Ministry of Education of Republic of Moldova

Technical University of Moldova

Department of Software engineering and Automatics

Report

Laboratory Work No.4

on SOMMIP

Performed by : Zaharia Gabriel

Checked by: Rostislav Calin

Chisinau, 2018

Objectives:

- CLI application with functionalities
- Ability to work with interrupts
- Processing the user input string
- Key interrupts
- Provide the same experience as in windows CLI

Tasks:

1. CLI should have an INVITATION/PROMPT at the beginning of the line and it should be formed by 1 to 3 symbols. Prompter symbols should be in 2-3 different colors.

2. Cursor after prompter should be visible and could have any form (blinking or not; underscore or solid block)

3. User should be able to type some string from the keyboard and CLI must print on the screen all alphanumeric symbols, one by one. If cursor position will reach last column than printing should continue from the next line. Typed string should be collected into the memory using any method.

5. Max. length of collected string should be 256 symbols. If the user will try to input string longer, than, CLI should stop at the 256th symbol and produce any single sound (BEEP) if the user will press any key from the keyboard, except ENTER or BACKSPACE keys.

6. CLI should be able to detect if the user will press BACKSPACE key and treat it accordingly. Deleted symbol (from the left side of the cursor) should disappear from the screen and it also should be deleted from the collected string. Cursor position should be updated. BACKSPACE treatment should also work in situations when it will be needed to move cursor to the previous line, in case of long strings.

7. If the user will press ENTER like a first key, than CLI should NOT print one empty line and just go to the next line with the prompter.

8. If the user will press BACKSPACE like a first key, than CLI should produce BEEP sound and no other action should be performed.

9. If the user will reach the last line of the screen CLI must continue to work and in case of moving to the next line by typing, or in case of pressing ENTER key, all the upper text from the screen should move one line up.

10. Implement the commands : help, about, ascii, reboot, clear screen and 2 customs.

Implementation of tasks:

Before starting performing this laboratory work I have studied about macro , procedures in assembly for helping me to write a good code and arrange them more clearly. Also, I have used the labels because I found them very useful in my case. Additional I have found examples given by assembly documentation useful for performing the OS, slightly difference was about interrupts.

So, according to the tasks I started to implement them in assembly. I have several labels which perform the main functionalities according to the tasks.

They are:

- **Printing** – this label print the character on the screen and jump to the other labels when it is found. So , how it prints? I used the int 16, 00 , which catch the keyboard control. Also, cmp the accessed key if it should do something or not. Moreover, cmp the length of string which could be less or equal with 256 characters. So description written above we can see in the Fig.2.

- **Backspace –** from name we can understand what it should do. How works it in assembly? I checked the di register which is an counter in my case , that counts the characters. After checking I delete if exist character if not it will provide an beep. Also, I have treated the case when it will be on edge on the line and it should continue from the ending of above line. I performed this with moving the cursor to the right edge of above line through testing the dl, if dl is 0.

- **Enter -** the functionality of enter key is to storing in the declared buffer the collected string and display it one line below. So, for more clear understanding I have used to different color for typing and displaying the string , which after displaying the string I have printed one more new line. Another, functionality of this key is the case when it is pressed first without some characters typed and it should go from new line without any newlines.

Besides labels I have created 2 macros and 2 procedures. The macros I have used for printing the character using the teletype mode , the second macro I have used for setting the position using the x,y coordinates(dl,dh).

The procedures I have used for performing the BEEP sound and for get the cursor position.

Besides basic functionalities I have predefined commands implemented in my OS. There mandatory commands and customs:

Mandatory :

- help
- about
- asci
- reboot
- clean

Custom:

- beep
- echo
- uppercase

For implementing the given commands I started the OS by creating the macro which will extract the characters from cmd_buff and put them into the argv_buff and also will check if there are spaces or tabs and if there are it put 0 instead of space and tab.For example: we type the **echo OS SYSTEM**, the cmd will show **OS SYSTEM.** This macro is called ***extract_words***. So, the input command or string , whatever which was typed by user is compared with predefined commands in the assembly code. If OS has the string typed it will do the respectively function which was given by developer. The user must type exact the command the same how in code is implemented for receiving the actions.Next, I will explain each command what performs.

help - will print a message about the available commands in our OS system.

about – will the message about the developer of OS system.

ascii – will the characters from asci table.

reboot – will restart the system.

clean – will clear the screen.

beep – will produce a beep.

echo – will show the arguments from echo command below.

