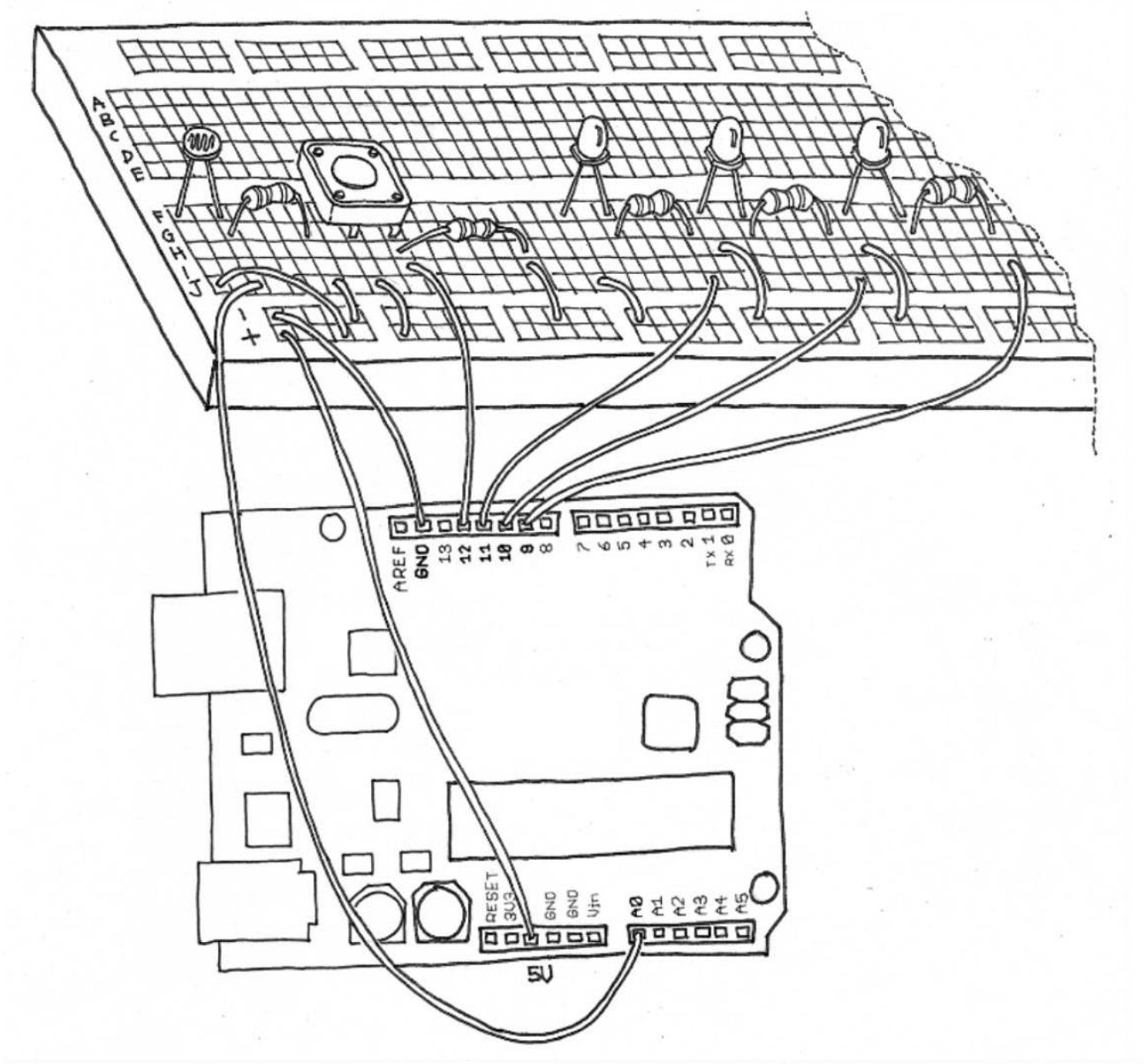
CS/EEE/INSTR F241
Microprocessor Programming and Interfacing

Lab 8 - BIOS Interrupts for Display



Dr. Vinay Chamola and Anubhav Elhence

BIOS Interrupts

BIOS interrupt calls are a facility that operating systems and application programs use to invoke the facilities of the Basic Input/Output System. DOS is one of the Operating Systems that BIOS can load. DOS and BIOS interrupts are used to perform some very useful functions, such as displaying data to the monitor, reading data from the keyboard, etc. BIOS function calls allow more control over the video display than do the DOS function calls.

