CT215-9
Keyboard and Screen Processing

Objectives

- Interrupts
- Basic Keyboard Operations
  - **Reading a character from the keyboard**
  - **Reading a string of characters from the keyboard**
- Basic Screen Operations
  - **Reading and Setting the cursor**
  - **Screen Clearing and Scrolling**
  - **Screen display in text mode**

I/O Operations

- Input from: keyboard, disk, mouse
- Output to: screen, printer, disk
- In 80x86, input/output operations are handling by the BIOS (Basic Input Output System)
- Some I/O operations can be handled through DOS, which in turn uses the BIOS services

I/O Operations

- DOS I/O services are more “user friendly”
- BIOS I/O services are more efficient and provide more control
- I/O operations are done by calling DOS or BIOS INTerrupt routines (also called interrupt handlers or interrupt service procedures).

I/O Operations

[label:]	INT	interupt number

Example:

```
INT 10h ;BIOS
INT 21h ;DOS
```

Interrupts

- An (software) interrupt
  - **is a call to an interrupt routine --- the calling procedure is suspended and the control is transferred to the interrupt routine**
  - **is a procedure that performs one or a set of related functions.**
- All system interrupt routines are loaded in memory when the computer is booted up and remain in memory until shut down.

Interrupts

- Each interrupt routine is identified with a unique number form **`00-FF`**

  Example:
  - **`21h` is the interrupt handler for DOS services**
  - **`10h` is the BIOS interrupt handler for screen display**
- There are a total of 256 interrupt handlers, 0-31 reserved by Intel
- BIOS handles interrupts 00h-1Fh
- DOS handles interrupts 20h-3Fh

Interrupts

- When the system is powered up, an *interrupt vector table* is loaded in the lowest 1,024 bytes of memory (in location 0000h-3FFh)
- The interrupt vector table lists the address of each interrupt routine

Memory	**INT 00h CS:IP**	**INT 01h CS:IP**	**INT 02h CS:IP**	**INT 03h CS:IP**	**. . .**
Address	**00h**	**04h**	**08h**	**0Ch**	**. . .**

Interrupts

When an interrupt routine is invoked:

- the contents of the Flags register, CS, and IP are pushed on the stack
- The address of the interrupt routine is extracted from the vector table and the CS and IP registers are loaded.
- the interrupt routine is executed
- The interrupt routine returns via a IRET instruction which restores IP, CS, and the Flags registers.

Keyboard and Screen Operations

- **BIOS Interrupt Routines**
  - `INT 10h ;for screen operations`
  - `INT 16h ;for keyboard operations`
- **DOS `INT 21h`**
  - **for both screen and keyboard operations**
- **Each of the above interrupt routines performs several functions depending on the value stored in register AH at the time the interrupt is invoked**

Video Screen Operations

- Screen Display with **`INT 21h`**
  - functions 02h to display a character
  - function 09h to display a string
- Setting the cursor position on the screen
  - **`INT 10h ;with function 02h`**
- Clearing the screening and Scrolling
  - **`INT 10h ;with function 06h`**

Displaying a Character

Load 02h in AH

Load the character to be displayed in DL

Call `INT 21h`

Example:

```
CHAR  DB 'X'
      MOV AH,02h  ; Step 1
      MOV DL,CHAR ; Step 2
      INT 21h     ; Step 3
```

Displaying a String of Characters

```
String   DB   'Hello world', 13, 10
                ...
         LEA   DI,STRING1
         MOV   AH,02h
         MOV   CX,13
LP1:     MOV   DL,[DI]
         INT   21h
         INC   DI
         LOOP  LP1
                ...
```

Displaying a String of Character

- **Use INT 21h with function 09h:**
- **Define a string in the data area**
  - String must be ended with a '$'
- **Load 09h in register AH**
- **Load the offset address of the string in DX**
- **Call `INT 21h`**

Example

```
String  DB ‘Hello There’,13,10,‘$’ ;Step 1
            ...
        MOV AH,09H                 ;Step 2
        LEA DX,String              ;Step 3
        INT 21h                    ;Step 4
            ...
```

Setting the Cursor Position in the Text Mode

- **INT 10h with function 02h**
- **Display area provides for 4 pages: page0-page3**
- **Most software use page 0**
  - **Load 02h in AH**
  - **Load page # in BH (generally 0)**
  - **Load column number in DL**
  - **Load row number in DH**
  - **Call `INT 10h`**

