OBJECTIVES

This chapter enables the student to:

• Code Assembly language instructions using INT 16H to get and check the keyboard input buffer and status bytes.
• Code Assembly language instructions for key press and detection.
• Use INT 33H to control mouse functions in text and graphics modes.
• Code Assembly language instructions to initialize the mouse and to set or get the mouse cursor position.
• Use INT 33H functions to retrieve mouse button press or release information.
• Limit mouse cursor positions by setting boundaries or defining exclusion areas.
5.1: INT 16H KEYBOARD PROGRAMMING

• The original IBM PC keyboard had 83 keys, in three major groupings:
  – 1. The standard typewriter keys.
  – 2. Ten function keys, F1 to F10.
  – 3. 15-key keypad.

• In later years, 101 key enhanced keyboards have become popular.
5.1: INT 16H KEYBOARD PROGRAMMING

keyboard scan codes

- Each key is associated with a scan code.

### Table 5-1: PC Scan Codes for 83 PC Keys

<table>
<thead>
<tr>
<th>Hex</th>
<th>Key</th>
<th>Hex</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Esc</td>
<td>17</td>
<td>I and i</td>
</tr>
<tr>
<td>02</td>
<td>! and 1</td>
<td>18</td>
<td>O and o</td>
</tr>
<tr>
<td>03</td>
<td>@ and 2</td>
<td>19</td>
<td>P and p</td>
</tr>
<tr>
<td>04</td>
<td># and 3</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>$ and 4</td>
<td>1B</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>% and 5</td>
<td>1C</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>^ and 6</td>
<td>1D</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>&amp; and 7</td>
<td>1E</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>* and 8</td>
<td>1F</td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>( and 9</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0B</td>
<td>) and 0</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>0C</td>
<td>and -</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>0D</td>
<td>+ and =</td>
<td>33</td>
<td>3</td>
</tr>
</tbody>
</table>

See the scan code tables on pages 162 - 163 of your textbook.
5.1: INT 16H KEYBOARD PROGRAMMING

• The same scan code is used for a given lowercase letter and its capital, and all keys with dual labels.
  – The keyboard shift status byte distinguishes the keys.
  • Some INT 16H function calls provide the status byte in AL.
  – For keyboard-motherboard, interaction IBM has provided INT 16H.

When a key is pressed, the OS stores its scan code in memory locations called a keyboard buffer, located in the BIOS data area.

<table>
<thead>
<tr>
<th>Bit</th>
<th>If = 1</th>
<th>Mask Code (OR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Right shift pressed</td>
<td>FEH</td>
</tr>
<tr>
<td>1</td>
<td>Left shift pressed</td>
<td>FDH</td>
</tr>
<tr>
<td>2</td>
<td>Ctrl pressed</td>
<td>FBH</td>
</tr>
<tr>
<td>3</td>
<td>Alt pressed</td>
<td>F7H</td>
</tr>
<tr>
<td>4</td>
<td>Scroll Lock toggled</td>
<td>EFH</td>
</tr>
<tr>
<td>5</td>
<td>NumLock toggled</td>
<td>DFH</td>
</tr>
<tr>
<td>6</td>
<td>CapsLock toggled</td>
<td>BFH</td>
</tr>
<tr>
<td>7</td>
<td>Ins toggled</td>
<td>7FH</td>
</tr>
</tbody>
</table>
For a program to run tasks continuously while checking for a keypress requires use of INT 16H. A BIOS interrupt used exclusively for the keyboard.

To check a keypress, use INT 16H function AH = 01.

```assembly
MOV AH, 0 ; get key pressed
INT 16H ; using INT 16H
```

- If ZF = 0, there is a key press.
- If ZF = 1, there is no key press.

This function does not wait for the user to press a key—it simply checks to see if there is a key press.

- If a character is available, it returns the scan code in AH, and the ASCII code in AL.
5.1: INT 16H KEYBOARD PROGRAMMING

checking a key press

• Program 5-1 sends the ASCII bell character, 07 hex to the screen continuously.

```assembly
.MODEL SMALL
.STACK
.DATA
MESSAGE DB 'TO STOP THE BELL SOUND PRESS ANY KEY$'
.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX
MOV AH,09
MOV DX,OFFSET MESSAGE ;DISPLAY THE MESSAGE
INT 21H
AGAIN: MOV AH,02 ;SENDING TO MONITOR A SINGLE CHAR
MOV DL,07 ;SEND OUT THE BELL CHAR
INT 21H
MOV AH,01 ;CHECK THE KEY PRESS
INT 16H ;USING INT 16H
JZ AGAIN ;IF NO KEY PRESS STAY IN THE LOOP
MOV AH,4CH ;IF ANY KEY PRESSED GO BACK TO DOS
INT 21H
MAIN ENDP
END
```

To stop the bell sound, the user must press any key.
5.1: INT 16H KEYBOARD PROGRAMMING

which key is pressed?

