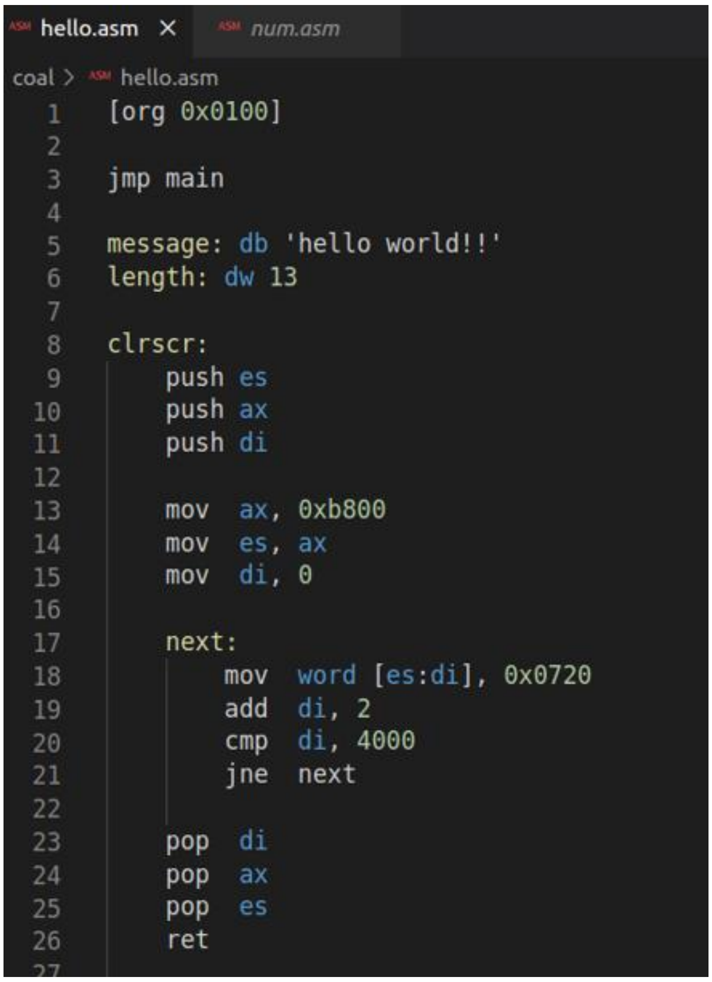
Lab Report #10
Saad Ahmad
(20P-0051)

ASCII Codes:

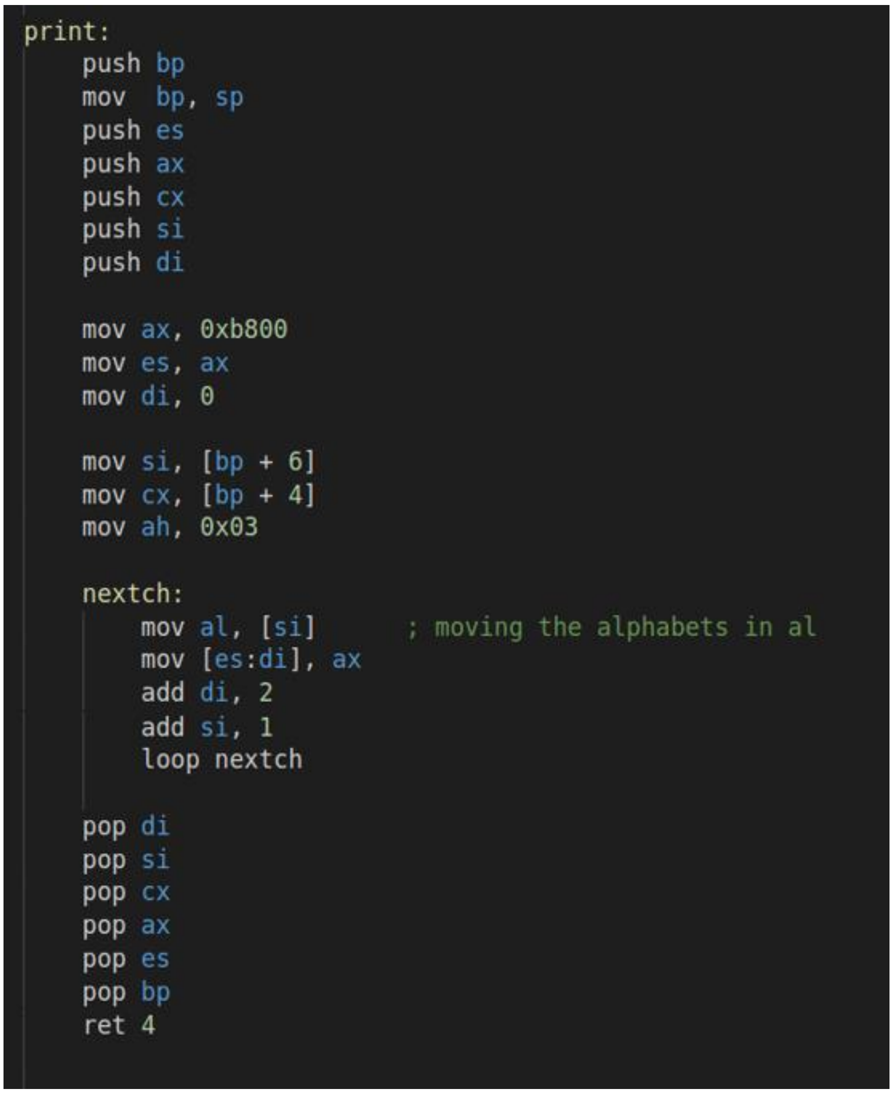
The computer listens, sees, and speaks in numbers. Even a character is a number inside the computer. For example, the keyboard is labelled with characters however when we press ‘A’, a specific number is transferred from the keyboard to the computer.
An ‘A’ on any computer and any operating system is 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:

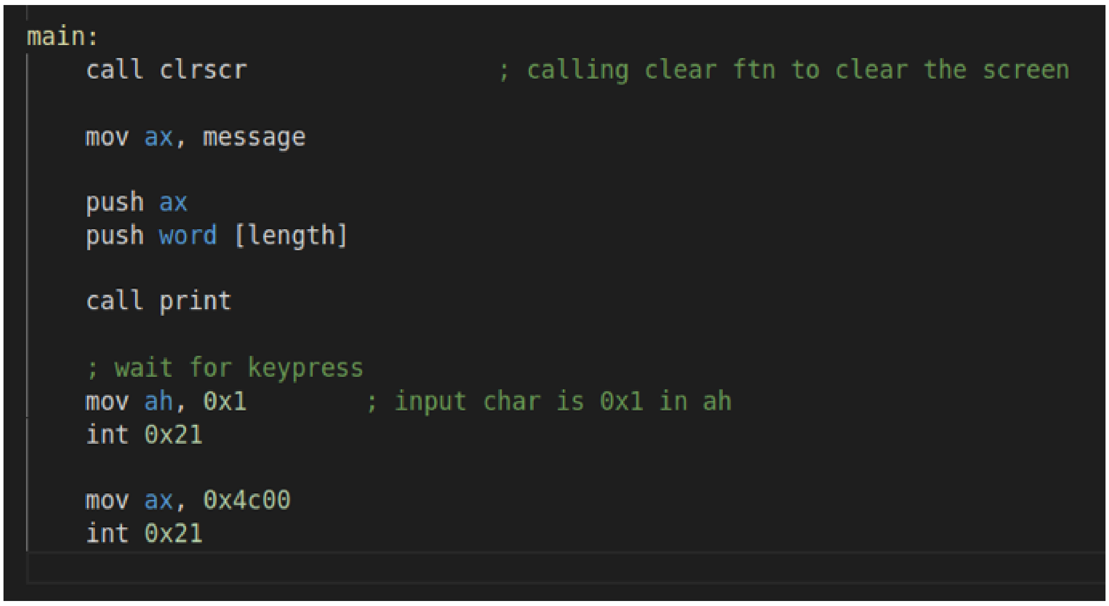
Code:



```
[org 0x0100]

jmp main

message: db 'hello world!!'
length: dw 13

clrscr:
    push es
    push ax
    push di

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

    next:
        mov  word [es:di], 0x0720
        add  di, 2
        cmp  di, 4000
        jne  next

    pop  di
    pop  ax
    pop  es
    ret
```

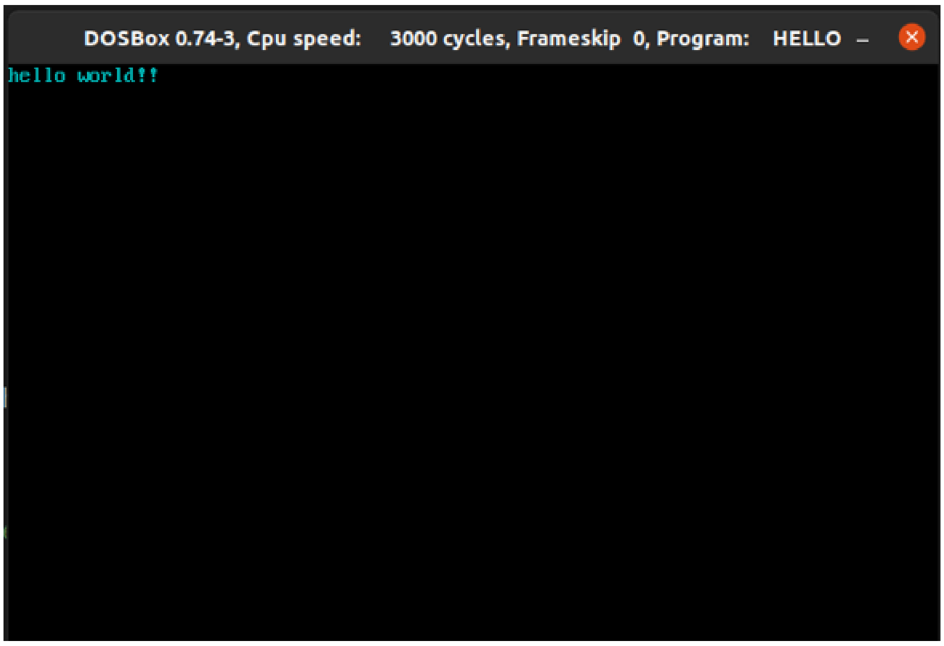


```
print:
    push bp
    mov  bp, sp
    push es
    push ax
    push cx
    push si
    push di

    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
```