Example

Set cursor at column 35, row 7

```
MOV AH,02H
MOV BH,00    ;set  page 0
MOV DL,35    ;column in DL
MOV DH,07    ;row in DH
INT 10H
```

Screen Clearing

INT 10h with function 06h

- Load 06h in AH
- Load 00h in AL (for clearing the full screen)
- Load attribute value in BH (color, blinking)
- Load starting position to scroll in CX (row:column)
  - row # in CH and column # in CL
- Load ending position in DX (row:column)
  - row # in DH and column # in DL

Example

Clear screen with white background and red foreground

```
MOV AX,0600h  ;AH=06h & AL=00h
MOV BH,74h    ;White background (7)
               red foreground(4)
MOV CX,0000h ;row 0 col 0
MOV DX,184Fh ;row 24 col 79(in
 Hex)
INT 10h
```

Scroll Up Screen

- **If a program display text past the bottom, the next line wraps around to start at the top**
- **Solution: scroll the full screen up by one line so that the displayed lines scroll off the top and blank line appear at the bottom**

Example

```
MOV AX,0607h  ;AH=06h,AL=07 (scroll
              up 7 lines)
MOV BH,74h    ;White background red
              foreground
MOV CX,0000h ;Scroll up the entire
              screen
MOV DX,184Fh
INT 10h
```

Clear Window

- **The window is 7 lines long (row 12 to row 18)**
  - set AL to 07
- **Need to clear the entire window**

```
CX = 0C19h      ; row 12 column 25
DX = 1236h      ; row 18 column 54
```

Example

```
MOV AX,0607  ;scroll up 7 lines
MOV BH,74    ;White background red
              foreground
MOV CX,0C19h ;From row 12,Column 25
MOV DX,1236h ;to row 18,column 54
INT 10h
```

Example

Scroll up 3 lines in [(12,25), (18, 54)]

```
MOV AX,0603  ;scroll up 3 lines
MOV BH,74    ;White background red
              foreground
MOV CX,0C19h ;From row 12,Column 25
MOV DX,1236h ;to row 18,column 54
INT 10h
```

Screed Display with File Handles

- **A *file handle* is a number that refers to a specific device such as keyboard, screen, printer or a disk file**
- **File handles are useful in case the output from a program is be redirected to a device other than the screen (e.g. a disk file) or if the input is to redirected from a device other than the keyboard**

File Handles

- **File handles are useful when reading from or writing to a disk file (Chapter 17)**
- **The file handles for standard devices such as the keyboard and the screen are preset.**

  The file handle for keyboard = 00

  The file handle for the screen = 01

Screed Display with File Handles

To display a string of character on the screen with file handles use INT 21h function 40h

- Set up string to be displayed as with function 09h
- Load AH with 40h
- Load DX with address of data to be displayed
- Load BX with 01 ; file handle for screen
- Load CX with number of characters to display
- Call INT 21h

Example

```
STRING1 DB  ‘Hello There’,0Dh ,0Ah ;Step 1
            ...
        MOV  AH,40h                ;Step 2
        LEA  DX,STRING1            ;Step 3
        MOV  BH,01                 ;Step 4
        MOV  CX,13                 ;Step 5
        INT  21h                   ;Step 6
```

Keyboard Operations

- **DOS INT 21h**

Read From Keyboard with DOS INT 21H

There are three ways to read the keyboard with DOS INT 21h:

- **INT 21h function 01h : Reads a character from keyboard.**
- **INT 21h function 0Ah : reads an entire line of characters from keyboard.**
- **INT 21h function 3Fh : same as 0Ah except it uses file handles**

Keyboard

- **Three Basic Types of Keys**
  - **Standard characters** (appendix B)
  - **Extended function key:**
    - Function keys: e.g. <F1>,< F2>, <SHIFT>+<F1>
    - Numeric Keypad with NumLock toggled off:
      <Home>, <END>, <Arrows>, <PgDn>
  - **No ASCII value but SCAN code** (appendix F)
  - **Control keys: <Alt>, <Ctrl>, and <Shift>, NumLock, CapsLock** (page 582)
    - No ASCII value.
    - Set the Shift status bytes in the BIOS

Keyboard Shift Status

- **BIOS keeps track of whether the control keys are pressed or not**
- **Shift Status Bytes: 2 bytes in the BIOS data area at addresses 40:17h and 40:18h.**
- **The bits in the Shift Status Bytes are set depending on which control keys are pressed or not.**
- **Example keys: CapsLock, NumLock, <ALT>, <CTRL>,<SHIFT>, etc (page 186)**

