Shaik Sharifa Jahan

AP22110011103

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

single-digit numbers.

org 100h

jmp start

input1 db "Enter the First Digit: $"

input2 db 0Dh,0Ah,"Enter the Second Digit: $"

output db 0Dh,0Ah,"LCM of the Given two numbers is: $"

start:

; Display "Enter the First Digit:"

mov dx, offset input1

mov ah, 09h

int 21h

; Read first digit

mov ah, 01h

int 21h

sub al, '0' ; Convert ASCII to numerical value

mov cl, al ; Store first number in CL

; Display "Enter the Second Digit:"

mov dx, offset input2

mov ah, 09h

int 21h

; Read second digit

mov ah, 01h

int 21h

sub al, '0' ; Convert ASCII to numerical value

mov ch, al ; Store second number in CH

; Copy CL and CH to registers for GCD calculation

mov al, cl ; AL = first number

mov bl, ch ; BL = second number

GCD\_LOOP:

cmp bl, 0

je DONE\_GCD

mov ah, 0

div bl ; Divide AL by BL, remainder in AH, quotient in AL

mov al, bl

mov bl, ah

jmp GCD\_LOOP

DONE\_GCD:

; Now AL contains the GCD

mov bh, al ; Store GCD in BH

; Calculate LCM = (CL \* CH) / GCD

mov al, cl ; AL = first number

mul ch ; AX = AL \* CH (product of the two numbers)

mov bl, bh ; Move GCD to BL

div bl ; AX / GCD, result in AL

mov bl, al

; Display "LCM of the Given two numbers is:"

mov dx, offset output

mov ah, 09h

int 21h

mov ax, 0 ; Clear AX register

mov al, bl ; Move the result from BL to AL

mov cx, 10 ; Set divisor to 10 (for decimal conversion)

dec\_to\_ascii:

xor dx, dx ; Clear DX for division

div cx ; AX / 10 -> Quotient in AL, Remainder in DL

add dl, '0' ; Convert remainder to ASCII

push dx ; Store remainder on stack

test al, al ; Check if quotient is zero

jnz dec\_to\_ascii ; Repeat if quotient is not zero

print\_decimal:

pop dx ; Get character from stack

mov ah, 02h ; DOS function to display character in DL

int 21h ; Print character

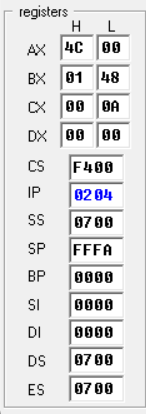
cmp sp, 0 ; Check if stack is empty

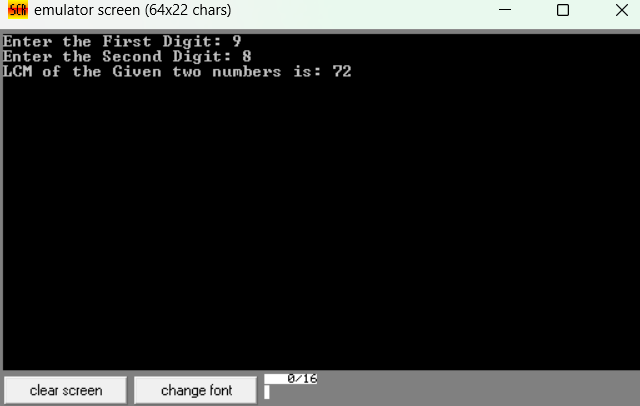
jne print\_decimal ; Repeat if stack not empty

; Exit program

mov ah,4Ch

int 21h





(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 (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

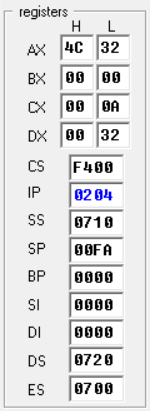
dec bx ; Decrement digit count

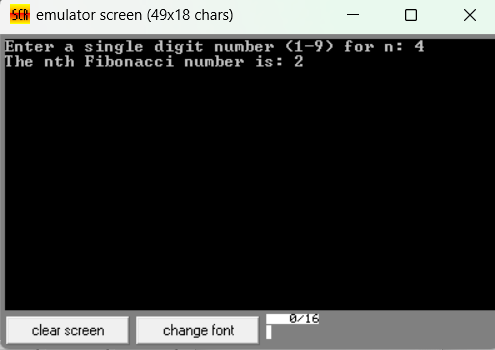
jnz display\_digits

ret

print\_number endp

end main





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 (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