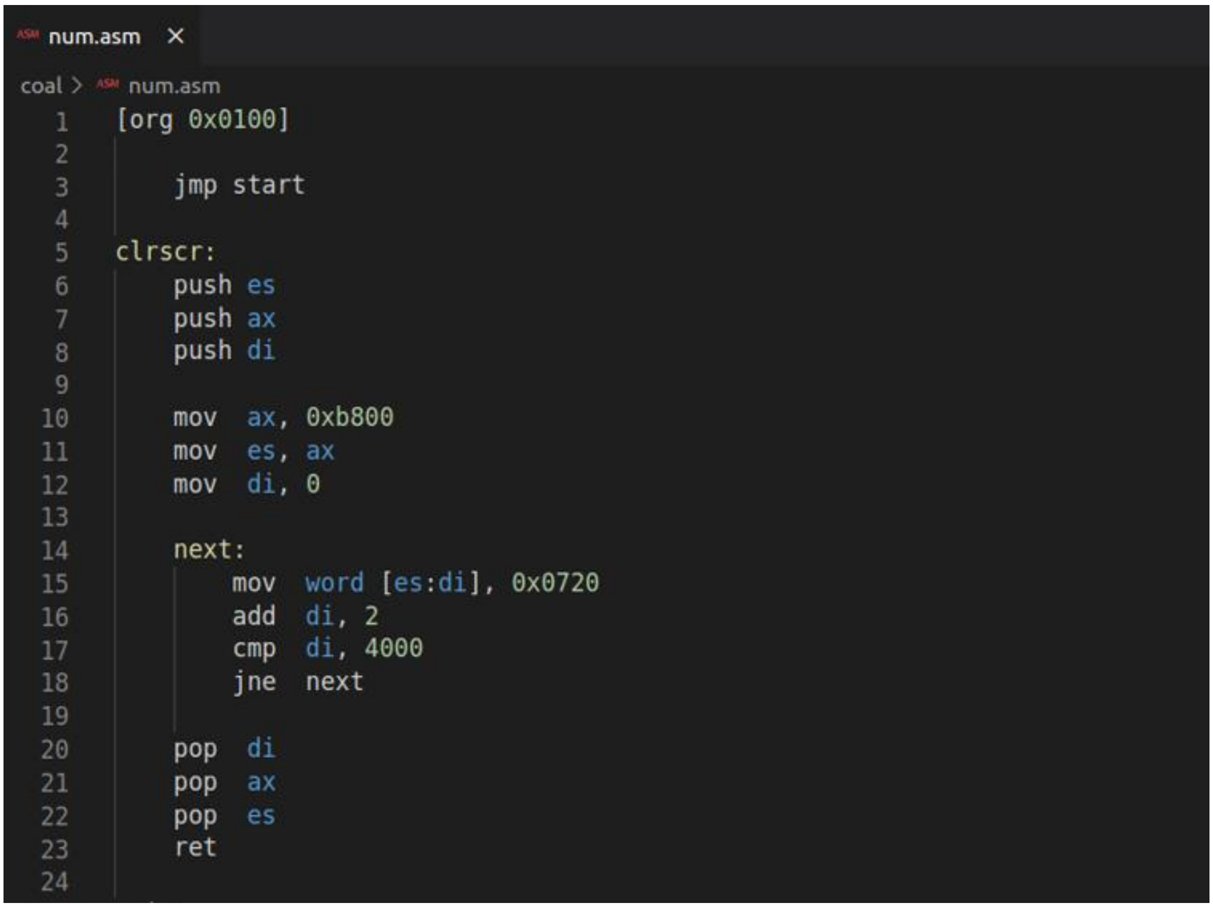
Output:


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

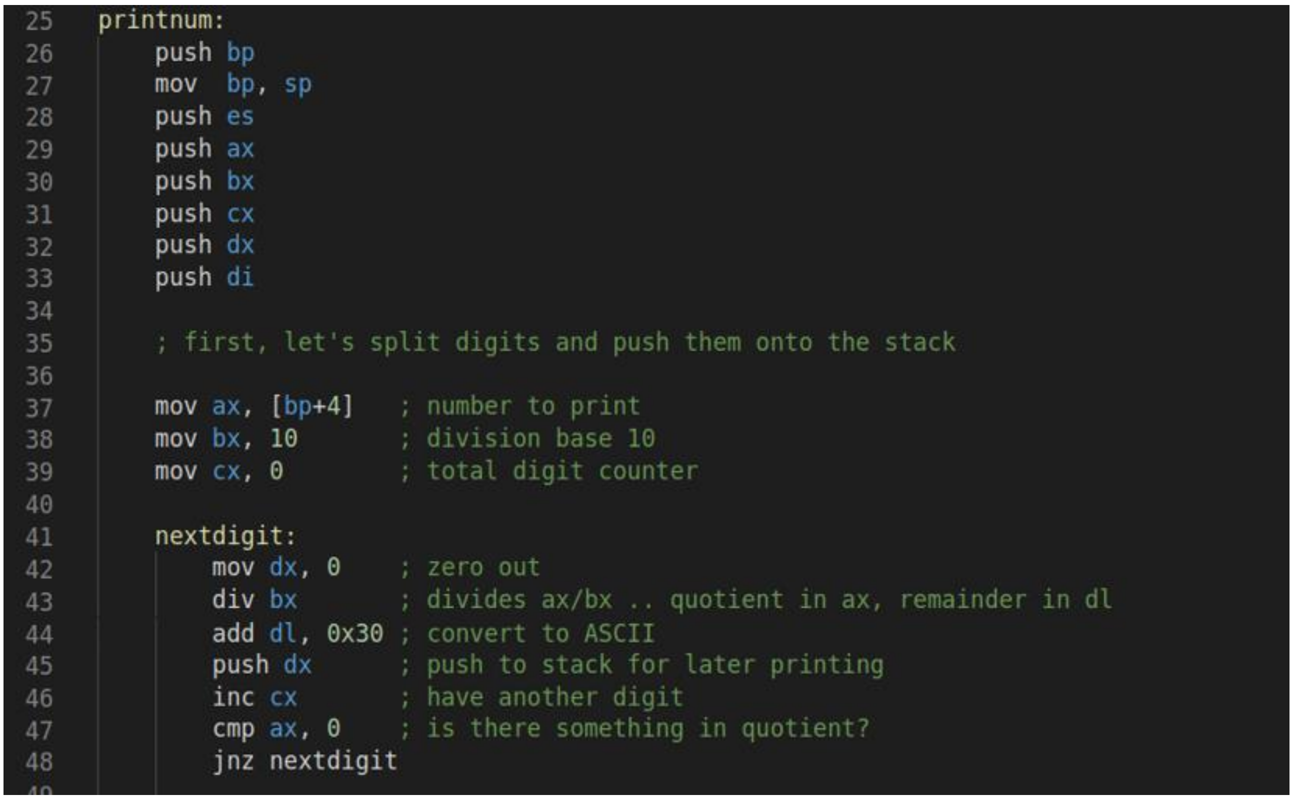
```
hello world!!
```

Printing **Numbers** in Assembly:

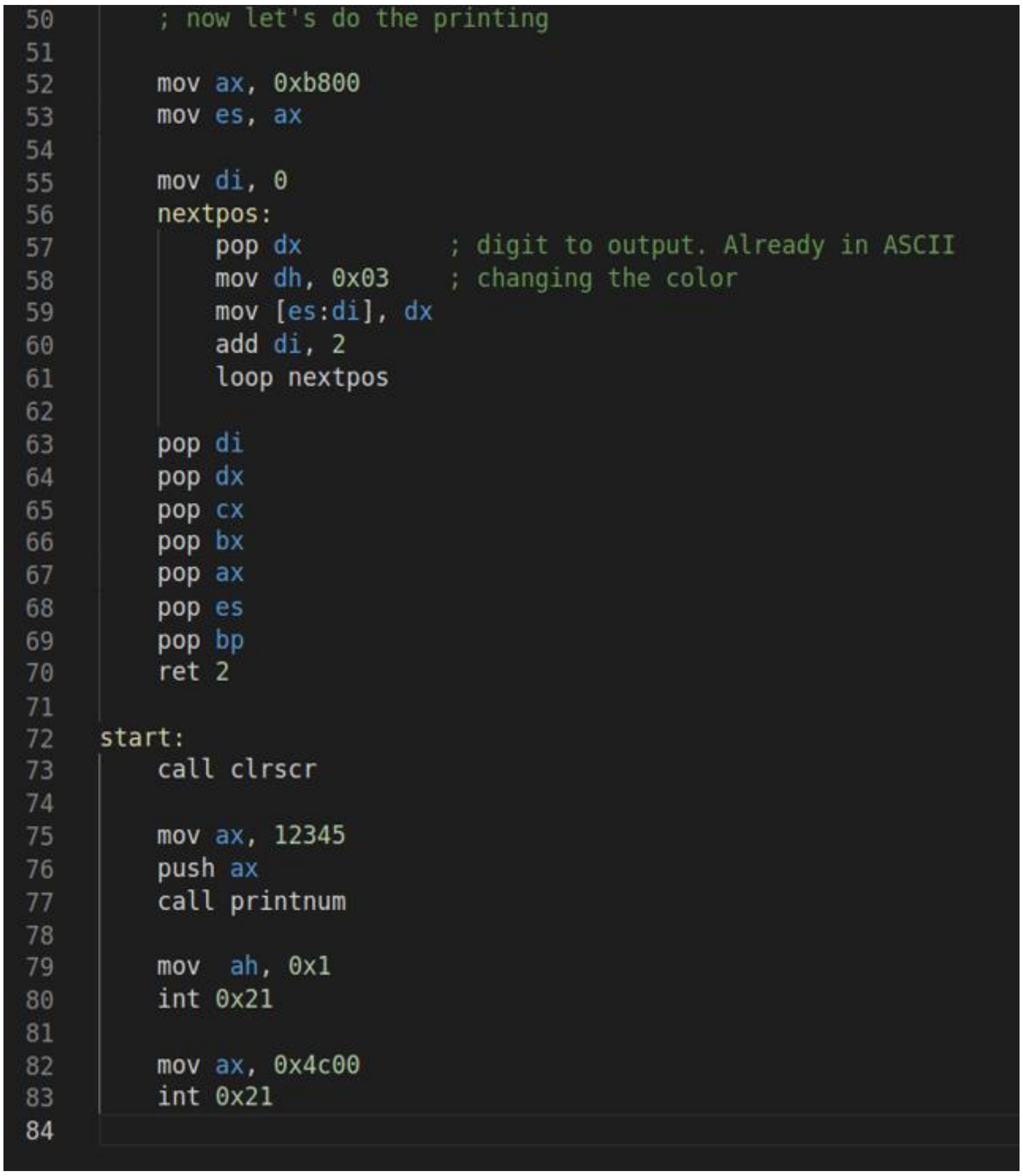
Code:



```
[org 0x0100]

    jmp start

clrscr:
    push es
    push ax
    push di

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

    next:
        mov  word [es:di], 0x0720
        add  di, 2
        cmp  di, 4000
        jne  next

    pop  di
    pop  ax
    pop  es
    ret

```

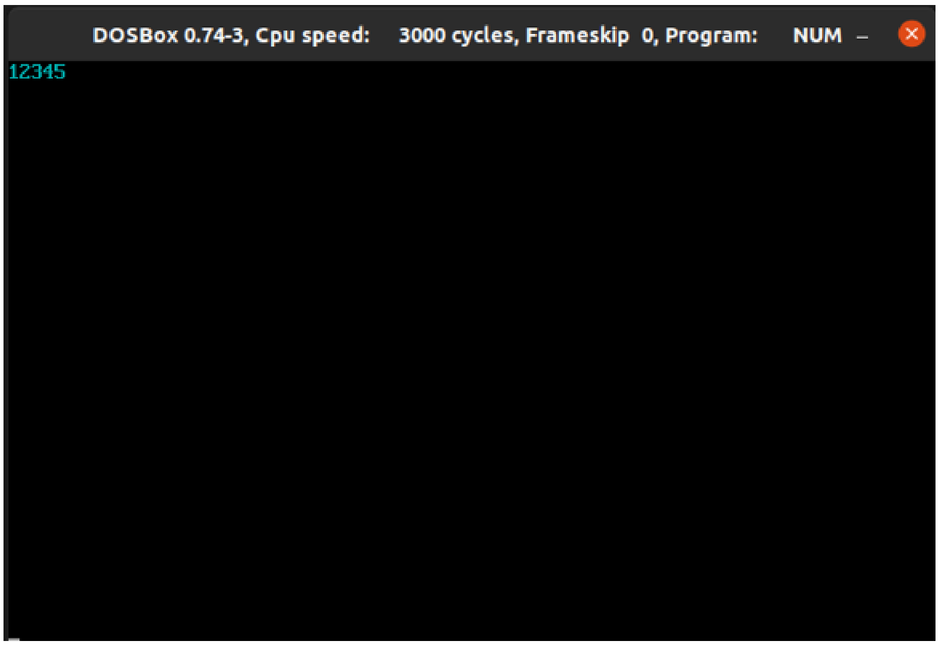


```
printnum:
    push bp
    mov  bp, sp
    push es
    push ax
    push bx
    push cx
    push dx
    push di

    ; first, let's split digits and push them onto the stack

    mov ax, [bp+4]    ; number to print
    mov bx, 10        ; division base 10
    mov cx, 0         ; total digit counter

    nextdigit:
        mov dx, 0     ; zero out
        div bx        ; divides ax/bx .. quotient in ax, remainder in dl
        add dl, 0x30 ; convert to ASCII
        push dx       ; push to stack for later printing
        inc cx        ; have another digit
        cmp ax, 0     ; is there something in quotient?
        jnz nextdigit
```



```
    ; now let's do the printing

    mov ax, 0xb800
    mov es, ax

    mov di, 0
    nextpos:
        pop dx          ; digit to output. Already in ASCII
        mov dh, 0x03    ; changing the color
        mov [es:di], dx
        add di, 2
        loop nextpos

    pop di
    pop dx
    pop cx
    pop bx
    pop ax
    pop es
    pop bp
    ret 2

start:
    call clrscr

    mov ax, 12345
    push ax
    call printnum

    mov  ah, 0x1
    int 0x21

    mov ax, 0x4c00
    int 0x21
```

