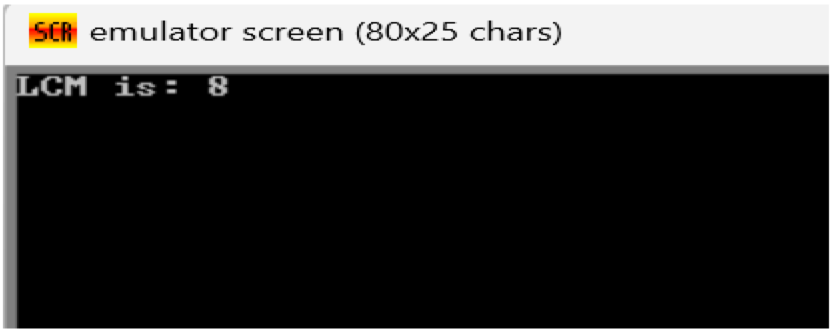
# LAB ASSIGNMENT 11

## 1. (a) Write a program in assembly language to find L.C.M of two single-digit numbers.

```
.model small
.stack 100h
.data
    num1 db 8         ; First number (single byte)
    num2 db 4         ; Second number (single byte)
    gcd_res db 0      ; To store GCD result (single byte)
    lcm_res dw 0      ; To store LCM result (two bytes for larger result)
    msg db 'LCM is: $' ; Message to display before the result
.code
main:
    mov ax, @data
    mov ds, ax        ; Initialize data segment
    ; Load num1 and num2 into AL and BL for GCD calculation
    mov al, num1
    mov bl, num2
    call gcd          ; Calculate GCD of num1 and num2
    mov gcd_res, al    ; Store GCD in gcd_res
    ; Calculate LCM using (num1 * num2) / GCD
    mov al, num1      ; Load num1 into AL
    mov ah, 0         ; Clear AH for 16-bit multiplication
    mov dl, num2      ; Load num2 into DL
    mul dl            ; AX = num1 * num2 (result in AX)
    ; Divide AX by the GCD (stored in gcd_res)
    mov cl, gcd_res    ; Load GCD into CL
    div cl            ; AX = (num1 * num2) / GCD

    ; Store the result in lcm_res

    mov lcm_res, ax

    ; Display "LCM is: "

    mov ah, 09h       ; DOS interrupt to display string

    lea dx, msg       ; Load the address of the message into DX

    int 21h

    ; Display the LCM result (convert to ASCII and print)

    mov ax, lcm_res   ; Load LCM result into AX

    call print_num    ; Call function to print number

    ; End the program

    mov ah, 4Ch

    int 21h

; Function to calculate GCD using the Euclidean algorithm

gcd proc

    cmp bl, 0

    je end_gcd        ; If BL = 0, GCD is in AL

gcd_loop:

    mov ah, 0

    div bl            ; Divide AL by BL, remainder in AH

    mov al, bl        ; Move BL to AL (new A)

    mov bl, ah        ; Move remainder to BL (new B)

    cmp bl, 0

    jne gcd_loop      ; Repeat until remainder (B) = 0

end_gcd:

    ret               ; Final GCD is in AL

gcd endp


; Function to print a number in AX
```

```
print_num proc
   ; Divide the number by 10 and print each digit
   mov cx, 0        ; Clear CX (will store digits)
   mov bx, 10       ; Divisor for base-10
convert_loop:
   xor dx, dx       ; Clear DX before division
   div bx           ; AX / 10, quotient in AX, remainder in DX
   push dx          ; Save remainder (digit) on the stack
   inc cx           ; Increment digit count
   cmp ax, 0
   jne convert_loop   ; Repeat until the quotient is 0
print_digits:
   pop dx           ; Get digit from stack
   add dl, '0'      ; Convert digit to ASCII
   mov ah, 02h      ; DOS interrupt to print character
   int 21h
   loop print_digits  ; Repeat for all digits
   ret
print_num endp
end main
```

