

Ministerul Educaţiei și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Report for laboratory work Nr. 1 for the course "Operational Systems"

Elaborated:

Felicia Lupascu, st. gr. FAF-212

Verified:

Rostislav Calin

Chişinău - 2023

**Subject:** Printing text using int 10H Video Services

**Tasks:**

1. ALL possible methods should be used in order to print text.

(update):

* M1: Write character as TTY
* M2: Write character
* M3: Write character/attribute
* M4: Display character + attribute
* M5: Display character + attribute & update cursor
* M6: Display string
* M7: Display string & update cursor
* M8(optional): Print directly to video memory

1. Compiled program should be used in order to create a floppy image and it should be bootable. Use this image to boot the OS in a VirtualBox VM and the text which you intended to print should appear on the screen.
2. You can use any assembly compiler.
3. Students should be able to modify the code, to recompile it and to boot the VM with a new version of the program.
4. In order to use documentation from TechHelp/XView DOS application, students can install DosBox.

**The result of performing the tasks: Method 1: Write character as TTY:**

go:

mov AH, 0Eh ; AH = 0Eh (BIOS function for Teletype Output)

; Set up AL register with the ASCII code of the character to be printed

mov AL, "R"

int 10h

; nasm -f bin -o first.bin first.asm

The program displays the character 'R' on the screen using the 0Eh function.

**The result:**

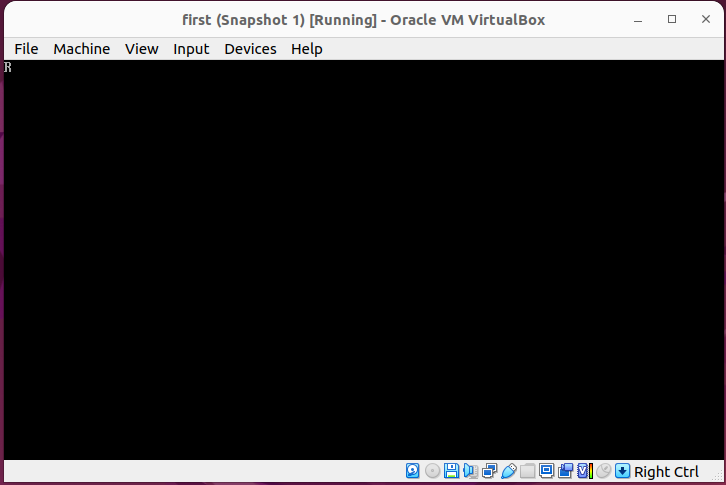


Figure 1 – Write character as TTY

**M2: Write character:**

go:

mov AH, 0Ah ; AH = 0Ah (BIOS function for Read String)

; Set up AL register with the default character to be displayed

mov AL, "R"

mov BH, 0

; Set up CX register with the maximum number of characters to read (including the initial character) in this case letter R 2 times

mov CX, 2

; Trigger BIOS interrupt 10h to perform Read String operation

int 10h

The second program writes character “R” using the 0Ah function. It also has a BH register to specify the video-page number and CX, which is the repeat count.

**The result:**

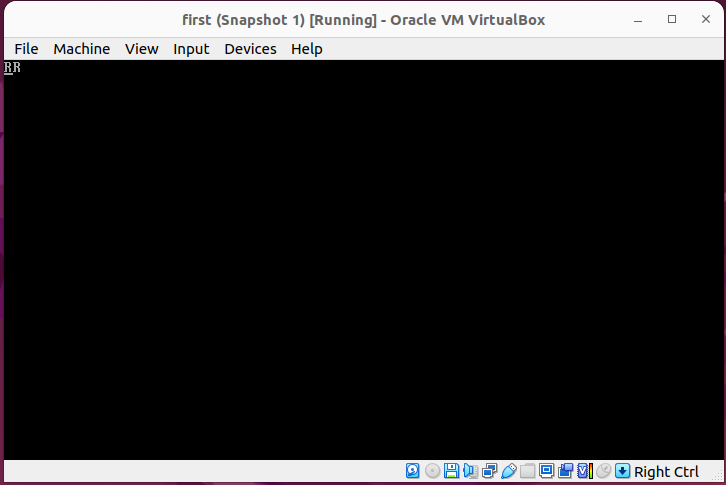


Figure 2 – Write character

**M3: Write character/attribute:**

go:

mov AH, 09h ; AH = 09h (BIOS function for Display String)

mov AL, "8" ; AL = ASCII code for the character '9'

mov BH, 0 ; BH = 0 (display on the default page)

mov BL, 0cH ; in this case is red color for background

mov CX, 3 ; CX = 3 (display the character '8' three times)

int 10h

; 0eH another colors

The third program writes the character “8” using the 09h function. It also has a BL register to specify the attribute for the character. For my code, it is the blue background color.