DOS interrupts are executed using the command INT 21H which generates the software interrupt 0x21 (33 in decimal), causing the function pointed to by the 34th vector in the interrupt table to be executed, which is typically an MS-DOS API call.

Similarly, BIOS interrupts are executed using the command INT 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.

Purpose: Set Display mode

Input: AH = 0 AL = desired video mode

These video modes are supported:

00h - text mode. 40x25. 16 colours. 8 pages.

03h - text mode. 80x25. 16 colours. 8 pages.

12h - graphical mode. 80x25. 256 colours. 720x400 pixels. 1 page.

Note though 8 pages we always use only the first page

Output

Mode updated

Display cleared

Example: Program Segment to set video mode

```
4    .code
5    .startup
6
7        MOV AH, 00H
8        MOV AL, 12h
9        INT 10H
```

Notice how your display is visible only for a brief second and the program terminates when you do this.

To hold the display we use a Blocking Function.

So, before .exit statement we have to specify a blocking function

e.g. of a blocking function

```
4    .code
5    .startup
6
7        MOV AH, 00H
8        MOV AL, 12h
9        INT 10H
10       mov ah,07h
11       x1: int 21h
12       cmp al,'%'
13       jnz x1
```

System will then retain programmed display mode until the % key is pressed.

Purpose: Set cursor position.

Input:

AH = 02H

DH = row.

DL = column.

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

Example: Program Segment to set cursor position

```
4    .code
5    .startup
6
7        MOV AH, 02H
8        MOV DL, 40
9        MOV DH, 12
10       MOV BH, 0
11       INT 10H
```

Purpose: Write character at cursor 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)
```

Example Code:

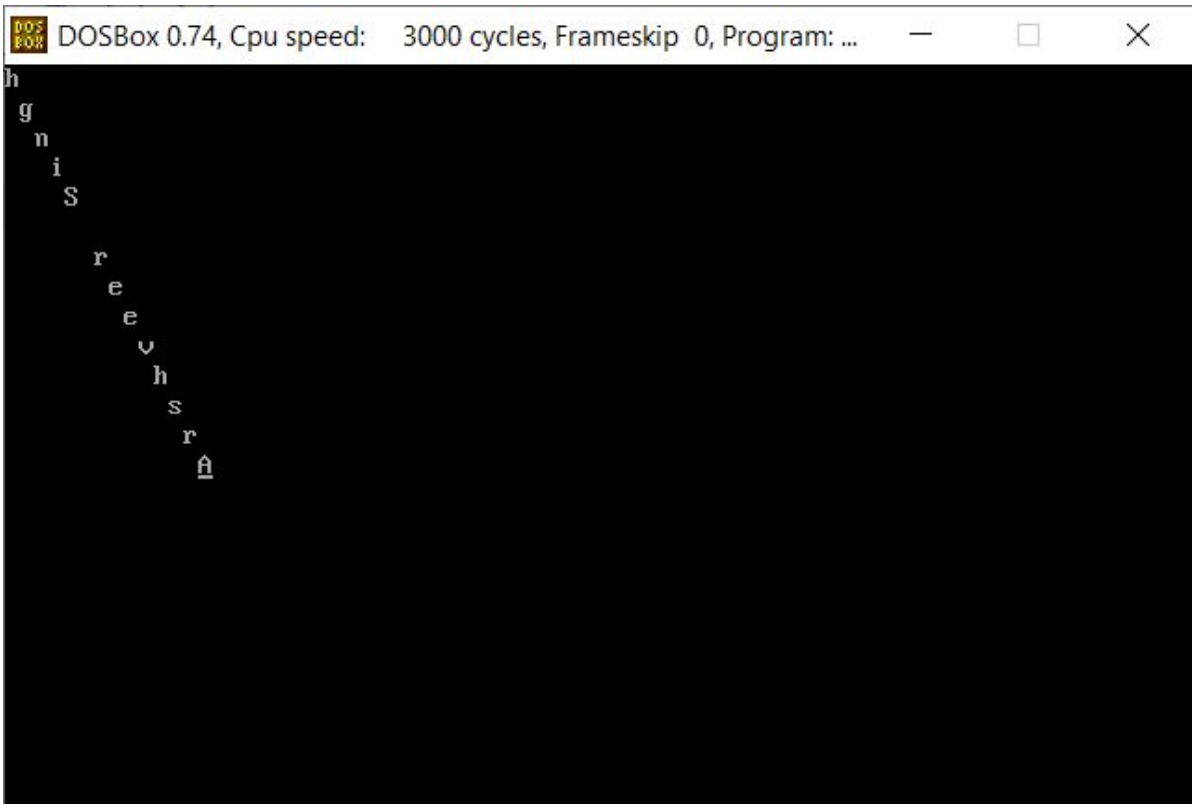
Example 1: Write your name at cursor position (20, 20) in blue blinking text with a black background. Use display mode 03H or Text VGA mode.

