LAB TASK 11  
1. (a) Write a program in assembly language to find L.C.M of two

single-digit numbers.  
CODE:  
.model small

.stack 100h

.data

num1 db 6 ; 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

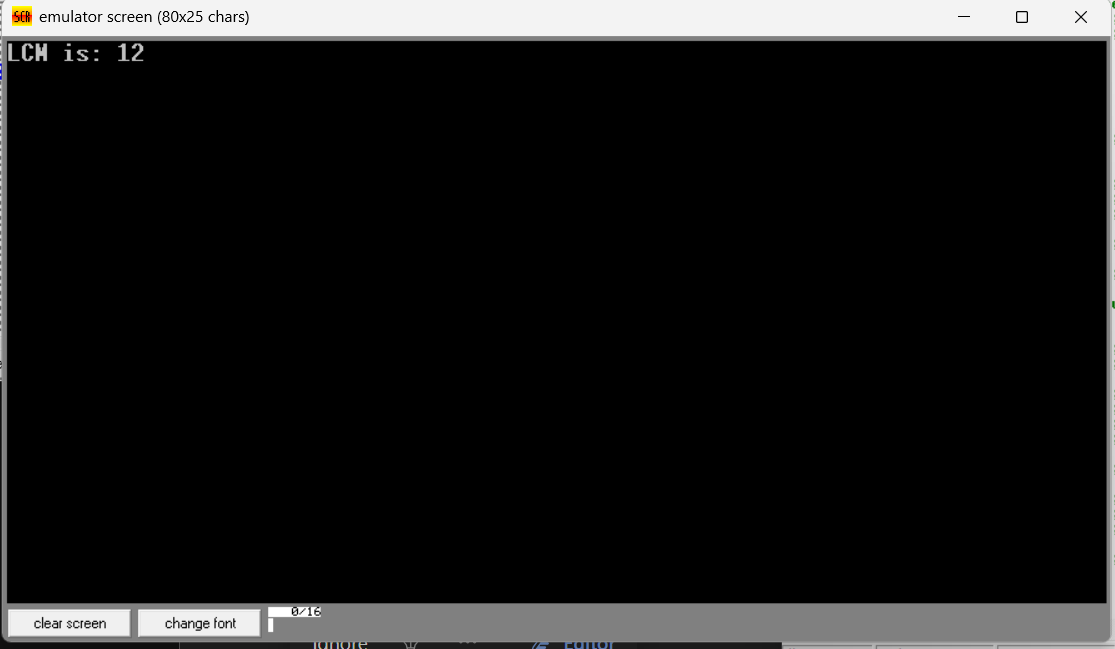
int 21h

loop print\_digits ; Repeat for all digits

ret

print\_num endp

end main

OUTPUT:::::  
  
(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.  
CODE:  
.model small

.stack 100h

.data

prompt db 'Enter a single digit number (1-9) for 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 (1-9)

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

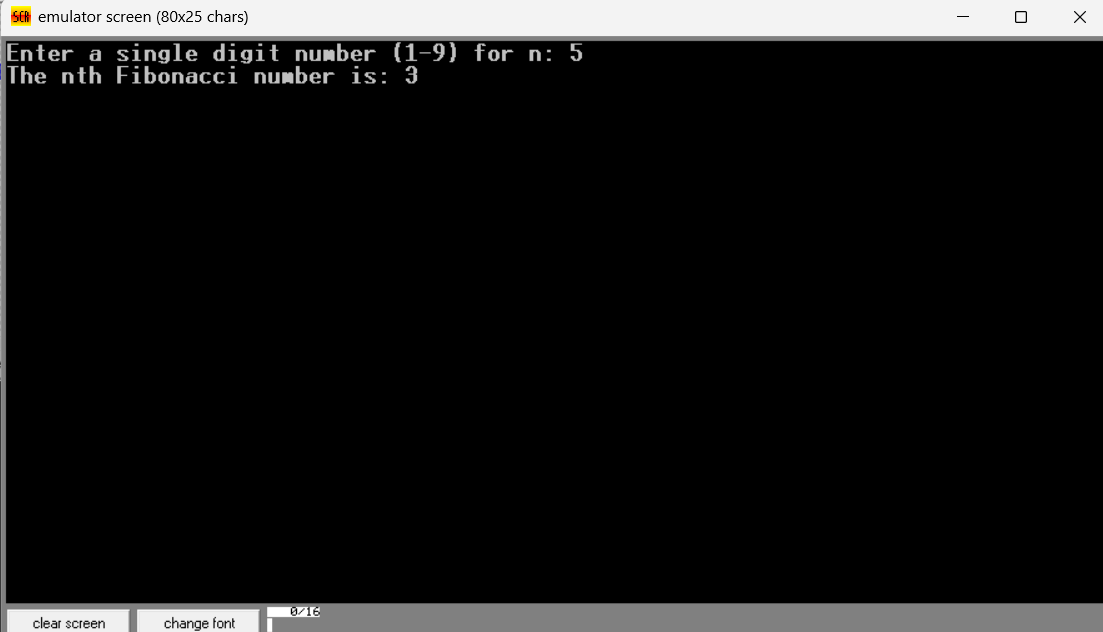
int 21h ; Display character

dec bx ; Decrement digit count

jnz display\_digits

ret

print\_number endp

end main  
  
OUTPUT:::::  
  
  
  
2. Write an assembly language program to find the factorial of a given single-digit number.  
CODE:  
.model small

.stack 100h

.data

prompt db 'Enter a single digit number (0-9): $'

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:::::  
