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

***single-digit numbers.***

**CODE:**

.STACK 100h

.DATA

num1 DB 4 ; First number (single byte)

num2 DB 6 ; 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)

resultMsg DB 'The LCM is: $' ; Message to display

.CODE

main:

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Load num1 and num2 into AL and BL for GCD calculation

MOV AL, num1 ; Move the first number to AL

MOV BL, num2 ; Move the second number to BL

CALL gcd ; Call GCD procedure to calculate GCD of num1 and num2

MOV gcd\_res, AL ; Store the calculated GCD result in gcd\_res

; Calculate LCM using the formula: (num1 \* num2) / GCD

MOV AL, num1 ; Load num1 into AL for multiplication

MOV AH, 0 ; Clear AH (required for 16-bit multiplication)

MOV DL, num2 ; Load num2 into DL

MUL DL ; Multiply AL by DL, result is stored in AX (num1 \* num2)

; Divide the product in AX by the GCD to find LCM

MOV CL, gcd\_res ; Load the GCD into CL

DIV CL ; Divide AX by GCD, result is in AL (remainder in AH)

; Store the LCM result in lcm\_res

MOV lcm\_res, AX ; Save the result of (num1 \* num2) / GCD in lcm\_res

; Print the result message

LEA DX, resultMsg ; Load the address of resultMsg into DX

MOV AH, 09h ; DOS function to print a string

INT 21h ; Display the result message

; Display the LCM value in decimal

MOV AX, lcm\_res ; Load the LCM result into AX for display

CALL PrintDecimal ; Call PrintDecimal procedure to print AX as a decimal

; End the program

MOV AH, 4Ch ; DOS function to terminate program

INT 21h

; Procedure to calculate GCD using the Euclidean algorithm

gcd PROC

; Check if BL is zero, which means the GCD is already in AL

CMP BL, 0

JE end\_gcd ; If BL is zero, exit gcd procedure (GCD is in AL)

gcd\_loop:

MOV AH, 0 ; Clear AH for division

DIV BL ; Divide AL by BL, remainder goes into AH

MOV AL, BL ; Move BL into AL (next dividend)

MOV BL, AH ; Move remainder into BL (next divisor)

CMP BL, 0 ; Check if remainder is zero

JNE gcd\_loop ; If remainder is not zero, repeat loop

end\_gcd:

RET ; GCD is in AL when BL becomes zero

gcd ENDP

; Procedure to print a 16-bit number in AX as a decimal number

PrintDecimal PROC

; This procedure converts the number in AX to decimal and prints each digit

MOV CX, 10 ; Set divisor to 10 for decimal system

MOV BX, 0 ; Initialize BX to count number of digits

decimal\_loop:

XOR DX, DX ; Clear DX before division

DIV CX ; Divide AX by 10, quotient in AX, remainder in DX

PUSH DX ; Push remainder (digit) onto the stack

INC BX ; Increase digit count

CMP AX, 0 ; Check if quotient is zero

JNE decimal\_loop ; If quotient is not zero, continue loop

print\_digits:

POP DX ; Retrieve digits from stack

ADD DL, '0' ; Convert the digit to ASCII character

MOV AH, 02h ; DOS function to print character

INT 21h ; Display the character

DEC BX ; Decrease digit counter

JNZ print\_digits ; Repeat until all digits are printed

RET

PrintDecimal ENDP

END main

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**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.**

**CODE:**

.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

jmp display\_result

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**Practice set:**

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

**given single-digit number.**

**CODE:**

.STACK 100h

.DATA

promptMsg DB 'Enter a single-digit number (0-9): $' ; Prompt message for user input

resultMsg DB 'The factorial is: $' ; Message to display result

num DB ? ; Variable to store the user input

factorial DW 1 ; To store the factorial result, initialized to 1

.CODE

main:

; Initialize data segment

MOV AX, @DATA

MOV DS, AX

; Display the prompt message

LEA DX, promptMsg

MOV AH, 09h ; DOS function to display string

INT 21h

; Take a single-digit number as input from the user

MOV AH, 01h ; DOS function to take single character input

INT 21h

SUB AL, '0' ; Convert ASCII to integer

MOV num, AL ; Store input number in 'num'

; Check if the number is 0 (since 0! = 1)

CMP num, 0

JE display\_result ; If num = 0, factorial is 1, skip calculation

; Initialize factorial calculation

MOV AX, 1 ; AX will hold the running factorial product

MOV CL, num ; Load the input number into CL as the counter

factorial\_loop:

MOV BX, AX ; Move the current factorial value to BX for multiplication

MUL CL ; Multiply AX by CL (AX = AX \* CL)

DEC CL ; Decrement CL by 1

JNZ factorial\_loop ; Repeat until CL = 0

; Store the result in factorial variable

MOV factorial, AX ; Store AX (factorial result) in factorial

display\_result:

; Display the result message

LEA DX, resultMsg

MOV AH, 09h ; DOS function to display string

INT 21h

; Display the factorial result

MOV AX, factorial ; Load factorial result into AX for display

CALL PrintDecimal ; Call PrintDecimal to display the number

; End the program

MOV AH, 4Ch ; DOS function to exit program

INT 21h

; Procedure to print a 16-bit number in AX as decimal

PrintDecimal PROC

MOV CX, 10 ; Set divisor to 10 for decimal conversion

MOV BX, 0 ; Initialize digit count

decimal\_loop:

XOR DX, DX ; Clear DX for division

DIV CX ; Divide AX by 10, quotient in AX, remainder in DX

PUSH DX ; Push remainder (digit) onto stack

INC BX ; Count the digit

CMP AX, 0 ; Check if AX is zero

JNE decimal\_loop ; Repeat until AX is zero

print\_digits:

POP DX ; Get digit from stack

ADD DL, '0' ; Convert to ASCII

MOV AH, 02h ; DOS function to print character

INT 21h ; Display the character

DEC BX ; Decrement digit counter

JNZ print\_digits ; Repeat until all digits are printed

RET

PrintDecimal ENDP

END main

**OUTPUT:**

A screenshot of a computer

Description automatically generated