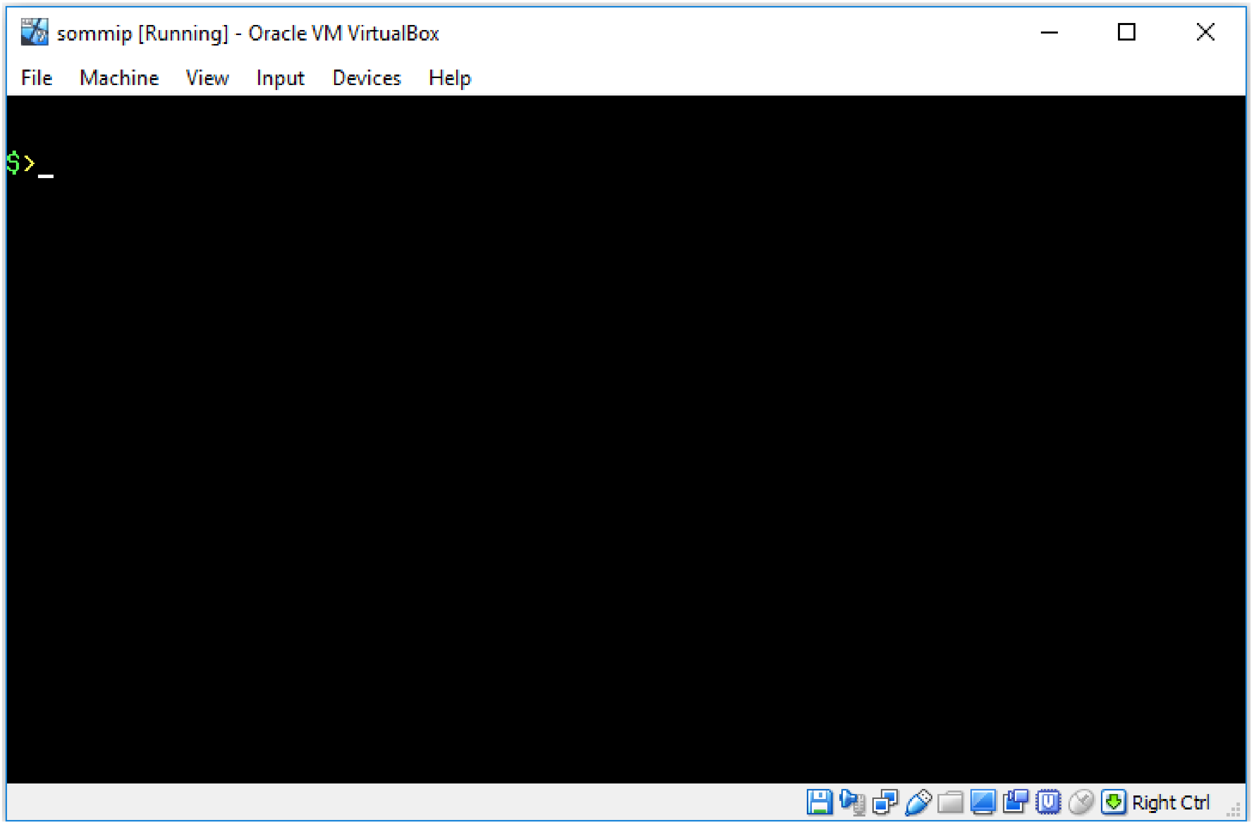
uppercase – will convert the next letters in uppercase.

Observations:

- If the macros and procedures are places in the main work space the assembly will read their first and the program will stop.
- Using jumps we can interpret the If-Else statement like in C language.
- Writing just one line of code we can change the whole logic of the program.
- Is requiring the few instructions for getting the result.
- BEEP sound doesn't work in the Virtual Box.
- The NOP instruction which "**Do nothing**" is very useful in cases with loader.

