Experiment #5

Using BIOS Services and DOS functions
Part 1: Text-based Graphics

5.0 Objectives:

The objective of this experiment is to introduce BIOS and DOS interrupt service routines to be utilized in assembly language programs.

In this experiment, you will use BIOS and DOS services to write programs that can do the following:

- Read a character/string from the keyboard
- Output a character/string to the display monitor
- Clear the display screen
- and display cursor at a desired location on the screen

5.1 Introduction:

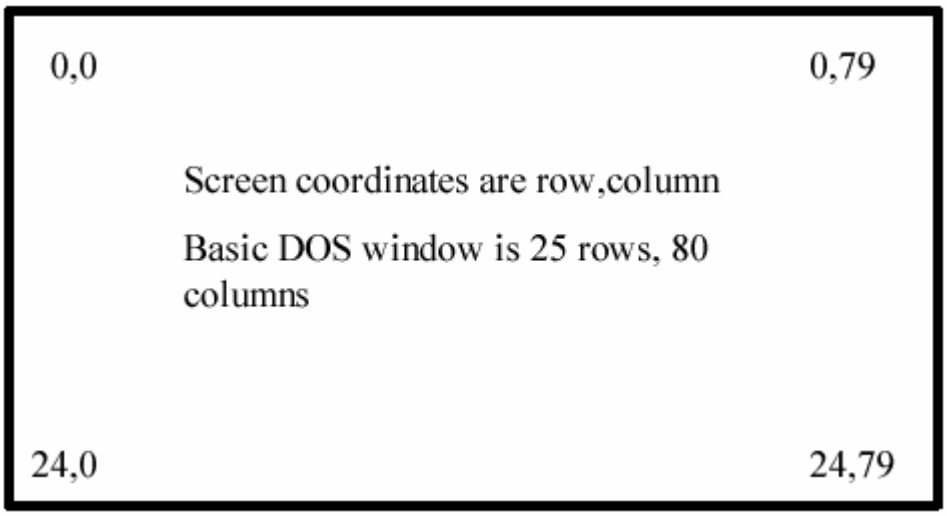
The Basic Input Output System (BIOS) is a set of x86 subroutines stored in Read-Only Memory (ROM) that can be used by any operating system (DOS, Windows, Linux, etc) for low-level input/output to various devices. Some of the services provided by BIOS are also provided by DOS. In fact, a large number of DOS services make use of BIOS services. There are different types of interrupts available which are divided into several categories as shown below:

Interrupt Types	Description
0h - 1Fh	BIOS Interrupts
20h - 3Fh	DOS Interrupts
40h - 7Fh	reserved
80h - F0h	ROM BASIC
F1h - FFh	not used

BIOS and DOS interrupt routines provide a number of services that can be used to write programs. These services include formatting disks, creating disk files, reading from or

writing to files, reading from keyboard, writing to display monitor, etc. The software interrupt instruction INT is used for calling these services.

5.1.1 Text Mode Programming



Positions on the screen are referenced using **(row, column)** coordinates. The upper left corner has coordinates (0,0). For an 80 x 25 display, the rows are 0-24 and the columns are 0-79.

5.1.2 Commonly used DOS functions

DOS contains many functions that can be accessed by other application programs. These functions are invoked using the assembly language instruction INT XX, where XX is replaced by the number of the appropriate interrupt. Most of the available functions are invoked through the INT 21H instruction.

Character input with echo (INT 21H, Function 01H):

Reads a character from the standard input device (usually the keyboard) and echoes it to the standard output device (usually the display screen), or waits until a character is available**.**

Description: (INT 21H, Function 01H)	**Example**

Invoked with: AH = 01H
Returns: AL = character input (ASCII code)
and displays the character on the screen
	MOV AH, 01H
	INT 21H
	MOV [SI],AL ; store char. in memory

Character input without echo (INT 21H, Function 07H):

Reads a character from the standard input device (usually the keyboard) **without** echoing it to the standard output device, or waits until a character is available**.** This function can be used when you don't want the input characters to appear on the display, for example, in the case of password entry.

Description: (INT 21H, Function 07H)	**Example**

Invoked with: AH = 07H
Returns: AL = character input (ASCII code)
	MOV AH, 07H
	INT 21H
	MOV [SI],AL ; store char. in memory

Display Character (INT 21H, Function 02H):

Displays a character at the standard output device (usually the display screen).

Description: (INT 21H, Function 02H)	**Example**

Invoked with: AH = 02H
DL = ASCII code for the char. to be displayed
Returns: Nothing
	MOV DL,'A' ; display character 'A'
	MOV AH, 02H
	INT 21H

Display Character String (INT 21H, Function 09H):

Displays a string of characters at the display screen. The string must be terminated with the character '$', which is not displayed.

Description: (INT 21H, Function 09H)	**Example**
Invoked with: AH = 09H DS : DX = segment : offset of string Returns: Nothing	MSG DB "Welcome",'$' ; string MOV DX, OFFSET MSG MOV AH, 09H INT 21H

Exit program and return control to DOS (INT 21H, Function 4CH):

Terminates current process and returns control either to the parent process or DOS.