Output:



```
LCM is: 8
```

## 1.(b) Write an assembly language program to display the nth term

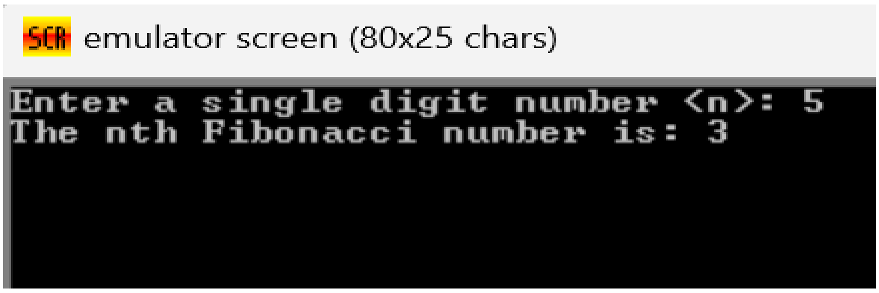
**of a fibonacci series. “n” must be a single digit number which may be taken from the user.**

```
.model small
.stack 100h
.data
    prompt db 'Enter a single digit number <n>: $'
    result_msg db 0Dh,0Ah,'The nth Fibonacci number is: $'
    fib dw 0       ; Store the nth Fibonacci number in a word (16 bits)
.code
main proc
    ; Initialize data segment
    mov ax, @data
    mov ds, ax
    ; Prompt the user for input
    mov ah, 09h
    lea dx, prompt
    int 21h
    ; Read a single character input
    mov ah, 01h
    int 21h
    sub al, '0'    ; Convert ASCII to integer
    mov cl, al     ; Store n in cl
    ; Check for n = 0 or n = 1 directly
    cmp cl, 1
    jbe single_digit_fib
    ; For n > 1, calculate Fibonacci using loop
    ; Initialize Fibonacci values
    mov ax, 0      ; First Fibonacci number (16-bit for larger values)
    mov bx, 1      ; Second Fibonacci number (16-bit)
```

```
fib_loop:
    dec cl        ; Decrease count
    jz store_result ; If count reaches zero, store result
    ; Calculate next Fibonacci number
    add ax, bx     ; F_n = F_(n-1) + F_(n-2)
    xchg ax, bx    ; Move F_(n-1) to F_(n-2) and update F_(n-1)
    jmp fib_loop   ; Repeat loop until cl = 0
store_result:
    mov fib, ax    ; Store the result in fib
single_digit_fib:
    ; For n = 0 or 1, bx already contains the correct Fibonacci number
    cmp cl, 0
    je show_fib0
    mov fib, bx    ; For n=1, F_1 is 1
    jmp display_result
show_fib0:
    mov fib, ax    ; For n=0, F_0 is 0
display_result:
    ; Display result message
    mov ah, 09h
    lea dx, result_msg
    int 21h
    ; Convert the result in fib to ASCII and display
    mov ax, fib       ; Load result into ax
    call print_number  ; Call subroutine to print the number
    ; Exit program
    mov ah, 4Ch
    int 21h
main endp
```

```
; Subroutine to print a number in AX as ASCII
print_number proc
   ; Divide ax by 10 repeatedly to extract each digit in reverse
   mov cx, 10       ; Set base to 10
   mov bx, 0        ; Initialize bx as digit storage
reverse_digits:
   xor dx, dx       ; Clear dx for division
   div cx           ; AX / 10, quotient in AX, remainder in DX
   push dx          ; Push remainder onto stack (digit)
   inc bx           ; Count digits
   test ax, ax      ; Check if quotient is 0
   jnz reverse_digits
display_digits:
   pop dx           ; Get last pushed digit
   add dl, '0'      ; Convert to ASCII
   mov ah, 02h      ; DOS print character function
   int 21h          ; Display character
   dec bx           ; Decrement digit count
   jnz display_digits
   ret
print_number endp
end main
```

Output:



```
Enter a single digit number (n): 5
The nth Fibonacci number is: 3
```

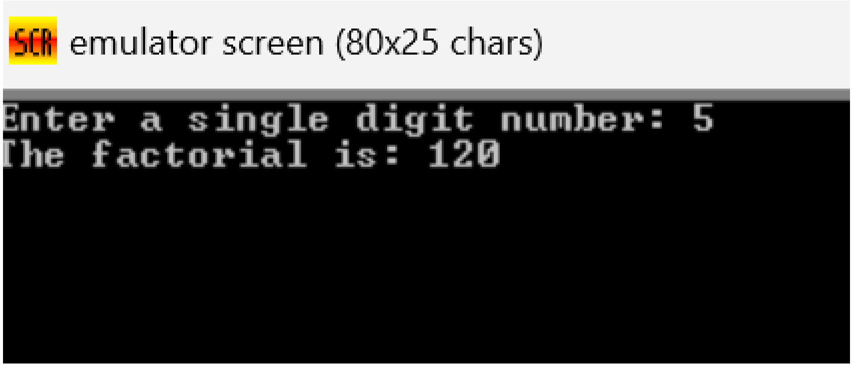
## 2. Write an assembly language program to find the factorial of a

