

Lab Report #10

Saad Ahmad

(20P-0051)

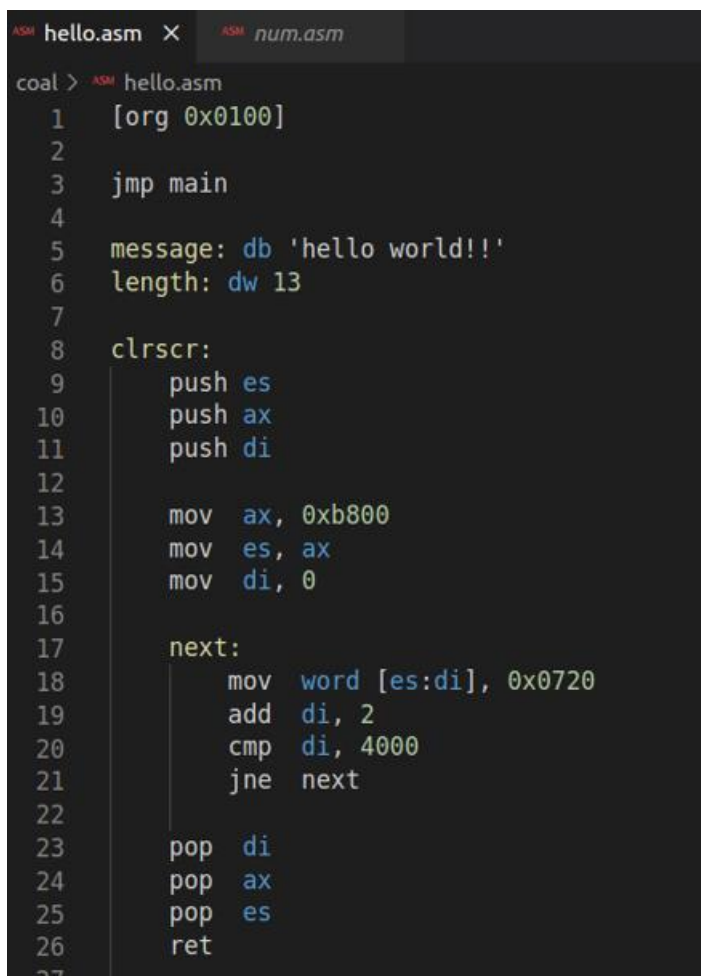
ASCII Codes:

The computer listens, sees, and speaks in numbers. Even a character is a number inside the computer. For example, the keyboard is labelled with characters however when we press 'A', a specific number is transferred from the keyboard to the computer.

An 'A' on any computer and any operating system is an 'A' on every other computer and operating system. This is because a standard numeric representation of all commonly used characters has been developed. This is called the ASCII code, where ASCII stands for American Standard Code for Information Interchange. The name depicts that this is a code that allows the interchange of information; 'A' written on one computer will remain an 'A' on another. All ASCII based computers use the same code.

“Hello World” in Assembly Language:

Code:



```
ASM hello.asm X ASM num.asm
coal > ASM hello.asm
1  [org 0x0100]
2
3  jmp main
4
5  message: db 'hello world!!'
6  length: dw 13
7
8  clrscr:
9      push es
10     push ax
11     push di
12
13     mov ax, 0xb800
14     mov es, ax
15     mov di, 0
16
17     next:
18         mov word [es:di], 0x0720
19         add di, 2
20         cmp di, 4000
21         jne next
22
23     pop di
24     pop ax
25     pop es
26     ret
27
```

```

print:
    push bp
    mov bp, sp
    push es
    push ax
    push cx
    push si
    push di

    mov ax, 0xb800
    mov es, ax
    mov di, 0

    mov si, [bp + 6]
    mov cx, [bp + 4]
    mov ah, 0x03

nextch:
    mov al, [si] ; moving the alphabets in al
    mov [es:di], ax
    add di, 2
    add si, 1
    loop nextch

    pop di
    pop si
    pop cx
    pop ax
    pop es
    pop bp
    ret 4

