CS/EEE/INSTR F241
Lab 8 – BIOS Interrupts for Display

Anubhav Elhence

DOS Interrupts and BIOS Interrupts

- In older computer systems, DOS and BIOS interrupts were used to interact with the hardware directly. Here are some ways to display data to the monitor using DOS and BIOS interrupts:
- DOS interrupt 21h: This interrupt provides many services, including printing characters to the screen. To print a string of characters to the screen, you can use the DOS function 09h.
- BIOS Interrupt 10h: This interrupt provides video services, including changing the video mode, scrolling the screen, and displaying characters on the screen. To display a character on the screen, you can use the BIOS function 0Ah.

BIOS Interrupt 10h

- There are four main aspects to Video Display
  - Setting an appropriate video mode (or resolution, as you know it)
  - Reading/Setting the cursor position
  - Reading/Writing an ASCII character at a given cursor position
  - Working at the pixel level on the screen (for e.g., drawing a line, square on the screen

- We can choose what action to perform by identifying the interrupt option type, which is the value stored in register AH and providing whatever extra information that the specific option requires. We shall only discuss the important interrupt types in the next section. However, additional interrupts are provided at the end for your benefit.

Follow Along Example 1

- This ALP sets the video mode to 720x400 with 256 colors, then continually reads keyboard input without echoing it to the screen until the user presses the '%' key. Once the '%' key is pressed, the program exits.

```
.model tiny
.data

.code
.startup

    MOV AH, 00H
    MOV AL, 12h
    INT 10H
    mov ah,07h
    x1: int 21h
    cmp al,'%'
    jnz x1



.exit
4 references
end
```

- MOV AH, 00H: Moves the value 00H (0) into the AH register. AH is the high byte of the AX register. In this case, it is used to set the video mode function in the BIOS interrupt 10H.
- MOV AL, 12h: Moves the value 12H (18) into the AL register. AL is the low byte of the AX register. In this case, it is used to set the video mode to 640x480, 16 colors.
- INT 10H: Calls the BIOS video interrupt 10H with the parameters in the AH and AL registers. This sets the video mode to 640x480, 16 colors as specified in the previous steps.
- mov ah,07h: Moves the value 07H (7) into the AH register. In this case, it is used to set the "read keyboard input without echo" function in the DOS interrupt 21H.
- x1: int 21h: Label "x1" is used as a loop start marker. Calls the DOS interrupt 21H with the parameters in the AH and AL registers. This reads a keyboard input without echoing it to the screen.
- `cmp al,'%': Compares the value in the AL register (the last key pressed) with the ASCII value of the character '%'. If the values are equal, the Zero Flag (ZF) is set, otherwise, it is cleared.
- jnz x1: Jumps back to the label "x1" if the Zero Flag is not set (i.e., the comparison in the previous step was not equal). This creates a loop that keeps reading keyboard input until the user presses the '%' key.

AL = 00H

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:  DOSBOX
AX=0725 BX=0000 CX=0012 DX=0000 SP=FFFE
BP=0000 SI=0000 DI=0000
DS=0863 ES=0863 SS=0863 CS=0863  IP=010E
NV UP EI PL ZR NA PE NC
0863:010E B44C                  MOV      AH,4
C
-
-q

D:\>
```

AL = 00H

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:  DEBUGX
0
0863:0102 B003                  MOV      AL,0
3
0863:0104 CD10                  INT      10
0863:0106 B407                  MOV      AH,0
7
0863:0108 CD21                  INT      21
0863:010A 3C25                  CMP      AL,2
5
0863:010C 75FA                  JNZ      0108

0863:010E B44C                  MOV      AH,4
C
0863:0110 CD21                  INT      21
0863:0112 A3D656                MOV      [56D
6],AX
0863:0115 8B169059              MOV      DX,[
5990]
0863:0119 2BF6                  SUB      SI,S
I
0863:011B 8B0ED458              MOV      CX,[
58D4]
0863:011F 8EC2                  MOV      ES,D
X
-
```

AL = 03H

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:  DEBUGX
AX=0725 BX=0000 CX=0012 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0863 ES=0863 SS=0863 CS=0863  IP=010E NV UP EI PL ZR NA PE NC
0863:010E B44C                  MOV      AH,4C
-
```

AL = 12H