Screens

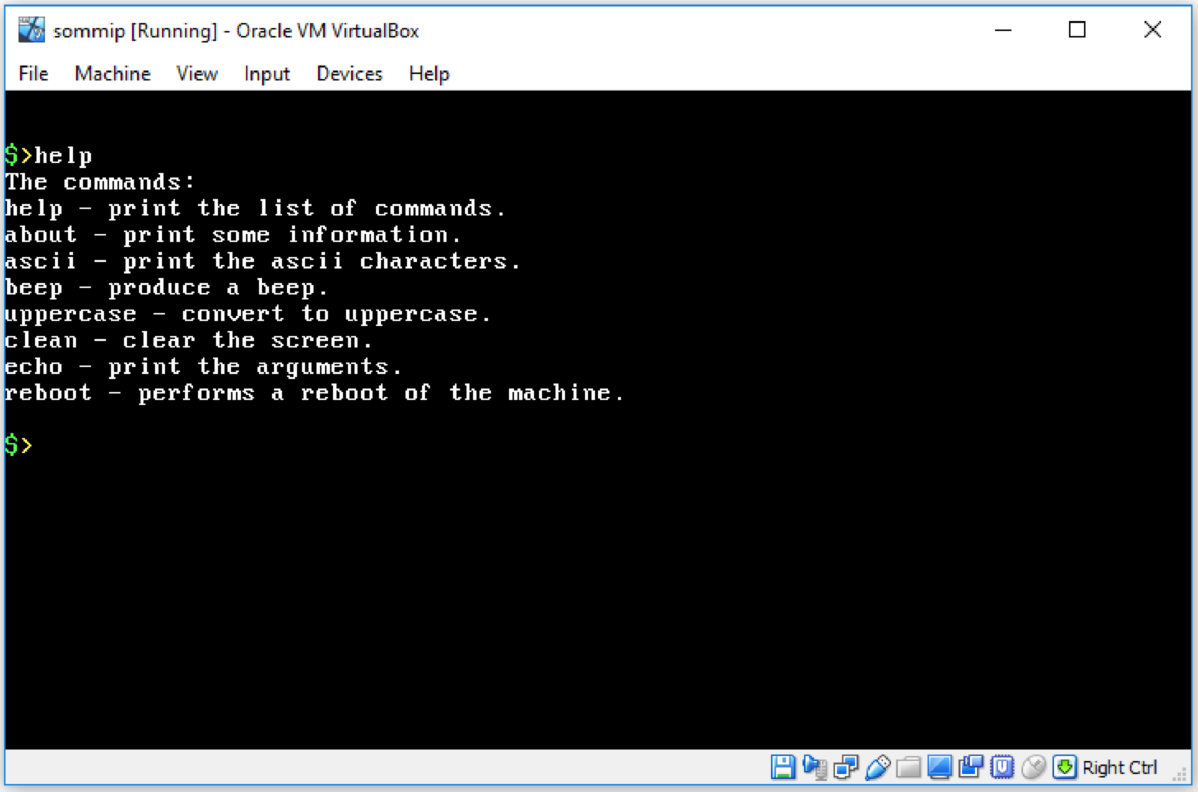
Starting of CLI



```
$>_
```

Fig.1 : First appearance of CLI

Available commands

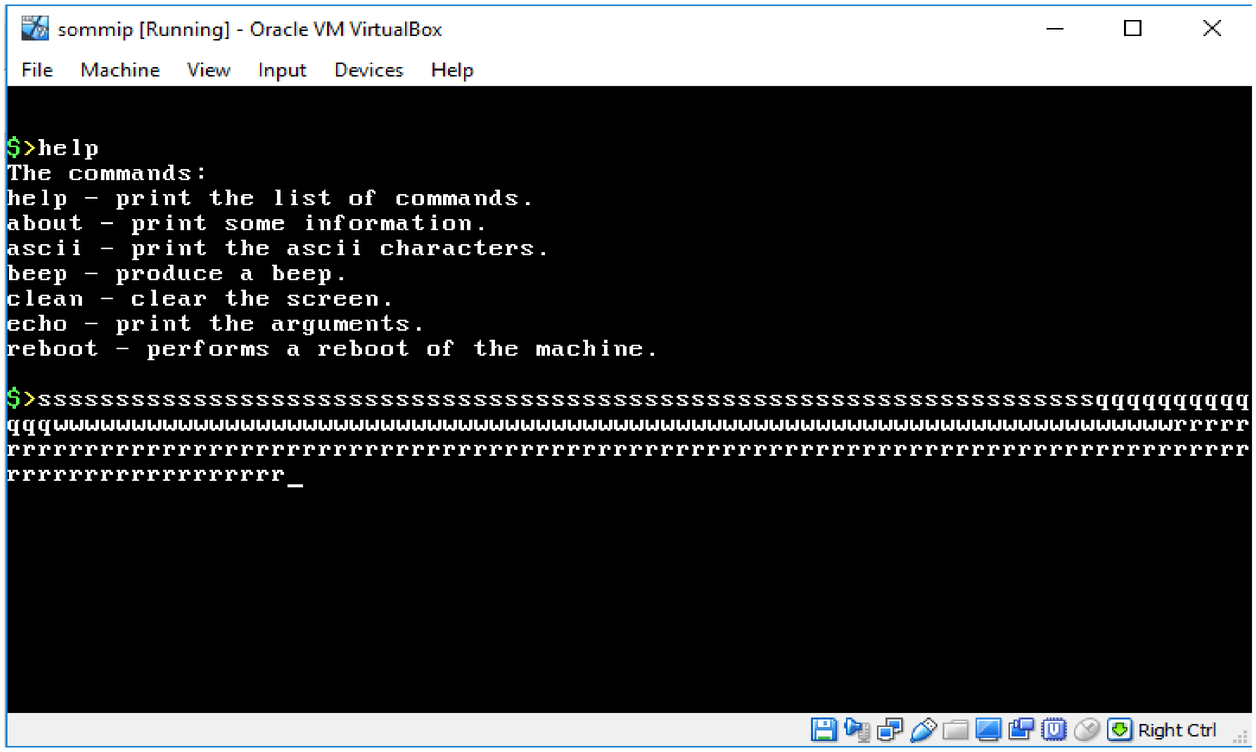

sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
$>help
The commands:
help - print the list of commands.
about - print some information.
ascii - print the ascii characters.
beep - produce a beep.
uppercase - convert to uppercase.
clean - clear the screen.
echo - print the arguments.
reboot - performs a reboot of the machine.

$>
```

Fig.2:Help command

Checking the limit of the 256 characters.