```

```

main:
    call clrscr ; calling clear ftn to clear the screen

    mov ax, message

    push ax
    push word [length]

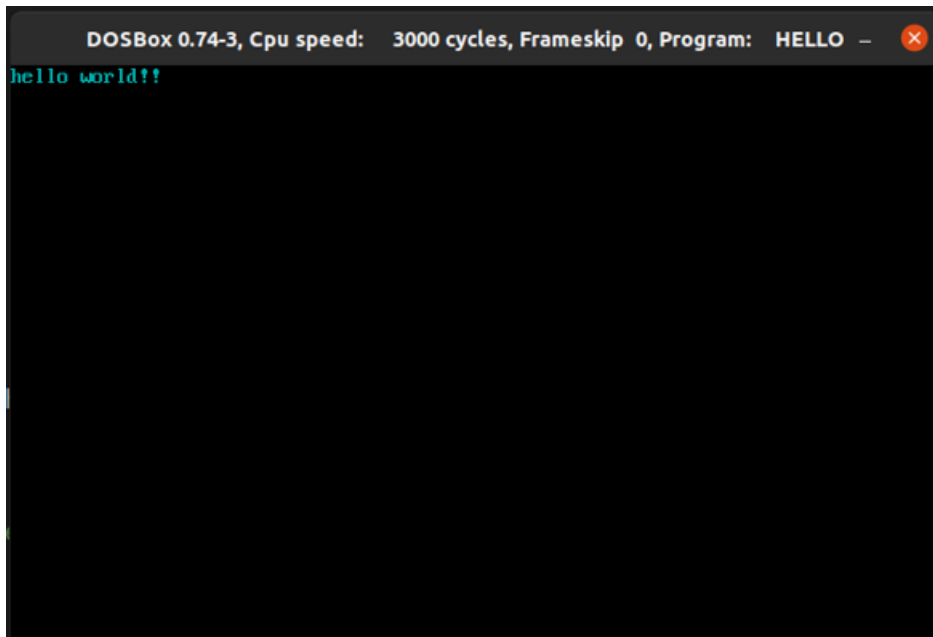
    call print

    ; wait for keypress
    mov ah, 0x1 ; input char is 0x1 in ah
    int 0x21

    mov ax, 0x4c00
    int 0x21

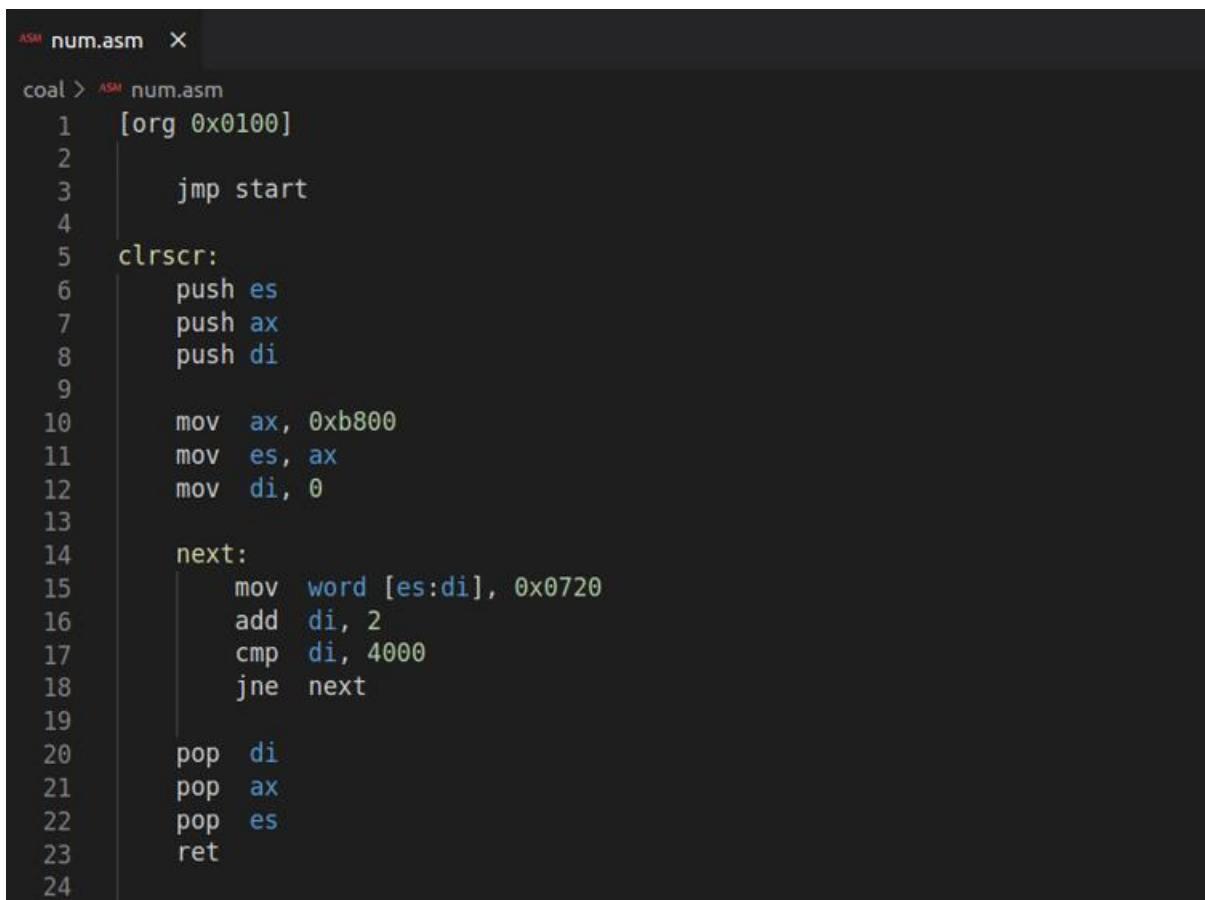
```