```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:  DEBUGX
0863:0112 A3D656                MOV      [56D6],AX
0863:0115 8B169059              MOV      DX,[5990]
0863:0119 2BF6                  SUB      SI,SI
0863:011B 8B0ED458              MOV      CX,[58D4]
0863:011F 8EC2                  MOV      ES,DX
0863:0121 26894C2C              MOV      ES:[SI+2C],CX
0863:0125 50                    PUSH     AX
0863:0126 FF7606                PUSH     WORD PTR [BP+06]
0863:0129 B8F909                MOV      AX,09F9
0863:012C BA2D15                MOV      DX,152D
-
0863:012F 52                    PUSH     DX
0863:0130 50                    PUSH     AX
0863:0131 1E                    PUSH     DS
0863:0132 07                    POP      ES
0863:0133 9A42075A02            CALL     025A:0742
0863:0138 8BE5                  MOV      SP,BP
0863:013A 5D                    POP      BP
0863:013B CB                    RETF
0863:013C 55                    PUSH     BP
0863:013D 8BEC                  MOV      BP,SP
0863:013F FF7606                PUSH     WORD PTR [BP+06]
0863:0142 9A2D14EF06            CALL     06EF:142D
0863:0147 8BE5                  MOV      SP,BP
0863:0149 5D                    POP      BP
0863:014A CB                    RETF
0863:014B 55                    PUSH     BP
0863:014C 8BEC                  MOV      BP,SP
0863:014E 833E065702            CMP      WORD PTR [5706],+02
-
```


Anubhav Elhence and Dr. Vinay Chamola

Setting Cursor Position

- Input:

  AH = 02H

  DH = row.

  DL = column.

  BH = page number (0...7). Usually 0

week8_c2.asm > end

```
.model tiny
.data

.code
.startup

    MOV AH, 02H
    MOV DL, 0
    MOV DH, 0
    MOV BH, 0
    INT 10H


.exit
6 references
end
```

week8_c2.asm > ...

```
.model tiny
.data

.code
.startup

    MOV AH, 02H
    MOV DL, 40
    MOV DH, 0
    MOV BH, 0
    INT 10H


.exit
6 references
end
```


Anubhav Elhence and Dr. Vinay Chamola

Write character at Particular position

- Input:
  AH = 09h
  AL = character to display.
  BH = page number.
  BL = attribute.
  CX = number of times to write a character.

- Output:
  Character displayed at current cursor position CX number of times.

- **Attribute**
  The attribute byte is used to specify the foreground and background of the character displayed on the screen.

  Bits 2-1-0 represent the foreground colour
  Bit 3 represents the intensity of foreground colour (0-low , 1- high intensity)
  Bits 6-5-4 represent the background colour
  Bit 7 is used for blinking text if set to 1
  The 3 bit colour code (with their high intensity counterparts (if bit3 is 1) is

  - 000 -black (gray)
  - 001 -blue (bright blue)
  - 010 -green (bright green)
  - 011 -cyan (bright cyan)
  - 100 -red (bright red)
  - 101 -magenta (bright magenta)
  - 110 -brown (yellow)
  - 111 -white (bright white)

Write character at Particular position