Description: (INT 21H, Function 4CH)	**Example**
Invoked with: AH = 4CH AL = 00H Returns: Nothing	MOV AX, 4C00H INT 21H

5.1.3 BIOS Video I/O Services

The BIOS function requests in this category are used to control text and graphics on the PC's display screen. The function request is chosen by setting the AH register to the appropriate value and issuing interrupt 10H.

Set Video Mode (INT 10H, Function 00H):

Selects the video mode and clears the screen automatically.

Description: (INT 10H, Function 00H)	**Example**

Invoked with: AH = 00H
AL = mode number to indicate the desired video mode
Returns: Nothing
	MOV AH, 00
	MOV AL, 03H ; text video mode
	INT 10H

See section "**6.1.1 BIOS Video I/O Services"** for a list of video modes.

Set Cursor Position (INT 10H, Function 02H):

Sets the position of the display cursor by specifying the character coordinates.

Description: (INT 10H, Function 02H)	**Example**

Invoked with: AH = 2
BH = video page number (usually 0)
DH = row (0-24)
DL = column (0-79 for 80x25 display)
Returns: Nothing
	MOV AH, 02
	MOV BH, 0
	MOV DH, 12 ; row 12
	MOV DL, 40 ; column 40
	INT 10H

5.2 Pre-lab:

1. The following program allows a user to enter characters from the keyboard using the character input function (AH=01) of INT 21h. This program also stores the characters entered into a buffer. Run the program after assembling and linking it, and verify it by executing the program in the Turbo Debugger (TD).

Note: You have to create your source file in the directory where the TAMS.exe and TLINK.exe programs are stored.

```
TITLE  "Program to enter characters from keyboard"
.MODEL SMALL                    ; this defines the memory model
.STACK 100                      ; define a stack segment of 100 bytes
.DATA                           ; this is the data segment

        char_buf     DB   20 DUP(?) ; define a buffer of 20 bytes

.CODE                           ; this is the code segment

        MOV AX,@DATA            ; get the address of the data segment
        MOV DS, AX              ; and store it in register DS

        LEA SI, char_buf        ; load the address offset of buffer to store the name
        MOV AH, 01              ; DOS interrupt for character input from keyboard
AGAIN: INT 21H                  ; call the DOS interrupt

        MOV [SI], AL            ; store character in buffer
        INC SI                  ; point to next location in buffer
        CMP AL, 0DH             ; check if Carriage Return <CR> key was hit
        JNE AGAIN               ; if not <CR>, then continue input from keyboard

        MOV AX, 4C00H           ; Exit to DOS function
        INT 21H

END                             ; end of the program
```

Procedure (to be followed for all programs):

a. **Edit** the above program using an editor. Type "**edit program1.asm**" at the DOS prompt. Save your file and exit the editor. Make sure your file name has an extension of "**.asm**".

b. **Assemble** the program created in (a). Type "**tasm program1**" at the DOS prompt. If errors are reported on the screen, then note down the line number and error type from the listing on the screen. To fix the errors go back to step (a) to edit the source file. If no errors are reported, then go to step (c).

c. **Link** the object file created in (b). Type "**tlink program1**" at the DOS prompt. This creates an executable file "program1.exe".

d. Type "**program1**" at the DOS prompt to run your program.

2. Modify the above program such that the characters entered from the keyboard are not echoed back on the screen (i.e., they are not displayed when keys are pressed). [Hint: use function AH=07 with INT 21h]. After that, add the following lines of code between "**JNE AGAIN**" and MOV AX, 4C00H to display the characters stored in the buffer on the screen.

```
        LEA DI, char_buf          ; load the address offset of buffer to store the name
        MOV AH, 02                ; DOS interrupt for character output
BACK: MOV DL, [DI]                ; move character to be displayed in DL
        INT 21H                   ; call the DOS interrupt
        INC DI                    ; point to next location in buffer
        CMP BYTE PTR [DI], 0DH    ; check for 0Dh - ASCII value for ENTER key
        JNE BACK                  ; if not ENTER key, then continue output to screen
```