Output:

A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: HELLO". The main window area is black with the text "hello world!!" printed in a light blue/cyan color at the top left.

Printing **Numbers** in Assembly:

Code:

A screenshot of an assembly editor window titled "num.asm". The code is as follows:

```
coal > ASM num.asm
1  [org 0x0100]
2
3  jmp start
4
5  clrscr:
6  push es
7  push ax
8  push di
9
10 mov ax, 0xb800
11 mov es, ax
12 mov di, 0
13
14 next:
15     mov word [es:di], 0x0720
16     add di, 2
17     cmp di, 4000
18     jne next
19
20 pop di
21 pop ax
22 pop es
23 ret
24
```

```

25 printnum:
26     push bp
27     mov bp, sp
28     push es
29     push ax
30     push bx
31     push cx
32     push dx
33     push di
34
35     ; first, let's split digits and push them onto the stack
36
37     mov ax, [bp+4] ; number to print
38     mov bx, 10     ; division base 10
39     mov cx, 0      ; total digit counter
40
41     nextdigit:
42         mov dx, 0 ; zero out
43         div bx    ; divides ax/bx .. quotient in ax, remainder in dl
44         add dl, 0x30 ; convert to ASCII
45         push dx    ; push to stack for later printing
46         inc cx     ; have another digit
47         cmp ax, 0  ; is there something in quotient?
48         jnz nextdigit
49

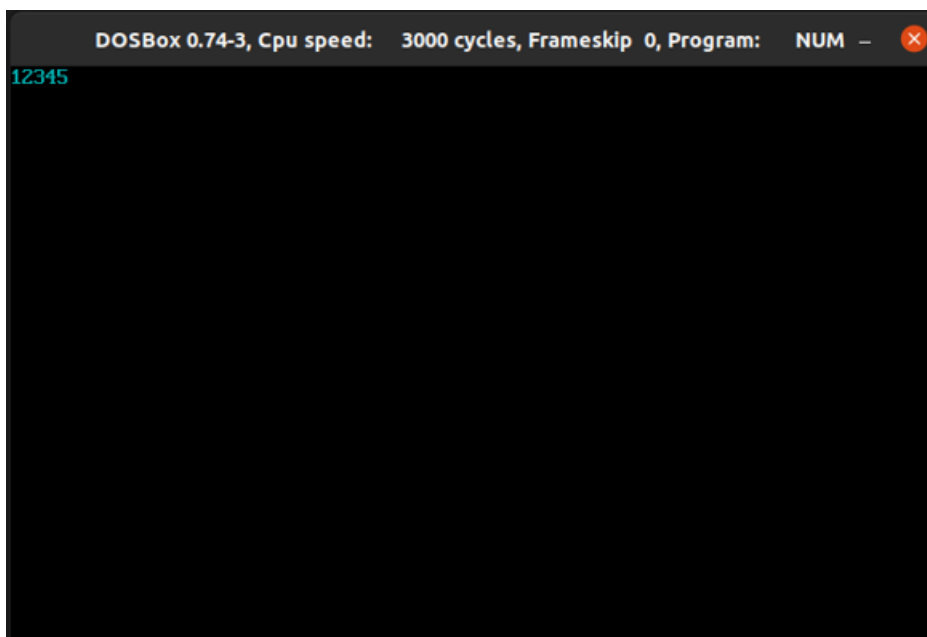
```

```

50      ; now let's do the printing
51
52      mov ax, 0xb800
53      mov es, ax
54
55      mov di, 0
56      nextpos:
57          pop dx          ; digit to output. Already in ASCII
58          mov dh, 0x03    ; changing the color
59          mov [es:di], dx
60          add di, 2
61          loop nextpos
62
63      pop di
64      pop dx
65      pop cx
66      pop bx
67      pop ax
68      pop es
69      pop bp
70      ret 2
71
72      start:
73          call clrscr
74
75          mov ax, 12345
76          push ax
77          call printnum
78
79          mov ah, 0x1
80          int 0x21
81
82          mov ax, 0x4c00
83          int 0x21
84

```

Output:



Screen Location Calculation:

As we used a fixed attribute and displayed at a fixed screen location. We will change that to use any position on the screen and any attribute. For mapping from the two-dimensional coordinate system of the screen to the one-dimensional memory, we need to multiply the row number by 80 since there are 80 columns per row and add the column number to it and again multiply by two since there are 2 bytes for each character.

