Mickey – unit of measurement (200th of an inch)
  - mickeys-to-pixels ratio (8 x 16) is variable
- INT 33h functions
- Mouse Tracking Program Example

Reset Mouse and Get Status

- INT 33h, AX = 0
- Example:

```
mov   ax,0
int   33h
cmp   ax,0
je    MouseNotAvailable
mov   numberOfButtons,bx
```

Show/Hide Mouse

- INT 33h, AX = 1 (show), AX = 2 (hide)
- Example:

```
mov   ax,1        ; show
int   33h
mov   ax,2        ; hide
int   33h
```

Get Mouse Position & Status

- INT 33h, AX = 4
- Example:

```
mov   ax,4
mov   cx,200        ; X-position
mov   dx,100        ; Y-position
int   33h
```

Get Button Press Information

- INT 33h, AX = 5
- Example:

```
mov   ax,5
mov   bx,0          ; button ID
int   33h
test  ax,1          ; left button down?
jz    skip          ; no - skip
mov   X_coord,cx    ; yes: save coordinates
mov   Y_coord,dx
```

Other Mouse Functions

- AX = 6: Get Button Release Information
- AX = 7: Set Horizontal Limits
- AX = 8: Set Vertical Limits

Mouse Tracking Program

- Tracks the movement of the text mouse cursor
- X and Y coordinates are continually updated in the lower-right corner of the screen
- When the user presses the left button, the mouse's position is displayed in the lower left corner of the screen
- Source code (c:\Irvine\Examples\ch15\mouse.asm)

Set Mouse Position

- INT 33h, AX = 3
- Example:

```
mov   ax,3
int   33h
test  bx,1
jne   Left_Button_Down
test  bx,2
jne   Right_Button_Down
test  bx,4
jne   Center_Button_Down
mov   Xcoord,cx
mov   yCoord,dx
```

Summary

- Working at the BIOS level gives you a high level of control over hardware
- Use INT 16h for keyboard control
- Use INT 10h for video text
- Use memory-mapped I/O for graphics
- Use INT 33h for the mouse

The End