• INT 16H  AH = 0 determines the key pressed.
  – This function must be used immediately after AH = 01.

MOV AH, 0 ;get key pressed
INT 16H ;using INT 16H

  – AH = 0 doesn’t return until a key is pressed.
    • AH = 1 comes back whether or not a key has been pressed.
  – AL contains the ASCII character of the pressed key.
    • The scan key is in AH.
  – For characters such as F1–F10 for which there is no
    ASCII code, the scan code is in AH and AL = 0.
    • Thus, if AL = 0, a special function key was pressed.
5.1: INT 16H KEYBOARD PROGRAMMING

which key is pressed?

```assembly
.MODEL SMALL
.STACK
.DATA
MESSAGE DB 'TO STOP THE BELL SOUND PRESS Q (or q) KEYS'
.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX
MOV AH, 09
MOV DX, OFFSET MESSAGE ; DISPLAY THE MESSAGE
INT 21H
AGAIN: MOV AH, 02
MOV DL, 07 ; SOUND THE BELL BY SENDING OUT BELL CHAR
INT 21H
MOV AH, 01 ; CHECK FOR KEY PRESS
INT 16H ; USING INT 16H
JZ AGAIN ; IF NO KEY PRESS KEEP SOUNDING THE BELL
MOV AH, 0 ; TO GET THE CHARACTER
INT 16H ; WE MUST USE INT 16H ONE MORE TIME
CMP AL, 'Q' ; IS IT 'Q'?
JE EXIT ; IF YES EXIT
CMP AL, 'q' ; IS IT 'q'?
JE EXIT ; IF YES EXIT
JMP AGAIN ; NO, KEEP SOUNDING THE BELL
EXIT: MOV AH, 4CH ; GO BACK TO DOS
INT 21H
 MAIN ENDP
END
```

To stop the bell sound, the user must press a specific key.

Test for the correct keypress to stop the bell.
5.1: INT 16H KEYBOARD PROGRAMMING

other INT 16H functions

- **INT 16H, AH = 10H**
  (read a character) - the same as AH = 0, except that it also accepts the additional keys on the IBM extended (enhanced) keyboard.

- **INT 16H, AH = 11H**
  (find if a character is available) - the same as AH = 1, except that it also accepts the additional keys on the IBM extended (enhanced) keyboard.
5.2: MOUSE PROGRAMMING WITH INT 33H
detecting the presence of a mouse

- Because the original IBM PC & DOS did not provide support for the mouse, interrupt INT 33H is not part of BIOS or DOS.
  - INT 33H is part of the mouse driver software installed when the PC is booted.
- The first task of any INT 33H program should be to verify the presence of a mouse and the number of buttons it supports, using INT 33H function AX = 0.
  - On return from INT 33H, if AX = 0, no mouse is supported.
  - If AX = FFFFH, the mouse is supported and the number of mouse buttons will be contained in register BX.
5.2: MOUSE PROGRAMMING WITH INT 33H
detecting the presence of a mouse

- Although most mice have two buttons, right and left, there are some with middle buttons as well.

```assembly
MOV AX, 0 ; mouse initialization option
INT 33H
CMP AX, 0 ; check AX contents after INT 33H
JE EXIT ; exit if AX=0 since no mouse available
MOV M_BUTTON, BX ; mouse is there, save number of buttons
...
EXIT:
```

- In INT 21H & INT 10H, register AH is used to select functions — not the case in INT 33H.
  - AL is used to select various functions and AH is set to 0.
  - The reason for "MOV AX, 0".

Do not forget the "H", indicating hex. If absent, the compiler assumes it is decimal & executes INT 21H. (33 decimal = 21H)
5.2: MOUSE PROGRAMMING WITH INT 33H

some mouse terminology

• The mouse *pointer* (or cursor) is the pointer on the screen indicating where the mouse is pointing at a given time.
  – In graphics mode it is an arrow.
  – In text mode, a flashing block.

• As the mouse is moved, the mouse cursor is moved.
5.2: MOUSE PROGRAMMING WITH INT 33H

some mouse terminology

- While movement of the mouse is measured in inches (or centimeters), movement of the mouse cursor on the screen is measured in units called *mickeys*.
  - Mickey units indicate mouse sensitivity.