```
.model tiny
.data

.code
.startup

    MOV AH, 09H
    MOV AL, '*'
    MOV BH, 00
    MOV BL, 10001001b
    MOV CX, 01
    INT 10H

    mov ah,07h
    x1: int 21h
    cmp al,'%'
    jnz x1
```

- Deciphering BL

$\underbrace{1}_{\text{blink}}\ \underbrace{000}_{\text{black}}\ \underbrace{1}_{\text{intensity}}\ \underbrace{001}_{\text{cyan}}$ || Please experiment

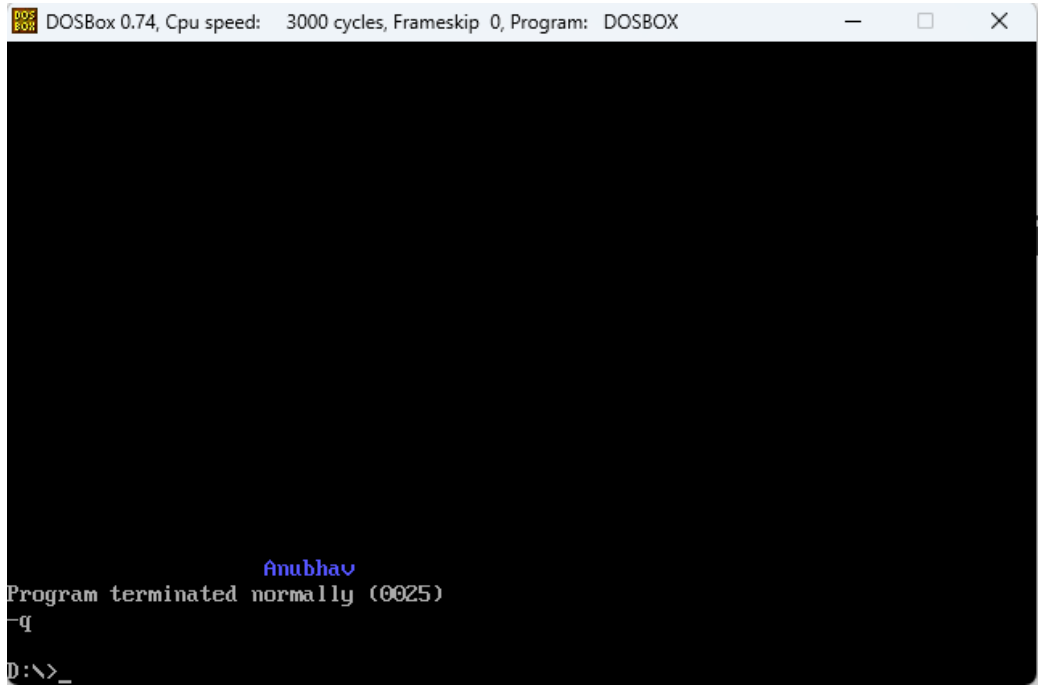
```
DOSBox 0.74, Cpu speed:   3000 cycles, Frameskip  0, Program:   DEBUGX
Object Modules [.obj]: week8_c3.obj/t
Run File [week8_c3.com]: "week8_c3.com"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

D:\>debugx WEEK8_C3.COM
-u
0863:0100 B409          MOV     AH,09
0863:0102 B02A          MOV     AL,2A
0863:0104 B700          MOV     BH,00
0863:0106 B389          MOV     BL,89
0863:0108 B90100        MOV     CX,0001
0863:010B CD10          INT     10
0863:010D B407          MOV     AH,07
0863:010F CD21          INT     21
0863:0111 3C25          CMP     AL,25
0863:0113 75FA          JNZ     010F
0863:0115 B44C          MOV     AH,4C
0863:0117 CD21          INT     21
0863:0119 2BF6          SUB     SI,SI
0863:011B 8B0ED458      MOV     CX,[58D4]
0863:011F 8EC2          MOV     ES,DX
-g 0115
*
```


Anubhav Elhence and Dr. Vinay Chamola

Follow Along Example:

- Example 1: Write your name at cursor position (20, 20) in blue blinking text with a black background. Use display mode 03H or Text VGA mode.



- As always, data initialization is the most important step

```
.model tiny ; Set memory model to tiny (code and data in one segment)
.386 ; Target 80386 processor

.data ; Data segment
5 references
inp1 db 'Anubhav' ; Define a byte array 'inp1' containing the input string 'MyName'
2 references
colmstr dw ? ; Define a word 'colmstr' to store the column position of the next character
1 reference
cnt db 07h ; Define a byte 'cnt' containing the length of the input string (6 characters)

.code ; Code segment
.startup ; Executable code starts here

```

- Setting the correct Display mode

```
    ; SET DISPLAY MODE
    ; Set video mode to 80x25 text, 16 colors
    MOV AH, 00H
    MOV AL, 03H
    INT 10H
```

- Initializing certain registers to begin writing the first character

```
; INITIALIZING
; Load the addresses of the input string, length counter, and column position into registers
LEA SI, inp1
LEA DI, cnt
MOV CH, 00h
MOV CL, [DI]
MOV colmstr, 20 ; Set initial column position to 20
LEA DI, colmstr

; WRITING CHAR
WRITE1:
PUSH CX ; Save count value on the stack
```

- Setting Cursor Position

```
; SETTING CURSOR POS
; Set the cursor position to row 20 and column specified by colmstr
MOV AH, 02H
MOV DH, 20
MOV DL, [DI]
MOV BH, 00
INT 10H
```


Anubhav Elhence and Dr. Vinay Chamola

Writing a single character at the specific cursor

```
; Write a single character with custom vertical spacing
MOV AH, 09H
MOV AL, [SI] ; Load character from input string
MOV BH, 00
MOV BL, 10001001b ; Set custom vertical spacing
MOV CX, 01
INT 10H
POP CX ; Restore count value from the stack
```

Changing Vertices