```
; This code is written in 16-bit x86 assembly language and uses BIOS interrupt calls for displaying a string
; with custom vertical spacing on the screen. It requires an assembler like NASM for compilation.

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

.data ; Data segment
5 references
inp1 db 'MyName' ; 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 06h ; Define a byte 'cnt' containing the length of the input string (6 characters)

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

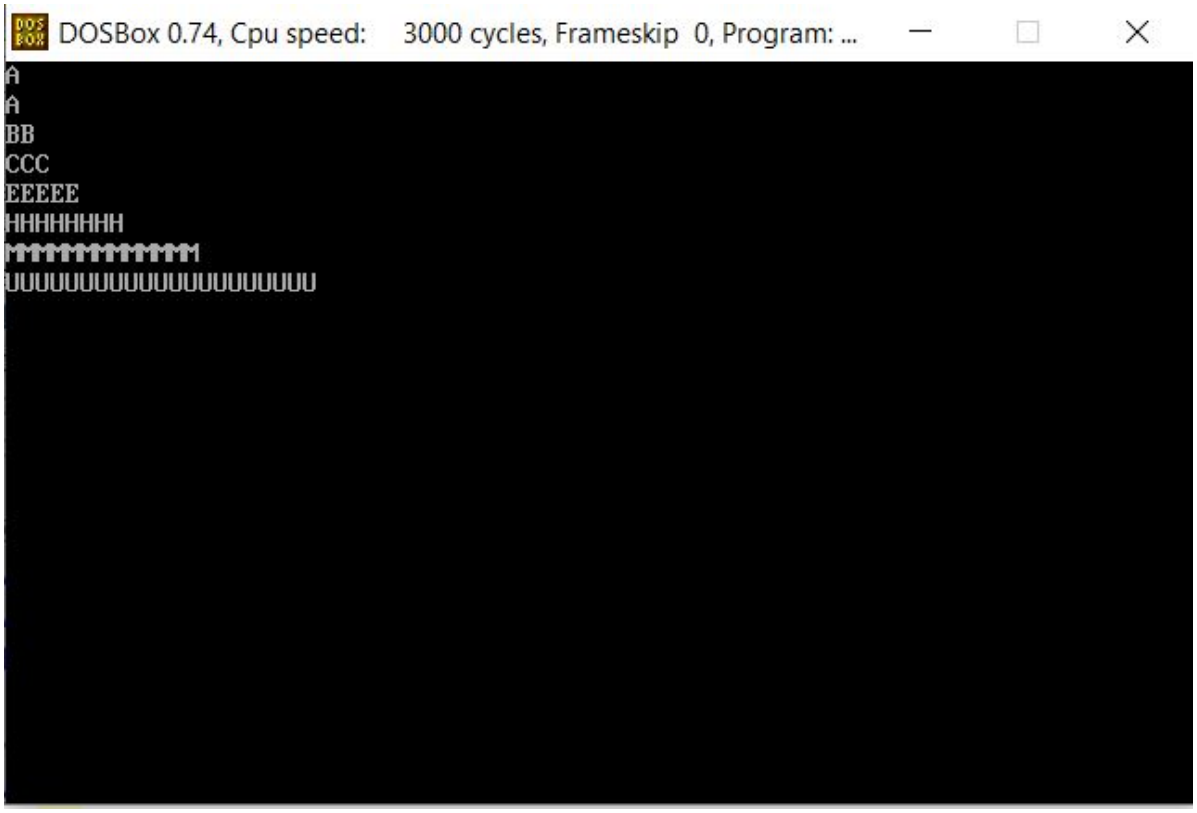
```
14
15      ; SET DISPLAY MODE
16      ; Set video mode to 80x25 text, 16 colors
17      MOV AH, 00H
18      MOV AL, 03H
19      INT 10H
20
21      ; INITIALIZING
22      ; Load the addresses of the input string, length counter, and column position into registers
23      LEA SI, inp1
24      LEA DI, cnt
25      MOV CH, 00h
26      MOV CL, [DI]
27      MOV colmstr, 20 ; Set initial column position to 20
28      LEA DI, colmstr
29
30      ; WRITING CHAR
31      WRITE1:
32      PUSH CX ; Save count value on the stack
33
34      ; SETTING CURSOR POS
35      ; Set the cursor position to row 20 and column specified by colmstr
36      MOV AH, 02H
37      MOV DH, 20
38      MOV DL, [DI]
39      MOV BH, 00
40      INT 10H
41
42      ; Write a single character with custom vertical spacing
43      MOV AH, 09H
44      MOV AL, [SI] ; Load character from input string
45      MOV BH, 00
46      MOV BL, 10001001b ; Set custom vertical spacing
47      MOV CX, 01
48      INT 10H
49      POP CX ; Restore count value from the stack
51        ; CHANGING VERTICES
52        ; Increment the input string pointer, column position, and decrement the length counter
53        INC SI