**given single-digit number.**

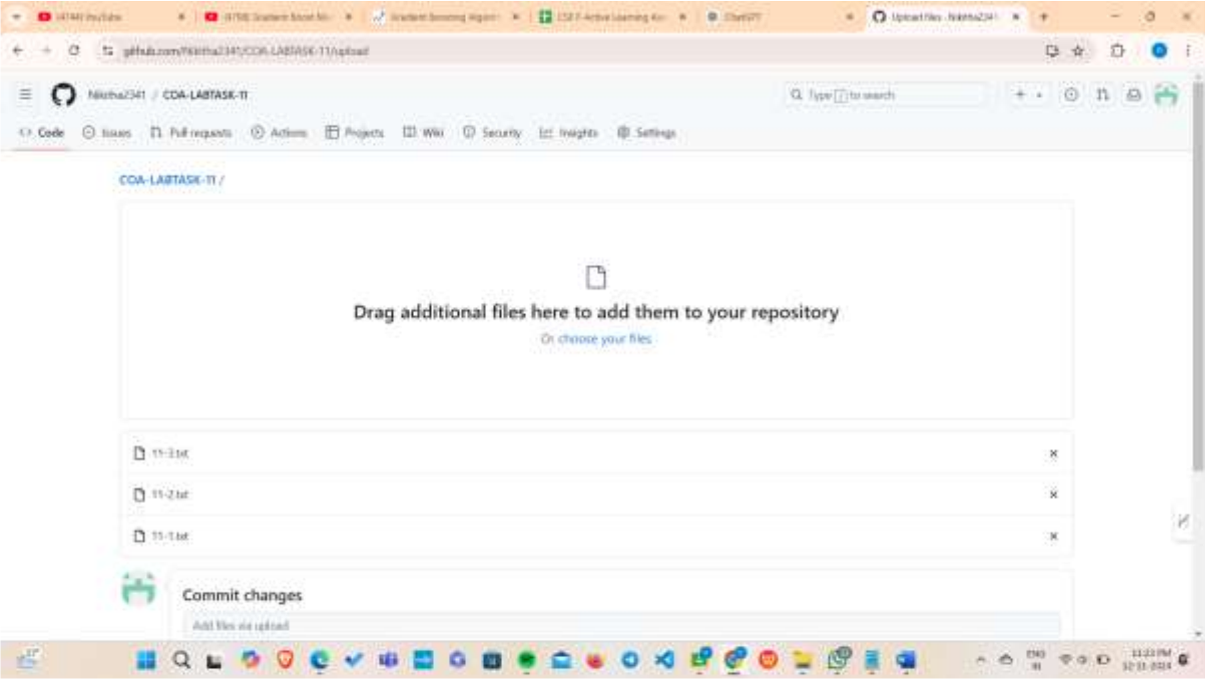
```
.model small
.stack 100h
.data
   prompt db 'Enter a single digit number: $'
   result_msg db 0Dh,0Ah,'The factorial is: $'
   factorial dw 1    ; 16-bit variable to store factorial result
.code
main proc
   ; Initialize data segment
   mov ax, @data
   mov ds, ax
   ; Display prompt to enter a number
   mov ah, 09h
   lea dx, prompt
   int 21h
   ; Read a single character input
   mov ah, 01h
   int 21h
   sub al, '0'       ; Convert ASCII to integer
   mov bl, al        ; Store the number in BL for calculation
   ; Special case for 0! which is 1
   cmp bl, 0
   jne calculate_factorial
   mov factorial, 1
   jmp display_result
calculate_factorial:
   mov cx, bx        ; Set loop counter to the number entered (n)
   mov ax, 1         ; AX will store the ongoing factorial result
```

```
factorial_loop:
    mul cx              ; AX = AX * CX (calculate factorial)
    loop factorial_loop ; Decrement CX and repeat until CX = 0
    mov factorial, ax  ; Store final factorial result in 'factorial'
display_result:
    ; Display result message
    mov ah, 09h
    lea dx, result_msg
    int 21h
    ; Convert the result in factorial to ASCII and display
    mov ax, factorial  ; Load factorial result into AX
    call print_number  ; Call subroutine to print the number
    ; Exit program
    mov ah, 4Ch
    int 21h
main endp
; Subroutine to print a number in AX as ASCII
print_number proc
    ; Divide ax by 10 repeatedly to extract each digit in reverse
    mov cx, 10        ; Set base to 10
    mov bx, 0         ; Initialize bx as digit storage
reverse_digits:
    xor dx, dx        ; Clear dx for division
    div cx            ; AX / 10, quotient in AX, remainder in DX
    push dx           ; Push remainder onto stack (digit)
    inc bx            ; Count digits
    test ax, ax       ; Check if quotient is 0
    jnz reverse_digits
display_digits:
    pop dx          ; Get last pushed digit
    add dl, '0'      ; Convert to ASCII
    mov ah, 02h       ; DOS print character function
    int 21h          ; Display character
    dec bx           ; Decrement digit count
    jnz display_digits
    ret
print_number endp
end main
```

Output:



```
Enter a single digit number: 5
The factorial is: 120
```



Nikitha2341 / COA-LABTASK-11

COA-LABTASK-11 /

Drag additional files here to add them to your repository

Or choose your files

11-3.txt

11-2.txt

11-1.txt

Commit changes

Add files via upload

Github Link:  https://github.com/Nikitha2341/COA-LABTASK-11/upload