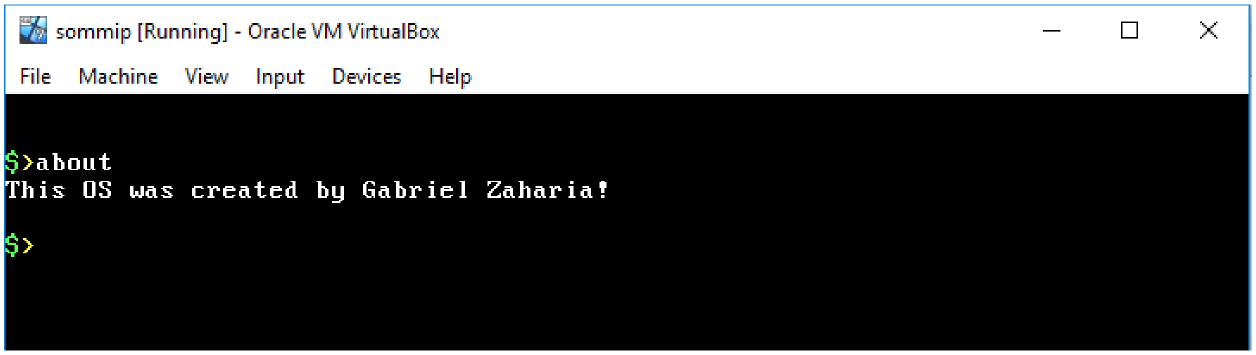

sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
$>help
The commands:
help - print the list of commands.
about - print some information.
ascii - print the ascii characters.
beep - produce a beep.
clean - clear the screen.
echo - print the arguments.
reboot - performs a reboot of the machine.

$>ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssqqqqqqqqqq
qqqwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwrrrrr
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrr_
```

Fig.3: Check the limit and the BEEP sound on trying typing more.