- A mouse that can move the cursor 200 units for every inch of mouse movement has a sensitivity of 200 mickeys.
  - In this case, one mickey represents 1/200 of an inch on the screen.
  - Some mice have a sensitivity of 400 mickeys in contrast to the commonly used 200 mickeys.
5.2: MOUSE PROGRAMMING WITH INT 33H
displaying and hiding the mouse cursor

- The AX = 01 function of INT 33H is used to display the mouse cursor.

```assembly
MOV AX, 01
INT 33H
```

- If the video mode is graphics, the mouse arrow is visible.
- If the video mode is text, a rectangular block representing the mouse cursor becomes visible.
  - The color of the mouse cursor block is the opposite of the background color in order to be visible.

- To hide the mouse cursor after making it visible, execute option AX = 02 of INT 33H.
5.2: MOUSE PROGRAMMING WITH INT 33H

video resolution vs. mouse resolution

- When the video mode is set to text mode, the mouse will automatically adopt the same resolution of 640 × 200 for its horizontal/vertical coordinates.
  - When a program gets the mouse cursor position, values are provided in pixels and must be divided by 8.
    - To get position in terms of character locations 0 to 79 (horizontal) and 0 to 24 (vertical) on the screen.
5.2: MOUSE PROGRAMMING WITH INT 33H

video resolution vs. mouse resolution

- In graphics modes, resolution is $640 \times 200$, $640 \times 350$ and $640 \times 480$.

<table>
<thead>
<tr>
<th>Video Mode</th>
<th>Video Resolution</th>
<th>Type</th>
<th>Mouse Resolution</th>
<th>Characters per Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL = 03</td>
<td>$640 \times 200$</td>
<td>Text</td>
<td>$640 \times 200$</td>
<td>$80 \times 25$</td>
</tr>
<tr>
<td>AL = 0EH</td>
<td>$640 \times 200$</td>
<td>Graphics</td>
<td>$640 \times 200$</td>
<td>$80 \times 25$</td>
</tr>
<tr>
<td>AL = 0FH</td>
<td>$640 \times 350$</td>
<td>Graphics</td>
<td>$640 \times 350$</td>
<td>$80 \times 44$</td>
</tr>
<tr>
<td>AL = 10H</td>
<td>$640 \times 350$</td>
<td>Graphics</td>
<td>$640 \times 350$</td>
<td>$80 \times 44$</td>
</tr>
<tr>
<td>AL = 11H</td>
<td>$640 \times 480$</td>
<td>Graphics</td>
<td>$640 \times 480$</td>
<td>$80 \times 60$</td>
</tr>
<tr>
<td>AL = 12H</td>
<td>$640 \times 480$</td>
<td>Graphics</td>
<td>$640 \times 480$</td>
<td>$80 \times 60$</td>
</tr>
</tbody>
</table>

The mouse also adopts these graphics resolutions.
5.2: MOUSE PROGRAMMING WITH INT 33H

getting the current mouse cursor position

- Option AX = 03 of INT 33H gets the current position of the mouse cursor.
  - On return, the X & Y coordinates are in registers CX (horizontal) and DX (vertical).
- BX contains the button status, 1 if *down*, 0 if *up*.
  - D0 = left button; D1 = right button; D2 = center button.
- The cursor position is given in pixels.
  - To get the mouse cursor character position, divide the horizontal and vertical values of CX & DX by 8.
- See Programs 5-3 & 5-4 on pages 168 - 171 of your textbook.
5.2: MOUSE PROGRAMMING WITH INT 33H
setting the mouse pointer position

• INT 33H option AX = 04 allows a program to set
  the mouse pointer to a new location anywhere
  on the screen.
    – Coordinates for the new location must be placed in
      CX for the horizontal (x coordinate) and DX for the
      vertical (y coordinate).

• Values must be in pixels.
    – In the range of 0–639 & 0–199 for 640 × 200 resolution.
      • Coordinate (0,0) is the upper left corner of the screen.
5.2: MOUSE PROGRAMMING WITH INT 33H
getting mouse button press information

• INT 33H option AX = 05 is used to get information about specific button presses since the last call to this function.

AX = 05
BX = 0 for left button; 1 for right button; 2 for center button
Upon return:
AX = button status where
D0 = Left button, if 1 it is down and if 0 it is up
D1 = Right button, if 1 it is down and if 0 it is up
D2 = Center button, if 1 it is down and if 0 it is up
BX = button press count
CX = x-coordinate at the last button press in pixels (horizontal)
DX = y-coordinate at the last button press in pixels (vertical)

Program 5-4 on pages 170 - 171 of your textbook shows one way to use this function.
5.2: MOUSE PROGRAMMING WITH INT 33H
the button press count program

• Program 5-5 on pages 172 - 173 uses the AX = 05 function to monitor the number of times the left button is pressed and then displays the count.
  - It prompts the user to press the left button a number of times.
    • When the user is ready to see how many times the button was pressed, any key can be pressed.
ENDS ; FIVE

Dec  Hex  Bin
5    5    00000101