54        INC WORD PTR[DI]
55        DEC CL
56        JNZ WRITE1 ; Repeat for all characters in the input string
57
58        ; BLOCKING FUNCTION
59        ; Wait for the user to press the '%' key to exit
60        END1:
61        MOV AH, 07H
62        INT 21h
63        CMP AL, "%"
64        JNZ END1
65
66        ; TERMINATE PROGRAM
67        TERM:
68        MOV AH, 4CH ; Exit function
69        INT 21H
70
71  .exit ; Mark the end of the program
      4 references
72  end ; End the program
```

Lab Task

Task 1: Print your full name in reverse diagonal order in white text with a black background. Use display mode 03H or Text VGA mode. For example, if your name is Arshveer Singh it should look like



Task 2: Print the fibonacci sequence vertically where the nth term of the sequence must be repeated n times and must represent the nth term of the alphabet. Print the first 8 terms for convenience. Use the same specifications as given above. Sample output has been provided below:



// ADDITIONAL (Not required to solve tasks)

Purpose: Get Display mode

Input

AH=0Fh

Output

AL=current video mode

AH=number of character columns

BH=page number

Purpose: Get cursor position and size

Input

AH = 3

BH = page number (usually 0)

Output

DH = row.

DL = column.

CH = cursor start line.

CL = cursor bottom line.

Purpose: Set cursor size

Input:

AH = 01h

CH = cursor start line (bits 0-4) and options (bits 5-7).

CL = bottom cursor line (bits 0-4).

Output:

Cursor size changed.

Purpose: Read character at Cursor position

Input:

AH = 08h

Bh = 0 (page no.)

Output:

Error if CF = 1, AX = error code (6)

AH = attribute.

AL = character.


BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
ज्ञानं परमं बलम्