Output:


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

```
12345
```

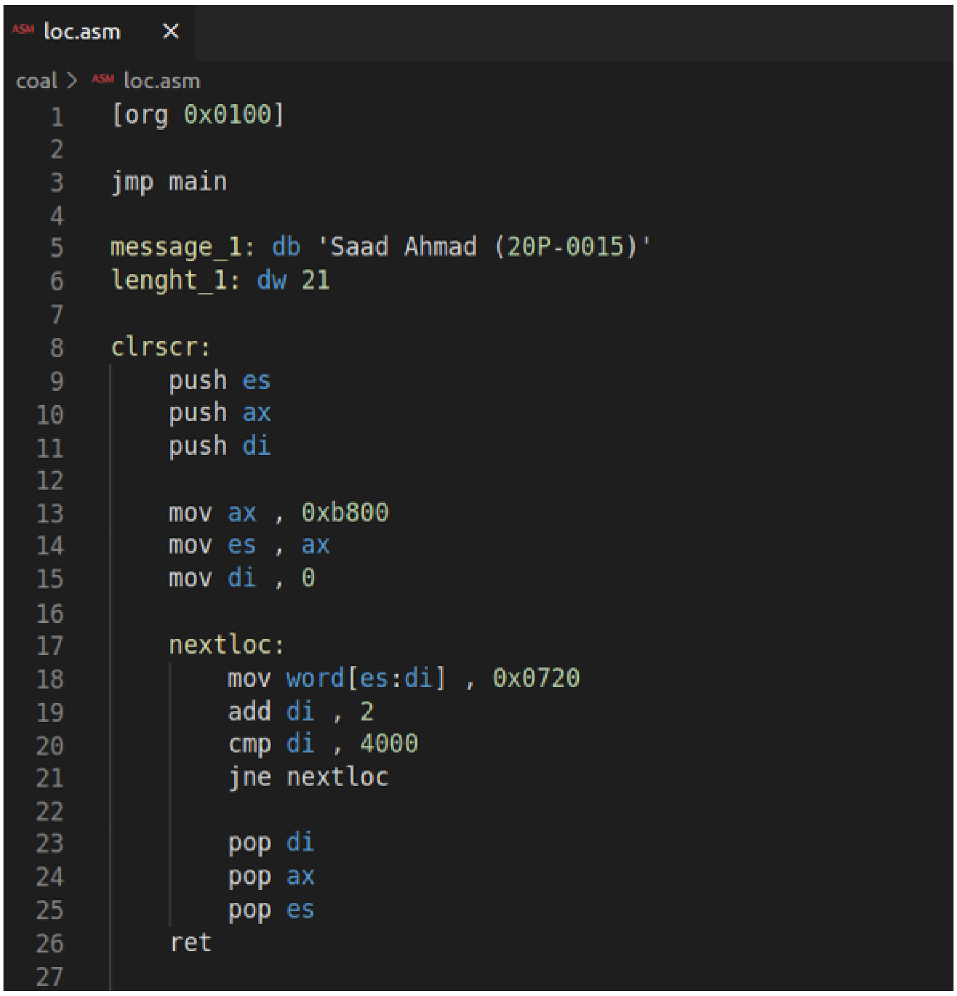
Screen Location Calculation:

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.