The function of about command

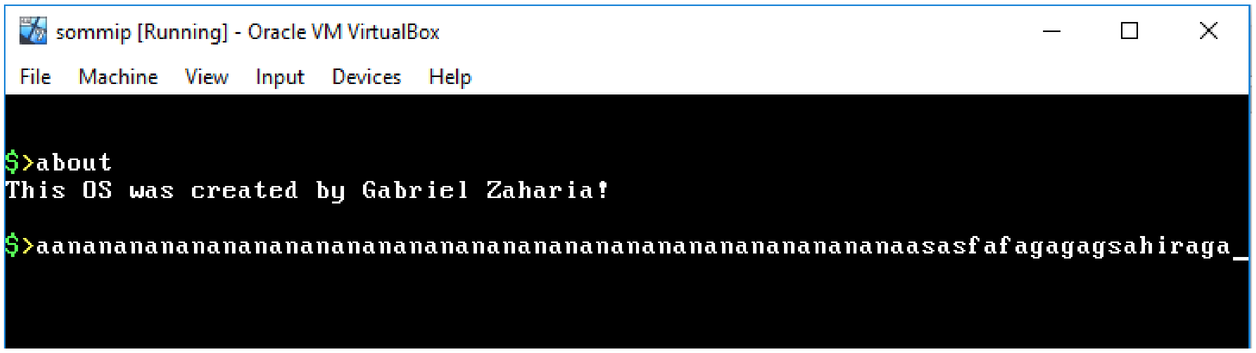

sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

```
$>about
This OS was created by Gabriel Zaharia!

$>
```

Fig.4:about command

Deleting the characters from the right edge

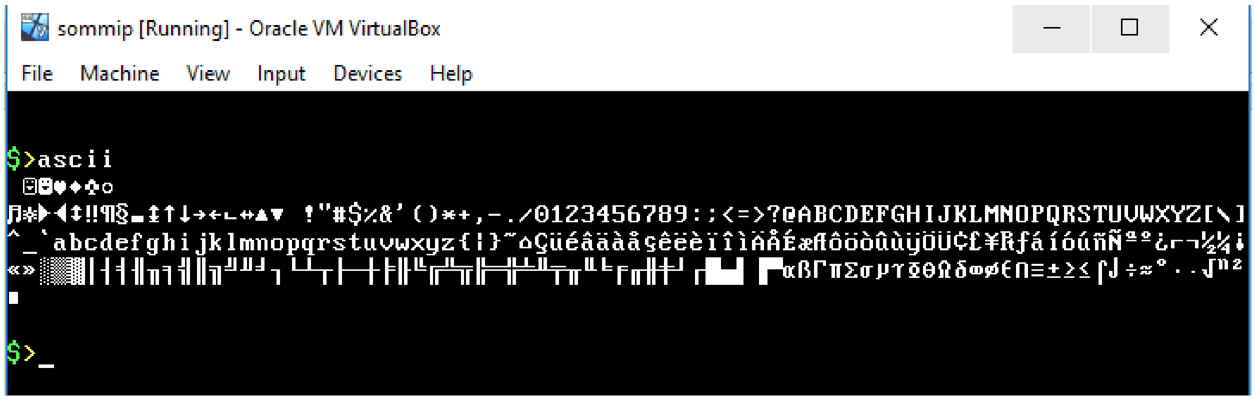


```
$>about
This OS was created by Gabriel Zaharia!

$>aananananananananananananananananananananananananananananaasasfafagagagsahiraga_
```

Fig.5: Treatment of case when is typed more characters than one line.

The function of asci command

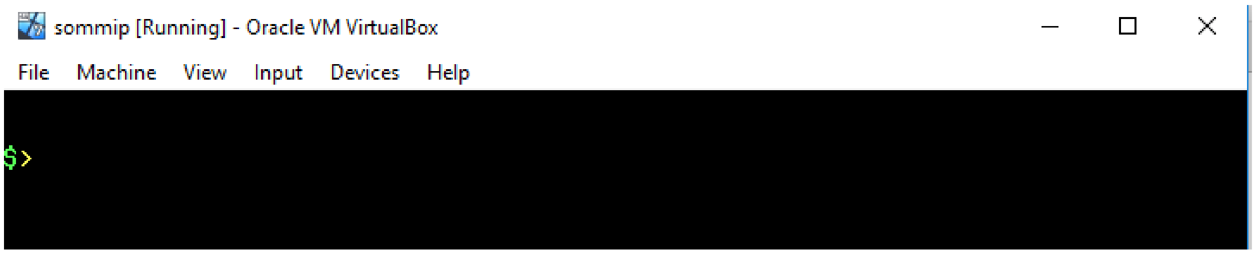


```
$>ascii
 ☺☻♥♦♣○
♫☼►◄↕‼¶§▬↨↑↓→←∟↔▲▼ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{¦}~⌂ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖÜ¢£¥₧ƒáíóúñÑªº¿⌐¬½¼¡
«»░▒▓│┤╡╢╖╕╣║╗╝╜╛┐└┴┬├─┼╞╟╚╔╩╦╠═╬╧╨╤╥╙╘╒╓╫╪┘┌█▄▌▐▀αßΓπΣσµτΦΘΩδ∞φε∩≡±≥≤⌠⌡÷≈°∙·√ⁿ²
■

$>_
```