**The result:**

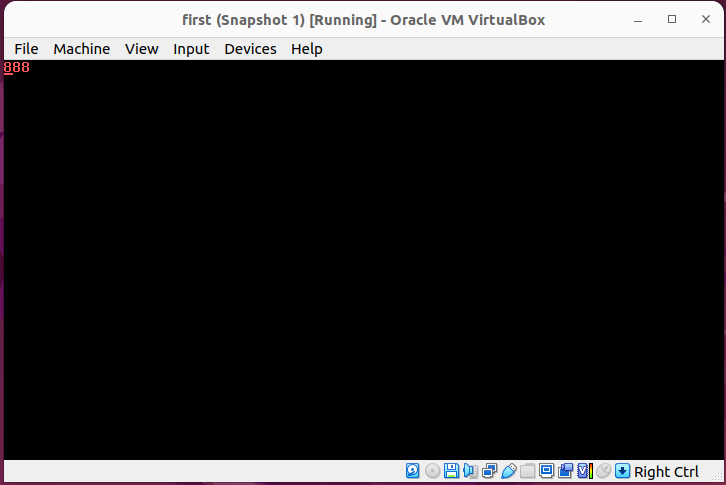


Figure 3 – Write character/attribute

**M4: Display character + attribute:**

org 7c00h

go:

mov BH, 0

mov AX, 0h

mov ES, AX

mov BP, message

mov AL, 1

mov CX, 5 ; Set CX register to 5, indicating the number of characters in the message.

mov DH, 5 ; Set DH register to 5, indicating the row position on the screen.

mov DL, 8 ; Set DL register to 8, indicating the column position on the screen.

mov BL, 8 ; Set BL register to 8, specifying the attribute for displaying the string (in this case, color).

mov AX, 1302h ; the BIOS function for printing a string with formatting.

int 10h

message db 'H', 09H, 'E', 0aH, 'L', 0bH, 'L', 0cH, 'O', 09H

**The result:**

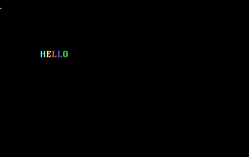


Figure 4 – Display character + attribute

**M5: Display character + attribute & update cursor:**

; M5: Display character + attribute & update cursor:

org 7c00H

go:

mov BH, 0

mov AX, 0h

mov ES, AX

mov BP, message

mov AL, 1

mov CX, 5

mov DH, 5

mov DL, 5

mov AX, 1303h ; Set AX register to 1303h (BIOS function for printing string with attribute & updating cursor)

int 10h

message db 'H', 0bH, 'E', 0eH, 'L', 0cH, 'L', 09H, 'O', 0aH

**The result:**

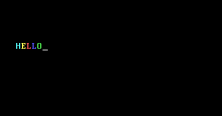


Figure 5 – Display character + attribute & update cursor

The programs 4 and 5 display the message “Hello” using the 1302h and 1303h functions, which are the functions to display character + attribute and differ by the fact that 1303h updates the cursor. For these functions it is necessary to enter ES:BP, which is the address of start of text to write. In the code, AX is initially cleared with mov AX, 0h, which sets ES to 0 by copying AX to ES. Afterward, BP is set to point to the address of the message 'RODICA'.

DH and DL registers are used to determine the row and column to start writing.