Formula:

$$\text{location} = (\text{hypos} * 80 + \text{epos}) * 2$$

Code:

```
ASM loc.asm X
coal > ASM loc.asm
1  [org 0x0100]
2
3  jmp main
4
5  message_1: db 'Saad Ahmad (20P-0015)'
6  lenght_1: dw 21
7
8  clrscr:
9      push es
10     push ax
11     push di
12
13     mov ax , 0xb800
14     mov es , ax
15     mov di , 0
16
17     nextloc:
18         mov word[es:di] , 0x0720
19         add di , 2
20         cmp di , 4000
21         jne nextloc
22
23         pop di
24         pop ax
25         pop es
26     ret
27
```

```

30 printstr:
31     push bp
32     mov bp , sp
33
34     push es
35     push ax
36     push cx
37     push si
38     push di
39
40     mov ax , 0xb800
41     mov es , ax
42     mov al , 80
43     mul byte [bp+10]
44     add ax , [bp + 12]
45     shl ax , 1
46     mov di , ax
47
48     mov si , [bp + 6]
49     mov cx , [bp + 4]
50     mov ah , [bp + 8]
51
52     next:
53         mov al , [si]
54         mov [es:di] , ax
55         add di , 2
56         add si , 1
57         loop next
58
59     pop di
60     pop si
61     pop cx
62     pop ax
63     pop es
64     pop bp
65
66     ret 10
67

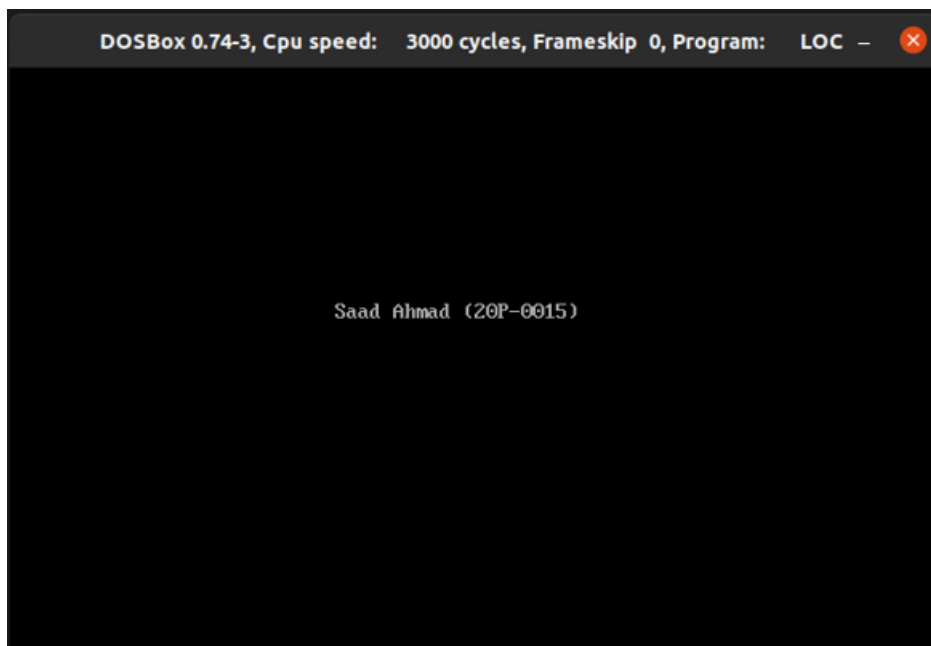
```

```

69  main:
70
71      call clrscr
72
73      mov ax , 28          ; x = 31
74      push ax
75      mov ax , 10          ; y = 10
76      push ax
77      mov ax , 7           ; color of the text and intensity
78      push ax
79      mov ax , message_1
80      push ax
81      push word[lenght_1]
82      call printstr
83
84      mov ah, 0x1          ; input char is 0x1 in ah
85      int 0x21
86
87      mov ax , 0x4c00
88      int 0x21
89

```

Output:



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: LOC

Saad Ahmad (20P-0015)