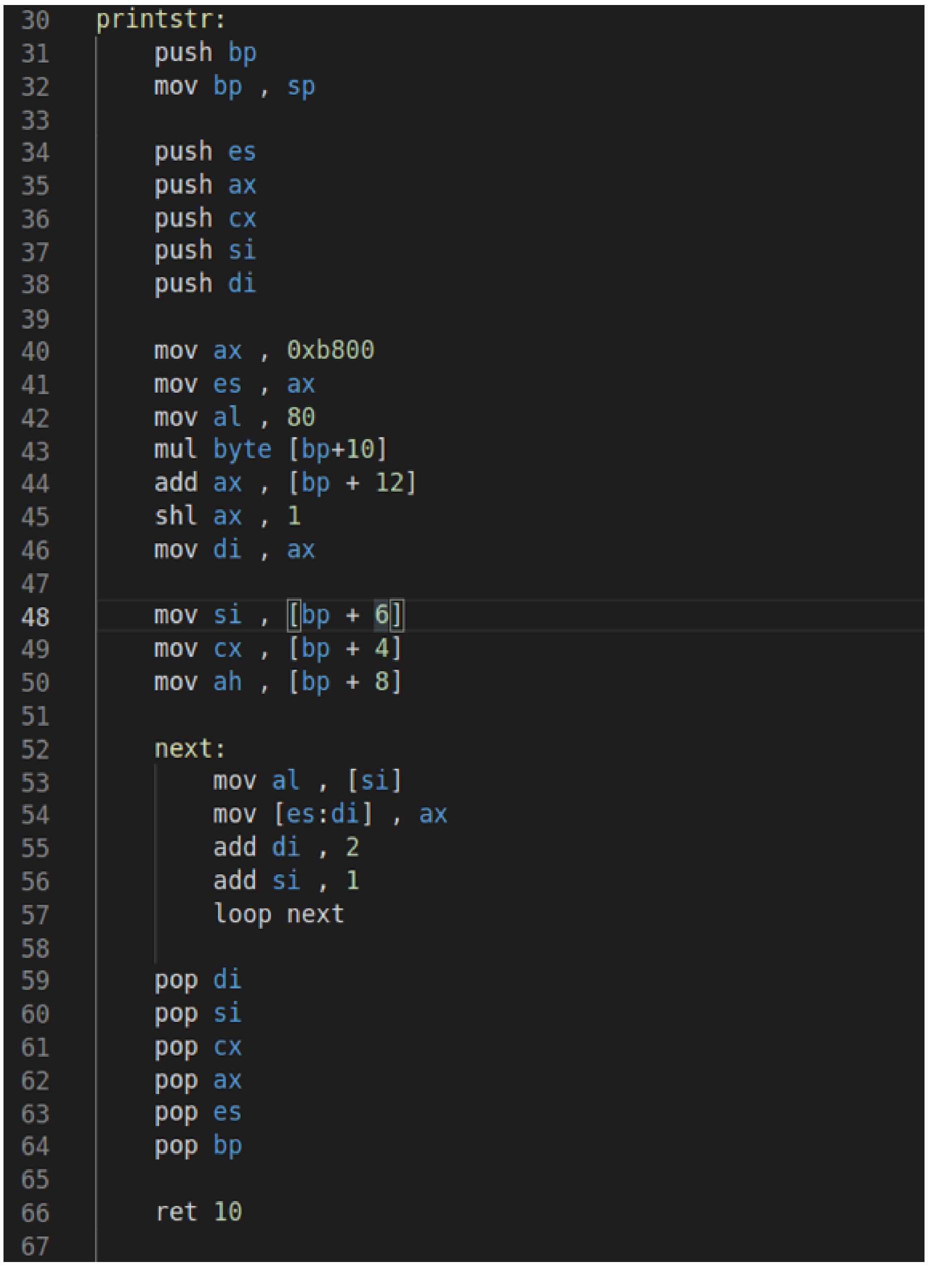
Formula:

location = (hypos * 80 + epos) * 2

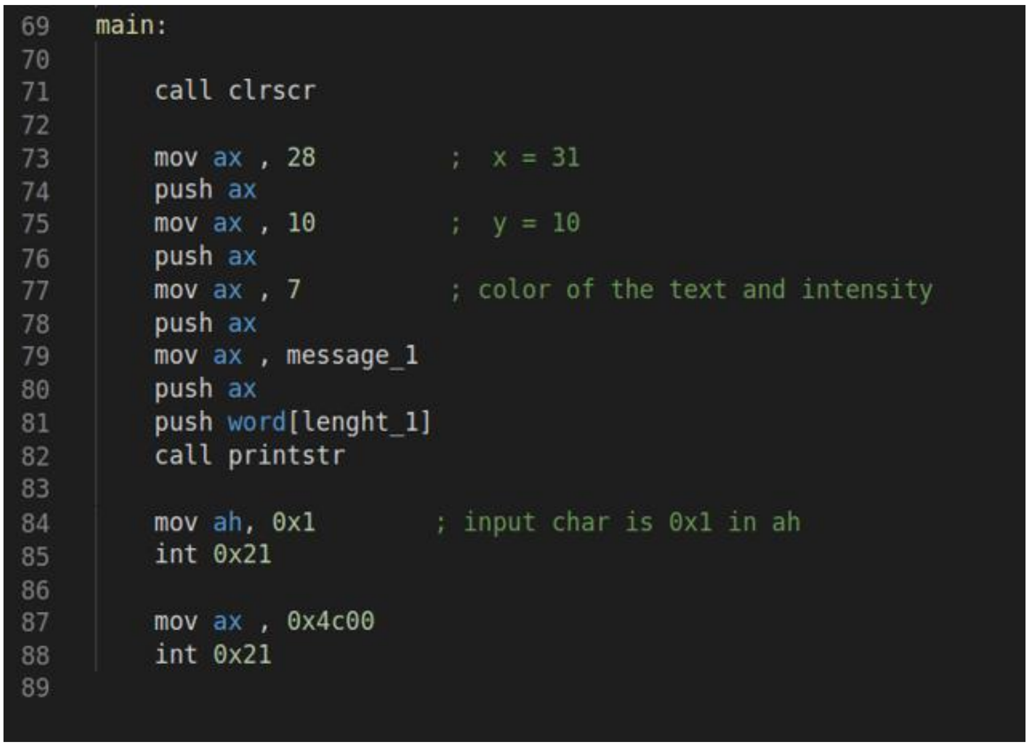
Code:



```
[org 0x0100]

jmp main

message_1: db 'Saad Ahmad (20P-0015)'
lenght_1: dw 21

clrscr:
    push es
    push ax
    push di

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

    nextloc:
        mov word[es:di] , 0x0720
        add di , 2
        cmp di , 4000
        jne nextloc

        pop di
        pop ax
        pop es
    ret

```

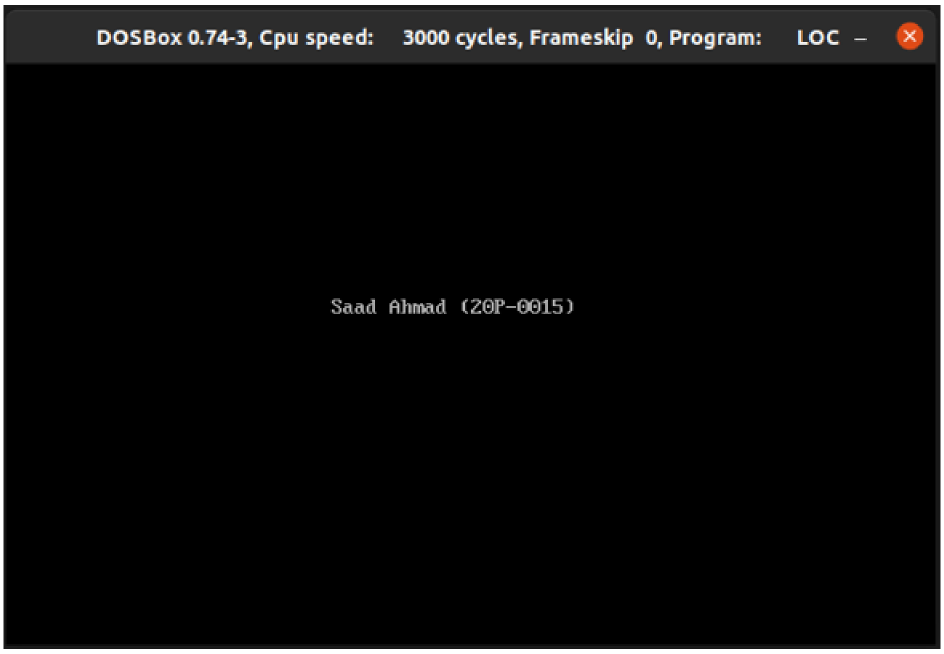


```
printstr:
    push bp
    mov bp , sp

    push es
    push ax
    push cx
    push si
    push di

    mov ax , 0xb800
    mov es , ax
    mov al , 80
    mul byte [bp+10]
    add ax , [bp + 12]
    shl ax , 1
    mov di , ax

    mov si , [bp + 6]
    mov cx , [bp + 4]
    mov ah , [bp + 8]

    next:
        mov al , [si]
        mov [es:di] , ax
        add di , 2
        add si , 1
        loop next

    pop di
    pop si
    pop cx
    pop ax
    pop es
    pop bp

    ret 10

```



```
main:

    call clrscr

    mov ax , 28          ;  x = 31
    push ax
    mov ax , 10          ;  y = 10
    push ax
    mov ax , 7           ; color of the text and intensity
    push ax
    mov ax , message_1
    push ax
    push word[lenght_1]
    call printstr

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

    mov ax , 0x4c00
    int 0x21

```

Output:



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

```
Saad Ahmad (20P-0015)
```