Fig.6: Asci command

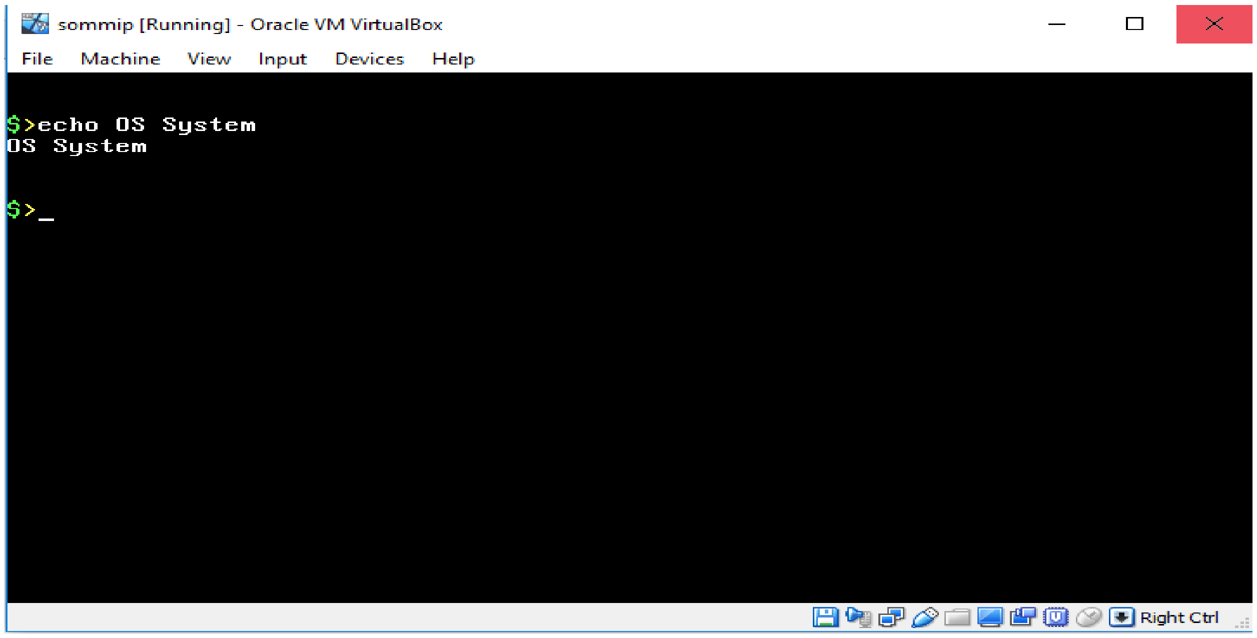
The function of clean command



```
$>
```