**M6: Display string:**

; M6: Display String

org 7c00h

go:

mov BH, 0

mov AX, 0h

mov ES, AX

mov BP, message ; Set BP register to the offset address of the message string

mov BL, 0fh ; Set BL register to 0fh, which represents the display attribute (text color)

mov AL, 1 ; Set AL register to 1 (display the string)

mov CX, 5 ; Set CX register to 5 (number of characters in the message)

mov DH, 5 ; Set DH register to 5 (row position on the screen)

mov DL, DH ; Set DL register to DH (column position on the screen)

mov AX, 1300h

int 10h

message dd "Hello"

**The result:**

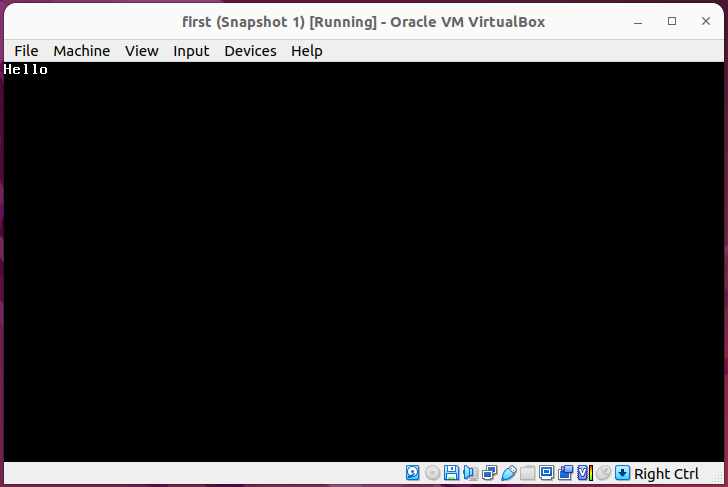
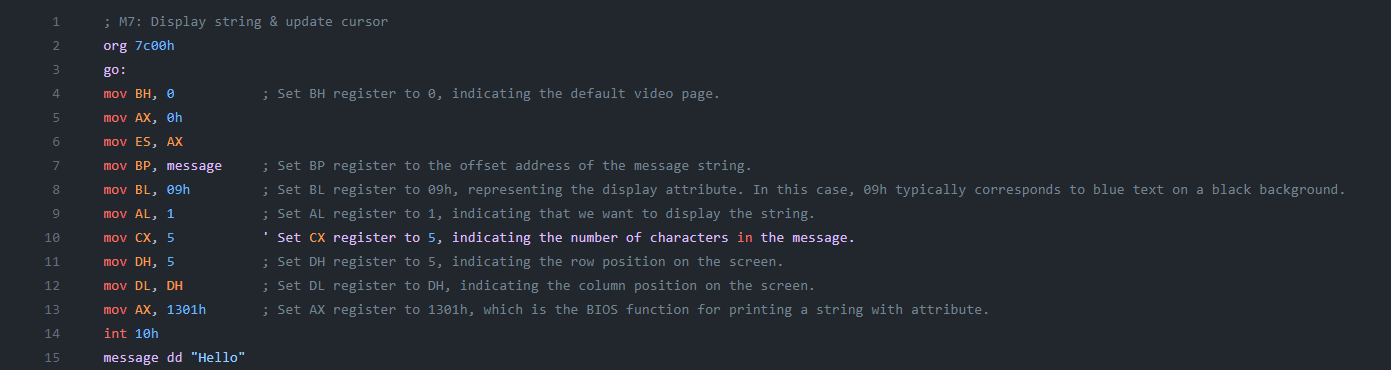