Keyboard Processing

Four important things

Shift status bytes -- two bytes stored in the BIOS data area. They enable a program to determine whether a control key is pressed or not

Scan code -- a unique number assigned to each key in the keyboard (appendix F page 579)

Keyboard Processing

Keyboard buffer -- Provides space in memory to store data typed on the keyboard. The buffer serves as temporary storage to hold input data until it is used by a program. The *keyboard buffer* is located in the BIOS data area (starting at address 40:1Eh) and can hold up to 15 characters.

Keyboard Processing

BIOS INT 09h -- the keyboard "watch dog"

- Whenever you press a keyboard key, the keyboard processor generates the key's scan code and requests INT 09h
- INT 09h reads the scan code, determines its ASCII character and delivers both to the keyboard buffer
- When a Control Key is pressed, INT 09h sets the shift status bytes accordingly (but does not deliver anything to the keyboard buffer)
- INT 09h is stored permanently in the ROM BIOS

Read a Character with Echo

INT 21h with function 01h :

- **Accept a character from the keyboard buffer. If buffer is empty wait for keyboard to be pressed.**
- **Operation returns one of two results:**
  - **If a standard ASCII character key is pressed, then AL is loaded with its ASCII code**
  - **If an extended function key is pressed (e.g. <Home>, <F1>, etc.), then AL is set to 0**
  - **Another call to INT 21h 01h will get the scan code of the control key**

Example

- **Write a procedure that reads one character from the keyboard (with echo) and stores it in AL**
- **If the pressed key is not a standard key, set the carry flag**

Procedure

```
KEY     PROC  FAR
        MOV   AH, 01
        INT   21h
        CMP   AL, 00h   ; test for 00h
        JNE   KEY1       ; jump if AL is not 00h
        INT   21h        ; get scan code if extended
        STC              ; set carry bit if extended
KEY1:   RET
KEY     ENDP
```

Reading a line of character from the keyboard

- **Reading a line of characters from the keyboard**

INT 21h function 0Ah

- **Need to give function 0Ah some parameters:**
  - Max Length of string
  - A place holder for the actual number of char read
  - A field in memory large enough to store the string

Example

```
ParaLst  Label Byte      ;Indicates start of
                         ;parameter list
Maxlen   DB 20           ;Max length is 20 char
Actlen   DB ?            ;set by INT 21h
Indata   DB 21 DUP(' ') ;space to store string
          . . .
         MOV AH,0AH      ;request function   0Ah
         LEA DX,PARALST
         INT 21h
```

Read a String from keyboard

- **INT 21h waits for the user to type in characters**
- **Typed characters are stored starting at address InData. Characters are also echoed to screen.**
- **Extended function keys are ignored.**
- **INT 21h terminates when <ENTER> key is hit.**
- **0Dh (carriage return) character is stored at end of string.**
- **INT21h stores the actual number of characters read in field Actlen (carriage return not counted)**

INT 21h with Function 3Fh

Read a string from keyboard with file handle

1. Set up a field in memory where string can be stored once it is read
2. Load Ah with 3Fh
3. Load DX with address of memory where string is to be stored
4. Load BX with 00; file handle for keyboard
5. Load CX with maximum length of string
6. Call INT 21h

Example

```
InData DB  20 DUP(‘ ‘)
           ...
       MOV AH,3Fh
       LEA DX,InData
       MOV BX,00
       MOV CX,20
       INT 21h
```

INT 21h with Function 3Fh

- **A successful INT operation clear the CF and sets AX to the actual number of characters read**
- **An unsuccessful INT operation sets the CF to 1 and sets the AX to an error code**

Summary

There are 3 ways to read the keyboard with DOS INT 21h:

1. **INT 21h function 01h -- Reads a key and echoes (displays) it on the screen**
2. **INT 21h function 0Ah -- reads an entire line of characters (with echo).**
3. **INT 21h function 3Fh -- same as 0Ah except it uses file handles**

INT 16h -- BIOS

Functions of INT 16h :

- Read one character from keyboard
  - **INT 16h function 10h** (enhanced keyboard)
- Determine whether a character is present in keyboard buffer
  - **INT 16h function 11h**
- Get the current keyboard shift status
  - **INT 16h function 12h**