Ministry of Education, Culture and Research of the Republic of Moldova
Technical University of Moldova
Department of Software and Automation Engineering

REPORT

Laboratory work no. 2
Keyboard Input

Elaborated:
st. gr. FAF-213 Botnari Ciprian

Verified:
lect. univ. Rostislav Călin

Chişinău – 2023

Table of Contents

Topic: Keyboard Input

Tasks

1. Create a program in assembler which will "echo" what is typed from the keyboard. Each ASCII character which will be pressed from the keyboard should appear on the screen and the cursor should move to the next position.

2. Special actions need to be implemented only for 2 special keys from the keyboard:
   a. "backspace key" - in this case symbol from the left side of the cursor should disappear and the cursor should be moved one position back (If the cursor already is in the first position, then nothing should happen. Special case is if the cursor is on the next line, then when is pressed Backspace in the first column, then cursor should move to the previous line in last column);
   b. "enter key" - in this case all previously introduced string should be printed to the screen starting with the next line and after one "empty" line (but if "enter key" will be pressed as the first key, in this case NO "empty" line should be added and the action should just go to the next line)

3. (OPTIONAL)
   The maximum length of input string should not exceed 256 characters. If the user will want to input more than 256 characters than the input should be stopped and in this case only "backspace" or "enter" keys should be accepted.

Requirements

- 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 Virtual Box VM and the text which you intended to print should appear on the screen.
- You can use any assembly compiler.
- You should be able to modify the code, to recompile it and to boot the VM with new version of program.
- In order to use documentation from TechHelp / XView DOS application, you can install DosBox.

Code

build.sh

```
#!/bin/bash

if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename.asm>"
  exit 1
fi

filename_with_extension="$1"
filename="${filename_with_extension%.*}"

asm_file="$filename_with_extension"
com_file="$filename.com"
flp_file="$filename.flp"
```

```
# Step 1: Compile the assembly code to a .com file
nasm -f bin -o "$com_file" "$asm_file"
if [ $? -ne 0 ]; then
  echo "Compilation failed. Check your assembly code."
  exit 1
fi
echo "Step 1: Compilation completed."

# Step 2: Copy the .com file to a .flp file
cp "$com_file" "$flp_file"
echo "Step 2: Copied $com_file to $flp_file."

# Step 3: Resize the .flp file to 1.44MB
truncate -s 1474560 "$flp_file"
echo "Step 3: Resized $flp_file to 1.44MB."

# Step 4: Change the storage to $flp_file in VirtualBox
VM_NAME="BestOS"
VBoxManage storageattach "$VM_NAME" --storagectl "Floppy" --port 0 --
device 0 --type fdd --medium "$flp_file"
echo "Step 4: Storage in VirtualBox changed to $flp_file."

echo "All steps completed successfully."
```

keyboard.asm

```
section .data
  row db 0                ; x
  column db 0             ; y
  buffer times 256 db 0   ; 256 bytes buffer

section .text
  global _start

_start:
  ; Set up the stack
  mov ax, 0x0000
  mov ss, ax
  mov sp, 0x7C00

  ; Call BIOS to clear the screen
  call clear_screen

  ; Initialize the cursor position (row and column)
  mov si, buffer
  mov byte [row], 0
  mov byte [column], 0
```

```
main:
  call read_char

  ; Backspace
  cmp al, 0x08
  je backspace
  ; Enter
  cmp al, 0x0D
  je enter

  call print_char
  jmp main

backspace:
  ; Check if we are at the beginning of the current line
  cmp byte [column], 0
  je prev_line

  ; Backspace, Print a space, Backspace again
  mov ah, 0x0E
  mov al, 0x08
  int 0x10
  mov al, 0x20
  int 0x10
  mov al, 0x08
  int 0x10

  ; Update the current column
  dec byte [column]

  ; Buffer is empty
  cmp si, buffer
  je main

  ; Remove the last character from the buffer
  dec si
  mov byte [si], 0

  jmp main

prev_line:
  ; Check if we are at the beginning of the screen
  cmp byte [row], 0
  je main

  ; Move the cursor to the beginning of the previous line
  dec byte [row]
  mov byte [column], 79
  mov ah, 0x02
  mov dh, byte [row]
  mov dl, byte [column]
  int 0x10

  jmp main
```

```
; Copy a string from SI to the screen
copy_string_loop:
  lodsb
  cmp al, 0
  je copy_string_exit
  int 0x10
  jmp copy_string_loop

copy_string_exit:
  ret ; return to the caller

enter:
  ; Check if we are at the end of the screen
  cmp byte [row], 24
  je main

  call new_line

  ; check if the buffer is empty
  mov si, buffer
  cmp byte [si], 0
  je main

  ; Print the buffer
  mov ah, 0x0E
  mov si, buffer
  call copy_string_loop

  ; Clear the buffer, reset the buffer pointer
  mov si, buffer
  xor cx, cx
  fill_buffer_loop:
    mov byte [si], 0
    inc si
    inc cx
    cmp cx, 256
    jl fill_buffer_loop

  ; Move the cursor to the beginning of the next line
  call new_line
  call new_line

  ; Reset the buffer pointer, reset the current column
  mov si, buffer
  mov byte [column], 0

  jmp main
```

```
new_line:
  inc byte [row]
  mov byte [column], 0
  mov ah, 02h           ; update cursor position
  mov bh, 0             ; page number
  mov dh, byte[row]     ; row
  mov dl, byte[column]  ; column
  int 0x10              ; call BIOS
  ret ; return to the caller

read_char:
  mov ah, 0x00 ; read char from keyboard
  int 0x16     ; call BIOS
  ret ; return to the caller

print_char:
  inc byte [column]     ; Update the current column

  cmp byte [column], 80 ; Check if we are at the end of the screen
  je enter

  mov [si], al    ; Store the character in the buffer
  inc si          ; Update the buffer pointer

  mov ah, 0x0E  ; print char
  int 0x10      ; call BIOS
  ret ; return to the caller

clear_screen:
  mov ah, 06h   ; scroll window up
  mov al, 0     ; clear entire screen
  mov cx, 0     ; upper left corner (0, 0)
  mov dx, 184fh ; lower right corner (24, 79)
  mov bh, 1Eh   ; yellow on blue
  int 0x10
  ret ; return to the caller
```

Results

Below is an example of the running program for different inputs. When Enter is pressed, the line is duplicated and a new line is added.

Figure 1. *Example of different inputs*

Conclusion

In a nutshell, this project was quite the journey into the world of assembly programming for me. I had to create a program that echoes whatever I type on the screen, with some special tricks for the "backspace" and "enter" keys. The handy build script made testing and tweaking my code a breeze. Knowing I can always go back and make improvements is reassuring. Overall, this laboratory work taught me a lot about low-level programming and its role in building an operating system.

GitHub

Sufferal/os (github.com)