Fig.7: Clean command

The function of echo command

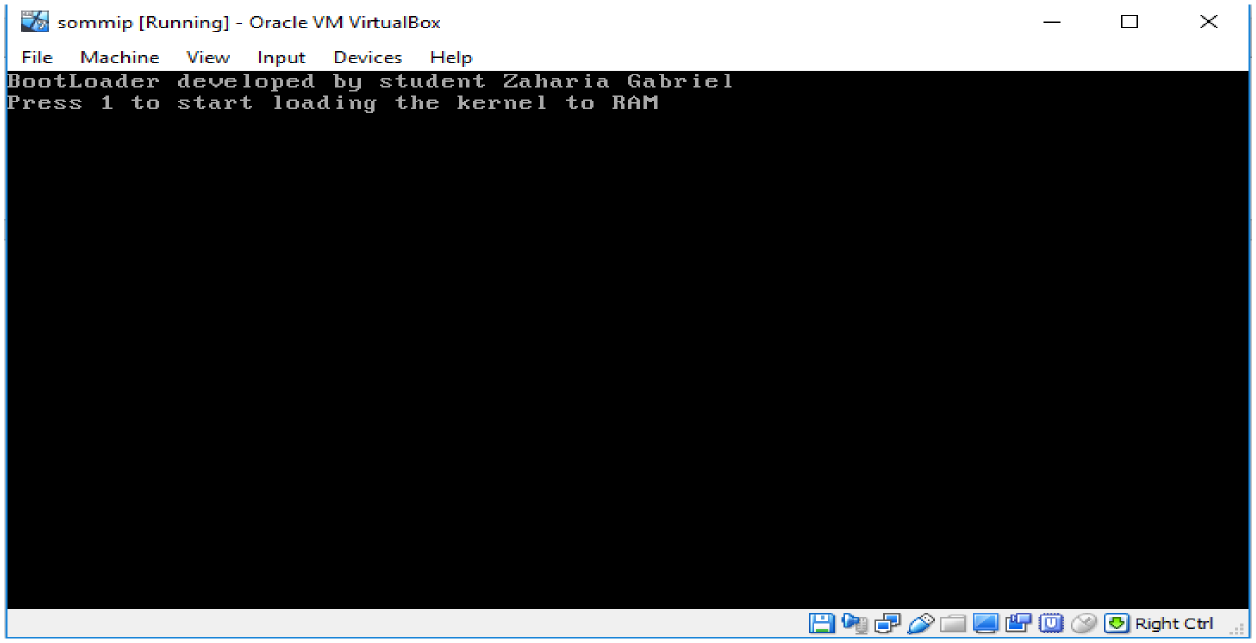

sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
$>echo OS System
OS System


$>_
```

Fig.8: Echo command

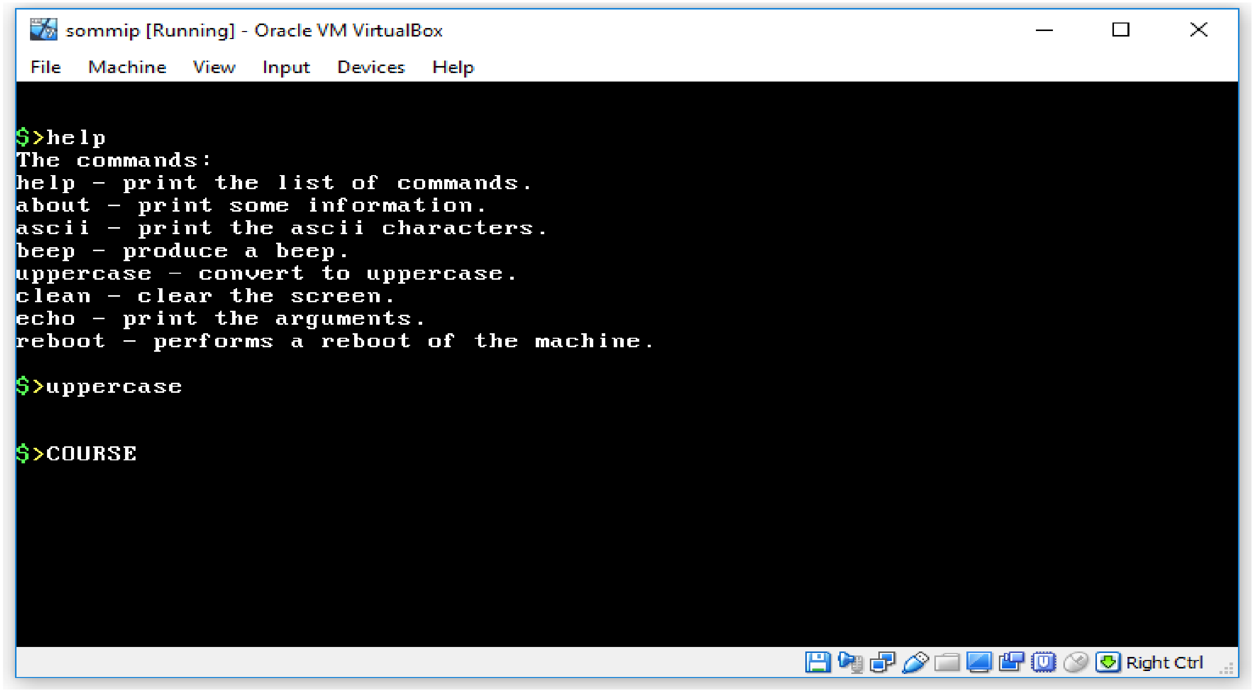
The function of reboot command


sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
BootLoader developed by student Zaharia Gabriel
Press 1 to start loading the kernel to RAM
```