```
; CHANGING VERTICES
; Increment the input string pointer, column position,
; and decrement the length counter
INC SI
INC WORD PTR[DI]
DEC CL
JNZ WRITE1 ; Repeat for all characters in the input string
```

Blocking function

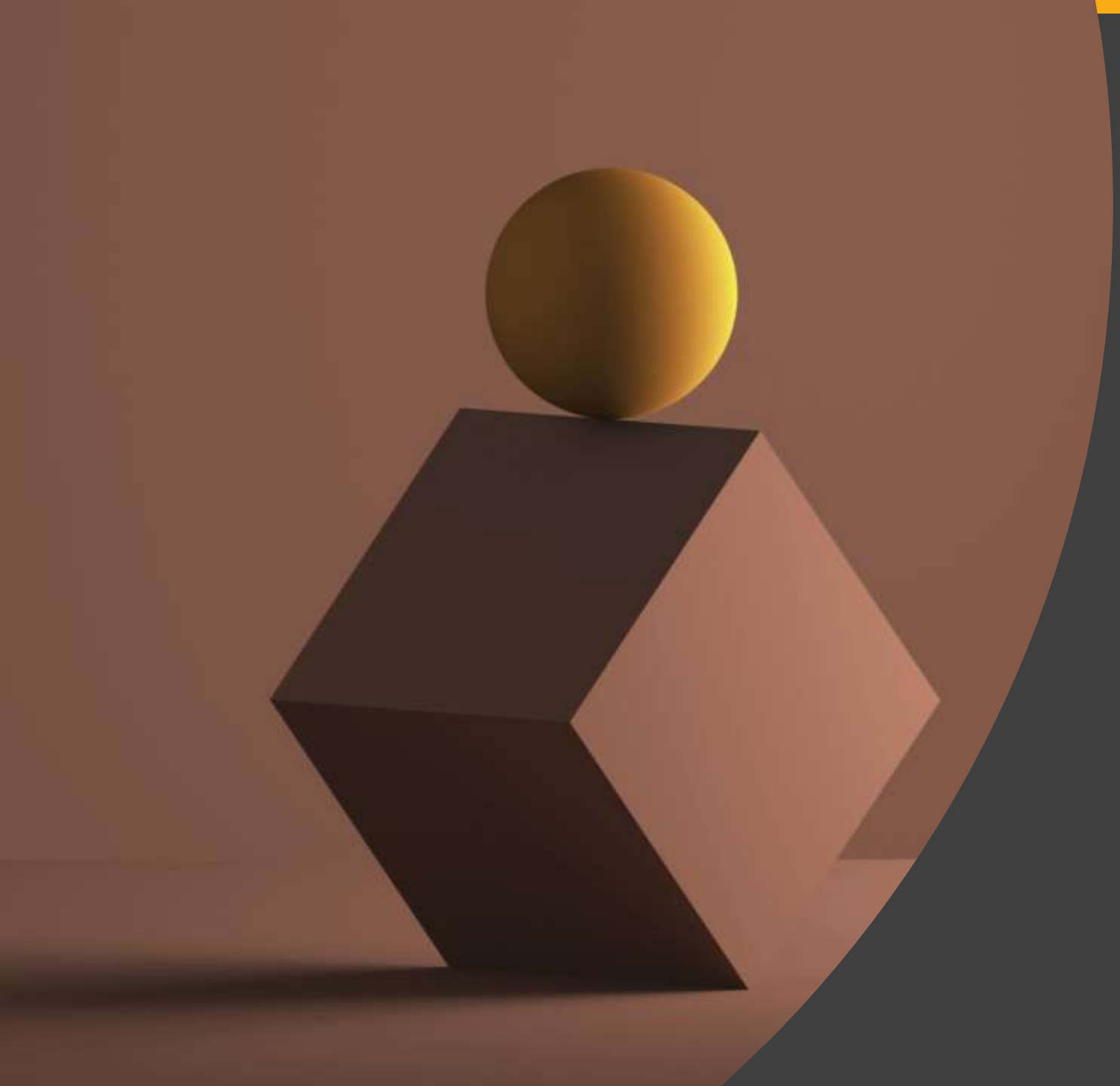
```
; BLOCKING FUNCTION
; Wait for the user to press the '%' key to exit
END1:
MOV AH, 07H
INT 21h
CMP AL, "%"
JNZ END1

; TERMINATE PROGRAM
TERM:
MOV AH, 4CH ; Exit function
INT 21H

.exit ; Mark the end of the program
```

Time for Lab Tasks:

Please check the description of this video.



Thankyou