3. The following program clears the screen and positions the cursor at a specified location on the screen using INT 10H functions. The program also displays a message string on the screen using function 09h of INT 21H. Run the program after assembling and linking.

```
TITLE  "Program to enter characters from keyboard"
.MODEL SMALL                      ; this defines the memory model
.STACK 100                        ; define a stack segment of 100 bytes
.DATA                             ; this is the data segment

        LF      EQU   10      ; Line Feed character (0A in Hex)
        CR      EQU   13      ; Carriage Return character (0D in Hex)

        msg1      DB "WELCOME !!! ", LF, CR, "$"
        msg2       DB " EE 390 Lab, EE Department, KFUPM ", LF, CR, "$"

.CODE

MAIN PROC

        MOV AX,@DATA              ; get the address of the data segment
        MOV DS, AX                ; and store it in register DS

        CALL CLEARSCREEN          ; clear the screen

        MOV DH, 10                ; row 10
        MOV DL, 13                ; column 13
        CALL SETCURSOR            ; set cursor position

        LEA DX, msg1              ; load the address offset of message to be displayed
        MOV AH, 09h               ; use DOS interrupt service for string display
        INT 21H                   ; call the DOS interrupt

        MOV DH, 20                ; row 20
        MOV DL, 13                ; column 13
        CALL SETCURSOR            ; set cursor position

        LEA DX, msg2              ; load the address offset of message to be displayed
        MOV AH, 09h               ; use DOS interrupt service for string display
        INT 21H                   ; call the DOS interrupt

        MOV AX, 4C00H             ; exit to DOS
```

```
        INT 21H

MAIN ENDP
CLEARSCREEN PROC

        MOV AH, 00          ; set video mode
        MOV AL, 03          ; for text 80 x 25
        INT 10H             ; call the DOS interrupt
        RET                 ; return to main procedure

CLEARSCREEN ENDP

SETCURSOR PROC

        MOV AH, 2           ; use DOS interrupt service for positioning screen
        MOV BH, 0           ; video page (usually 0)
        INT 10H             ; call the DOS interrupt
        RET                 ; return to main procedure

SETCURSOR ENDP

END MAIN
```

Notes:

1. The above program uses three procedures – MAIN, SETCURSOR, and CLEARSCREEN. The SETCURSOR and CLEARSCREEN procedures are called from the MAIN procedure using the CALL instruction.
2. The SETCURSOR procedure sets the cursor at a specified location on the screen whereas the CLEARSCREEN procedure uses the SET MODE function 00H of INT 10H to set the video mode to 80 x 25 text which automatically clears the screen.
3. You can display a string of characters on the screen, without using a loop, by using MOV AH, 09 with INT 21h. But the string must end with '$' character. You must also load the effective address of the string in register DX.
4. To display a string on a new line, you need to put CR after your string and LF and '$' at the end. CR stands for Carriage Return (or Enter key) and LF stands for Line Feed. You can also put 0Dh or 13 instead of CR (or cr), and 0Ah or 10 instead of LF (or lf).

5.3 Lab Work:

The following program clears the screen and positions the cursor in the middle of the screen. Two memory locations 'row' and 'col' are used to keep track of the cursor position.

```
TITLE  "Program to move the cursor on the screen"
.MODEL SMALL                    ; this defines the memory model
.STACK 100                      ; define a stack segment of 100 bytes
.DATA                           ; this is the data segment

        row     DB      12      ; define initial row number
        col     DB      39      ; define initial column number

.CODE

MAIN PROC

        MOV AX,@DATA            ; get the address of the data segment
        MOV DS, AX              ; and store it in register DS

        CALL CLEARSCREEN        ; clear the screen

        CALL SETCURSOR          ; set the cursor position

        MOV AX, 4C00H           ; exit to DOS
        INT 21H

MAIN ENDP

CLEARSCREEN PROC

        MOV AH, 00              ; set video mode
        MOV AL, 03              ; for text 80 x 25
        INT 10H                 ; call the DOS interrupt
        RET                     ; return to main procedure

CLEARSCREEN ENDP

SETCURSOR PROC

        MOV DH, row             ; load row number
        MOV DL, col             ; load column number
        MOV AH, 2               ; use DOS interrupt service for positioning screen
        MOV BH, 0               ; video page (usually 0)
        INT 10H                 ; call the DOS interrupt
        RET                     ; return to main procedure

SETCURSOR ENDP

END MAIN
```

Note that the SETCURSOR procedure shown above gets its row and column positions directly from the memory variables 'row' and 'col'.

Modify the MAIN procedure in the above program to read an arrow key value from the keyboard using the DOS single character input function INT 21h, AH=7 which waits for a character and does not echo the character to the screen. Depending on which arrow key is pressed, the program must move the cursor accordingly, as indicated below:

Key pressed	ASCII value read from keyboard	Movement
↑ (Up)	48h	Move up (decrement row)
→ (Right)	4Dh	Move right (increment col)
↓ (Down)	50h	Move down (increment row)
← (Left)	4Bh	Move left (decrement col)

The following can be defined in the data segment:

```
LEFT    EQU  4Bh
RIGHT   EQU  4Dh
UP      EQU  48h
DOWN    EQU  50h
```

The following table shows some 80 x 25 screen positions.

Position	Decimal Value	Hexadecimal
Upper left corner	(0,0)	(0,0)
Lower left corner	(0,24)	(0,18)
Upper right corner	(79,0)	(4F,0)
Lower right corner	(79,24)	(4F,18)
Center screen	(39,12)	(27,C)

The program must wrap the cursor correctly around to the next boundary, for e.g., if the cursor moves off the right edge it should appear at the left edge and vice-versa. Similarly, if the cursor moves off the bottom edge it should appear at the top edge and vice-versa.

The program must continuously check for a key press (using the ASCII values given above) inside a loop, and move the cursor to a new position only when an arrow key is pressed. The program must exit the loop and return to DOS when the ENTER key (ASCII value 0Dh) is pressed.