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

à CODE

.model small

.stack 100h .data

num1 db 48 ; First number (single byte) num2 db 18 ; 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\_gcd db 'GCD: $'

msg\_lcm db 'LCM: $'

.code main:

mov ax, @data

mov ds, ax ; Initialize data segment

; Display message for GCD

mov ah, 09h ; DOS function to display string lea dx, msg\_gcd int 21h

; 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

; Display GCD result mov al, gcd\_res

call display\_result

; 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 message for LCM

mov ah, 09h ; DOS function to display string lea dx, msg\_lcm

int 21h

; Display LCM result mov ax, lcm\_res

call display\_result

; 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 display a number in AX as decimal display\_result proc

mov bx, 10 ; Divisor for decimal conversion

xor cx, cx ; Clear CX to use as counter for digits

convert\_loop:

xor dx, dx ; Clear DX for division

div bx ; Divide AX by 10, remainder in DX (last digit) push dx ; Push remainder onto stack inc cx ; Increment digit counter cmp ax, 0 ; Check if quotient is 0

jne convert\_loop ; If not, continue dividing

print\_digits:

pop dx ; Pop digit from stack add dl, '0' ; Convert to ASCII

mov ah, 02h ; DOS function to display character

int 21h ; Display digit

loop print\_digits ; Repeat for all digits

ret

display\_result endp

end main

àoutput



(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.

àCODE

.model small

.stack 100h .data

msg db 'Enter the value of n (0-9): $' ; Message to prompt user fib\_res db ? ; To store nth Fibonacci term

n db ? ; User input (single-digit number)

result\_msg db 0Dh, 0Ah, 'Fibonacci term: $' ; Message to display result result db '00$', 0Dh, 0Ah ; Space to store result as string

.code main:

mov ax, @data

mov ds, ax ; Initialize data segment

; Display message to enter the value of n mov ah, 09h lea dx, msg

int 21h

; Take single-digit input from user mov ah, 01h int 21h

sub al, '0' ; Convert ASCII to integer mov n, al ; Store user input in 'n'

; Check if input is 0 or 1 mov al, n cmp al, 0

je fib\_zero ; If n = 0, set result to 0 cmp al, 1

je fib\_one ; If n = 1, set result to 1

; Initialize Fibonacci terms for calculation mov cl, al ; Move n to CL for loop count mov al, 1 ; Set AL = 1 for F(1) mov bl, 0 ; Set BL = 0 for F(0)

dec cl ; Adjust count to loop n-1 times

fib\_loop:

; Calculate next term: F(n) = F(n-1) + F(n-2) mov ah, al ; Store current F(n-1) in AH add al, bl ; AL = F(n) = F(n-1) + F(n-2) mov bl, ah ; Update F(n-2) to previous F(n-1) dec cl

jnz fib\_loop ; Loop until CL becomes zero (reached nth term)

; Store the nth Fibonacci term in fib\_res mov fib\_res, al

display\_result:

; Display result message mov ah, 09h lea dx, result\_msg int 21h

; Convert result to ASCII and store in 'result' for correct display

mov al, fib\_res

aam ; Split AL into AH (tens) and AL (units) add ah, '0' ; Convert tens to ASCII add al, '0' ; Convert units to ASCII mov result[0], ah ; Store tens digit in result mov result[1], al ; Store units digit in result jmp display\_final

single\_digit:

add al, '0' ; Convert single digit to ASCII mov result[0], al ; Store single digit in result mov result[1], '$' ; Add end-of-string marker

display\_final: ; Display the result lea dx, result mov ah, 09h

int 21h

; End the program

mov ah, 4Ch int 21h

fib\_zero:

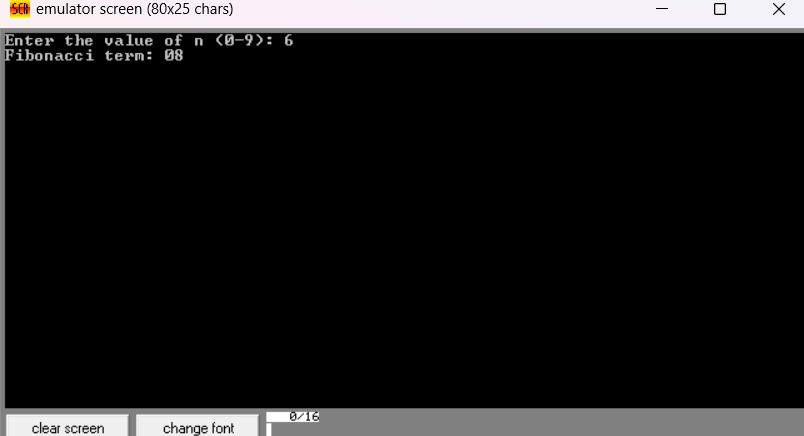
mov fib\_res, 0 ; F(0) = 0

jmp display\_result

fib\_one:

mov fib\_res, 1 ; F(1) = 1

output



Practice set:

2. Write an assembly language program to find the factorial of a given single-digit number.

à CODE

.MODEL SMALL

.STACK 100H

.DATA

msg db 'Enter a single-digit number (0-9): $' ; Prompt message for input

result\_msg db 0Dh, 0Ah, 'Factorial: $' ; Message to display the result

result db '00000$', 0Dh, 0Ah ; Space to store factorial result as a string

num db ? ; Variable to store the input number

fact dw 1 ; Variable to store the factorial result

.CODE

main:

; Initialize data segment

mov ax, @data

mov ds, ax

; Display prompt message

mov ah, 09h

lea dx, msg

int 21h

; Take single-digit input from user

mov ah, 01h

int 21h

sub al, '0' ; Convert ASCII to integer

mov num, al ; Store user input in 'num'

; Initialize factorial calculation

mov al, num

mov ah, 0 ; Clear AH to extend AL to AX

mov cx, ax ; Move AX to CX (counter)

mov ax, 1 ; Initialize AX to 1 (factorial result)

factorial\_loop:

cmp cx, 1 ; Compare CX to 1

je end\_factorial\_loop ; If CX is 1, end the loop

mul cx ; Multiply AX by CX

loop factorial\_loop ; Decrement CX and repeat the loop

end\_factorial\_loop:

; Store the factorial result in 'fact'

mov fact, ax

display\_factorial:

; Display result message

mov ah, 09h

lea dx, result\_msg

int 21h

; Convert the factorial result to ASCII

mov ax, fact

mov cx, 10 ; Prepare divisor (10) for unpacking digits

lea di, result + 4 ; Start storing result from the end

convert\_to\_ascii:

xor dx, dx ; Clear DX for division

div cx ; AX = AX / 10, DX = remainder (last digit)

add dl, '0' ; Convert remainder to ASCII

mov [di], dl ; Store ASCII character in result

dec di ; Move to the next character position

cmp ax, 0 ; Check if quotient is zero

jne convert\_to\_ascii ; Repeat if there are more digits

; Display the factorial result

lea dx, result

mov ah, 09h

int 21h

; End the program

mov ah, 4Ch

int 21h

end main // Fcatorial ke correct code

output

