Ministry of Education and Investigation Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

REPORT

Laboratory work nr.1

on the course “Operating Systems”

Executed by:

st. gr. FAF-212 Ciumac Alexei

Verified by:

prof. univ. Rostislav Calin

Chişinău - 2023

Topic: displaying the characters and strings using int 10h BIOS interruptions;

Tasks:

- Write character as TTY;
- Write character;
- Write character/attribute;
- Display character/attribute + update cursor;
- Display string;
- Display string + update cursor;
- Print directly to video memory;

Implementation and results:

First of all we need to compile the code using NASM in this way:

```
nasm -f bin lab1.asm -o lab1.bin
```

After that code on python make it into a bootable image

```
import os

# Input and output file names
input_file_name = 'lab1.bin'
output_file_name = 'labImg1.img'

# Get the size of the input file
input_file_size = os.path.getsize(input_file_name)

# Read the contents of the input file
with open(input_file_name, 'rb') as input_file:
    file_content = input_file.read()

# Create a new file and write the content of the input file
with open(output_file_name, 'wb') as output_file:
    output_file.write(file_content)
```

```
# Calculate the number of zero bytes to fill
remaining_size = 1474560 - input_file_size

# Fill the rest of the file with zero bytes
with open(output_file_name, 'ab') as output_file:
    output_file.write(b'\x00' * remaining_size)

print(f"File '{output_file_name}' created with size 1474560
bytes.")
```

With the floppy image created in the same folder it can be inserted into the floppy controller of the virtual machine (Virtual Box).

The first six tasks were implemented in a single file. After some of the instructions it was necessary to move the cursor explicitly in order to avoid the character overlap.

```
org 7c00h
; nasm -f bin lab1.asm -o  lab1.bin
section .data
  mystr db "START!", 0
  let dd 'S'
section .text
  global _start


_start:
  ; printing letter 'H'
  ; TTY output \ update cursor position
  mov ah, 0eh
  mov al, 72 ; ascii code for the letter 'H'
  mov bl, 0x08 ; white foreground color attribute
  int 10h
```

```
 ; printing letter 'I'
 mov ah, 0ah
 mov al, 73 ; ascii code for the letter 'I'
 mov bl, 0x09
 int 10h


 ; move cursor
mov ah, 02h;
mov dh, 0x00;
mov dl, 0x04; column
int 10h;


 ; write char/attribute
mov ah, 09h;
mov al, 69; E - ascii number
mov bl, 0x02; red color attribute
mov cx, 1;
int 10h;


; move cursor
mov ah, 02h;
mov dh, 0x05;
mov dl, dh;
int 10h;


 ; Use di register to point to the destination offset
 lea di, [let] ; lea instruction that performs memory
addressing calculations but doesn't actually address memory.
```

```
  ; Load the character 'S' from the memory location pointed
by di
  mov al, [di]
  ; Call interrupt 10h to perform video services
  mov ax, 1302h
  int 10h


 ; move cursor
 mov ah, 02h;
 mov dh, 0x07;
 mov dl, dh;
 int 10h;


  ; Use dj register to point to the destination offset
  lea di, [let]


  ; Load the character 'S' from the memory location pointed
by di
  mov al, [di]
  ; Call interrupt 10h to perform video services
 mov ax, 1303h
 int 10h;


  ; printing the string "START!"
  mov ah, 13h
  mov al, 0 ; attribute
  mov bh, 0 ; page number
  mov cx, 6 ; number of characters
  mov dh, 0x03 ; row
  mov dl, dh; column
```

```
 mov dx, 14 ; column and row position
 mov bp, mystr ; pointer to the string
 int 10h


 ; display string + update cursor
mov ax, 0h;
mov es, ax;
mov bl, 0x03; cyan color attribute
mov cx, 0x06; length of string
mov dh, 0x10; row to start writing
mov dl, dh; column to start writing
mov bp, mystr;
mov ax, 1301h;
int 10h;
```

Here we have some screenshots of the results of code:

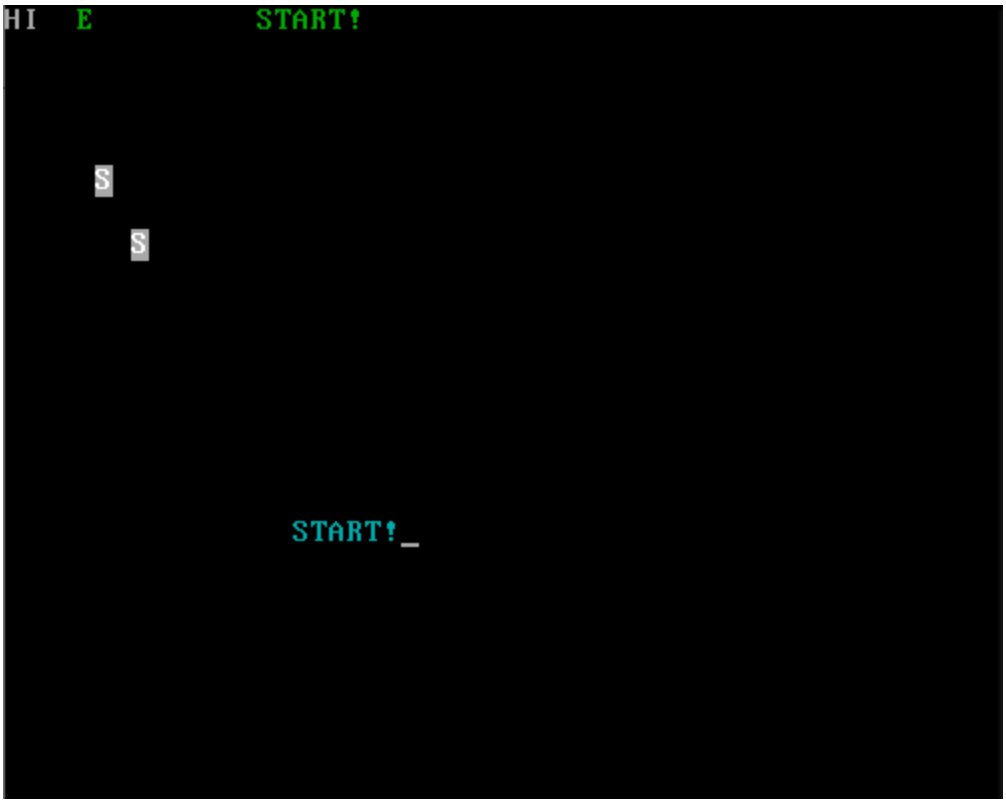

Figure 1 - Output of the first seven tasks

And the advanced task together with its output will be presented below:

```
org 7c00h
; nasm -f bin lab1a.asm -o  lab1a.bin
section .text
 global _start


_start:
 mov ax, 0xB800; address of the Video memory
 mov es, ax;
```

```
 xor di, di; offset to write characters to video memory
pointer

 mov ax, 'B';
 stosb; write the character to the memory
 mov   ax, 0x04; text color
 stosb; write the attribute to the memory
```

Figure 2 - Output of the advanced task

Conclusions:

During this lab assignment, I delved into the intricacies of the int 10h BIOS video services and their associated interrupts. These services offer diverse methods for displaying characters or strings on the screen; however, their efficiency for real-world tasks is notably sluggish. The exploration led me to realize that for improved performance, a more viable approach involves directly writing characters to video memory. Furthermore, I gained insights into the critical importance of instructing the program about the initial memory segment to work with, minimizing the risk of memory errors from the outset. Also I understood that comments in th code are very useful as for me as for people who is looking through.