Figure 6 – Display string

**M7: Display string & update cursor:**

**The result:**

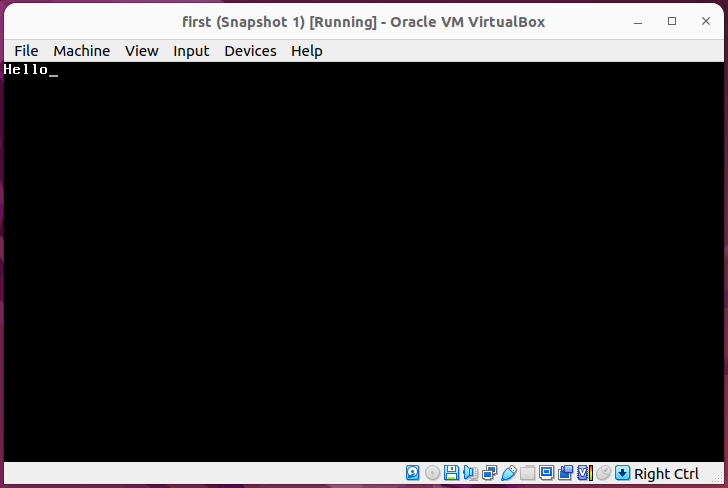


Figure 7 – Display string & update cursor

The programs 6 and 7 display string “Hello” using the 1300h and 1301h functions, differ by the fact that 1301h updates the cursor. For these functions it is necessary to enter ES:BP, which is the address of start of text to write. In the code, AX is initially cleared with mov AX, 0h, which sets ES to 0 by copying AX to ES. Afterward, BP is set to point to the address of the message 'Hello'.

DH and DL registers are used to determine the row and column to start writing.

**M8(optional): Print directly to video memory:**

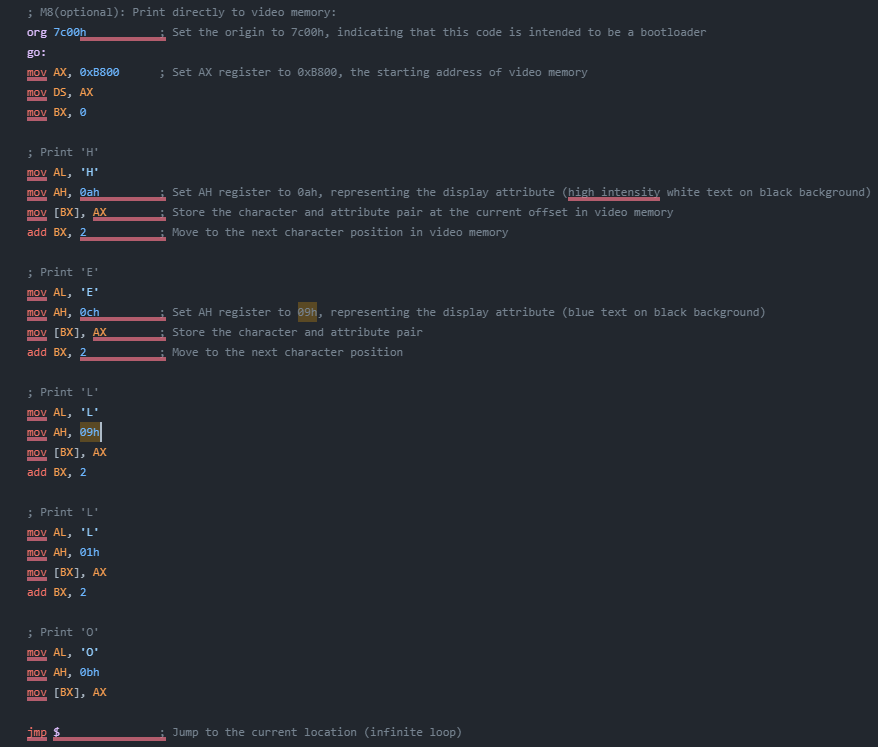
**The result:**



Figure 1

The program 8 prints the word “Hello” directly to video memory using the 0xB800 instruction. It sets the AX register to the video memory segment. Then, mov DS, AX sets the data segment (DS) to the video memory segment, making it the default segment for memory operations. After this setup, the code uses mov [BX], AX to write the content of the AX register (which holds the character and attribute) directly to the memory location pointed to by BX. The add BX, 2 instructions increment the BX register by 2, ensuring that each character is written to the next memory location in the video memory, allowing to display multiple characters on the screen.

**Compiling and getting floppy images:**

To get the floppy image it is necessary to compile the assembly code to get the bin file with the command:

nasm -f bin -o prog4.bin prog4.asm

After that we have to ensure that the file is exactly 1474560 bytes and to convert it into an image. This floppy image we can pass in the VirtualBox to the floppy controller and see the result.

**Conclusion:**

In conclusion, this laboratory work provided me with a comprehensive understanding of assembly functions related to character and string display. Through hands-on exploration, I gained proficiency in manipulating text attributes such as color, position, and cursor updates. Additionally, I acquired practical skills in printing characters directly to video memory and successfully executed floppy disk images in VirtualBox, enhancing my overall grasp of low-level programming concepts and their real-world applications.