Fig.9:Reboot command

The function of uppercase command


sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

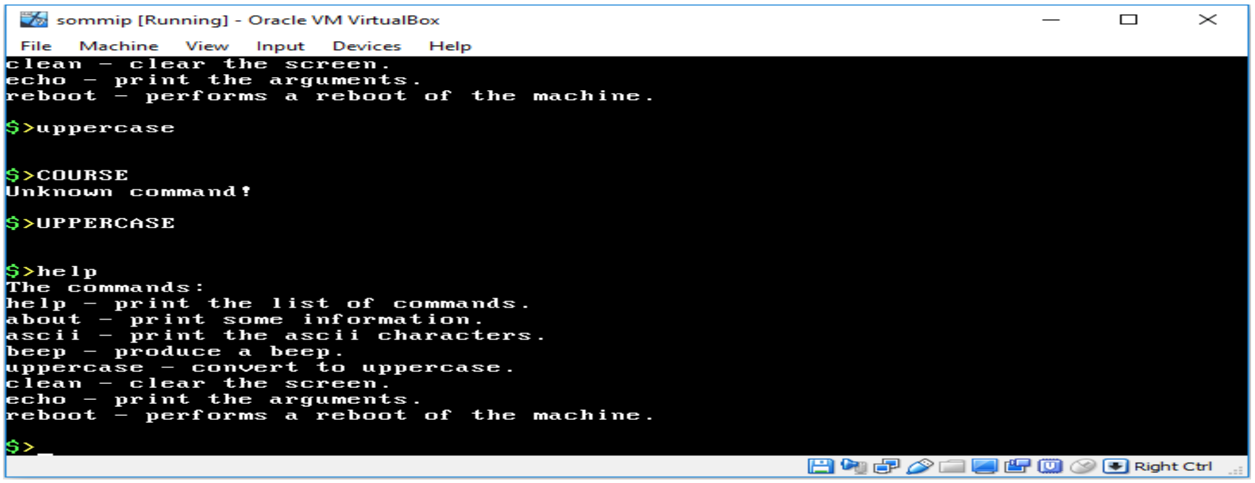
```
$>help
The commands:
help - print the list of commands.
about - print some information.
ascii - print the ascii characters.
beep - produce a beep.
uppercase - convert to uppercase.
clean - clear the screen.
echo - print the arguments.
reboot - performs a reboot of the machine.

$>uppercase


$>COURSE
```

Fig.10: Uppercase command

The function of UPPERCASE command


sommip [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
clean - clear the screen.
echo - print the arguments.
reboot - performs a reboot of the machine.

$>uppercase


$>COURSE
Unknown command!

$>UPPERCASE


$>help
The commands:
help - print the list of commands.
about - print some information.
ascii - print the ascii characters.
beep - produce a beep.
uppercase - convert to uppercase.
clean - clear the screen.
echo - print the arguments.
reboot - performs a reboot of the machine.

$>_
```

Fig.11:UPPERCASE command, disable the uppercase

Conclusion:

During this laboratory work I obtained skills operating with keyboard interrupts, printing the characters , storing the characters from the user’s input and display it on the screen. So, performing the tasks I find out that assembly is very precious language and whole based on hard parts and memory and it use the translation method for processing the code, it starts from assembly -> instructions set architecture -> digital logic.

Also, I gained a lot of skills operating with macros and how calling them into procedures or in the code for reducing the amount of replication the code and reduce the time.

The OS System imitates the CMD from Windows which has the same basic commands like it.

Due to the implementation and the course and practical lessons we can run our OS on different Machines and to be more detailed due to the used interrupts and location of org. We have tested this in Virtual Box which gave us the possibility to run on DOS operating system.