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 ? ; First number (input by user)
num2 db ? ; Second number (input by user)
gcd_res db 0 ; To store GCD result (single byte)
lcm_res dw 0 ; To store LCM result (two bytes for larger result)
msg_num1 db 'Enter first single-digit number: $'
msg_num2 db 0Dh, 0Ah, 'Enter second single-digit number: $'
msg_gcd db 0Dh, 0Ah, 'GCD: $'
msg_lcm db 0Dh, 0Ah, 'LCM: $'
.CODE
main:
mov ax, @data
mov ds, ax ; Initialize data segment
; Prompt for first number
mov ah, 09h ; DOS function to display string
lea dx, msg_num1
int 21h
; Read first number
mov ah, 01h ; DOS function to read a character
int 21h
sub al, '0' ; Convert ASCII to integer
mov num1, al ; Store first number in num1
; Prompt for second number
mov ah, 09h ; DOS function to display string
lea dx, msg_num2
int 21h
; Read second number
mov ah, 01h ; DOS function to read a character
```

```
int 21h
sub al, '0' ; Convert ASCII to integer
mov num2, al ; Store second number in num2
; 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
```

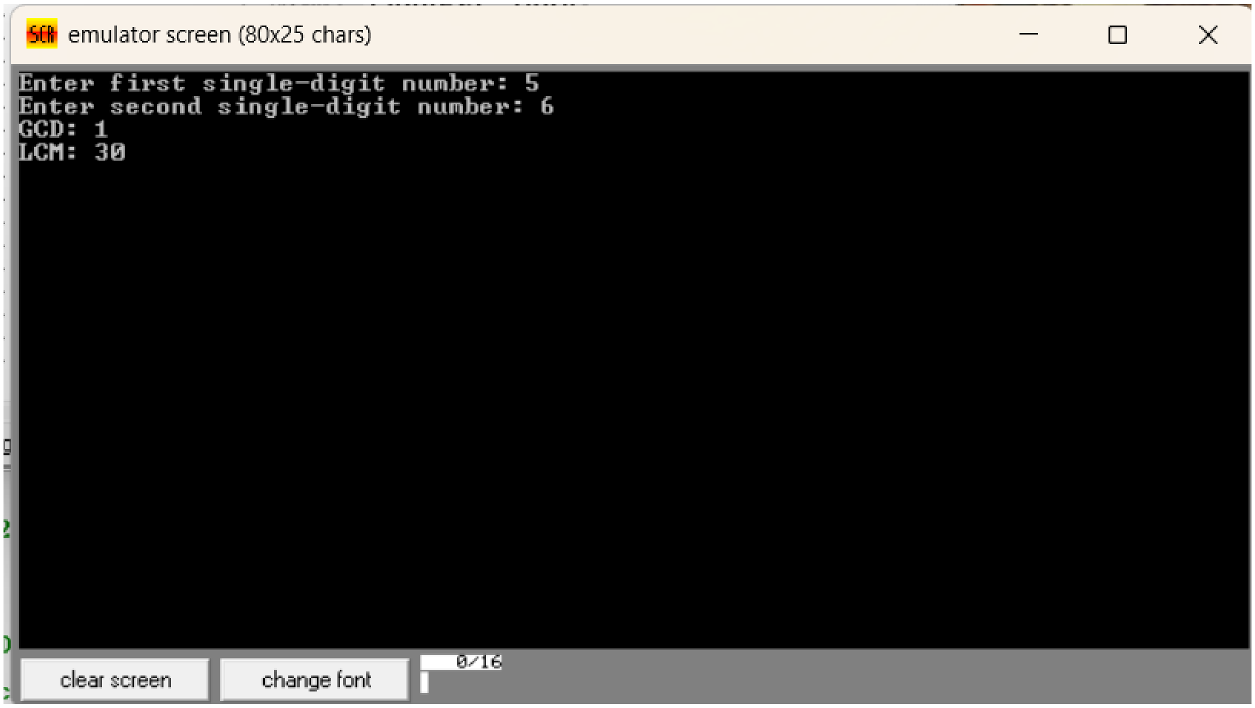
```
; End the program
mov ah, 4Ch
int 21h

; 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
```

OUTPUT:



```
Enter first single-digit number: 5
Enter second single-digit number: 6
GCD: 1
LCM: 30
```

(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)
```

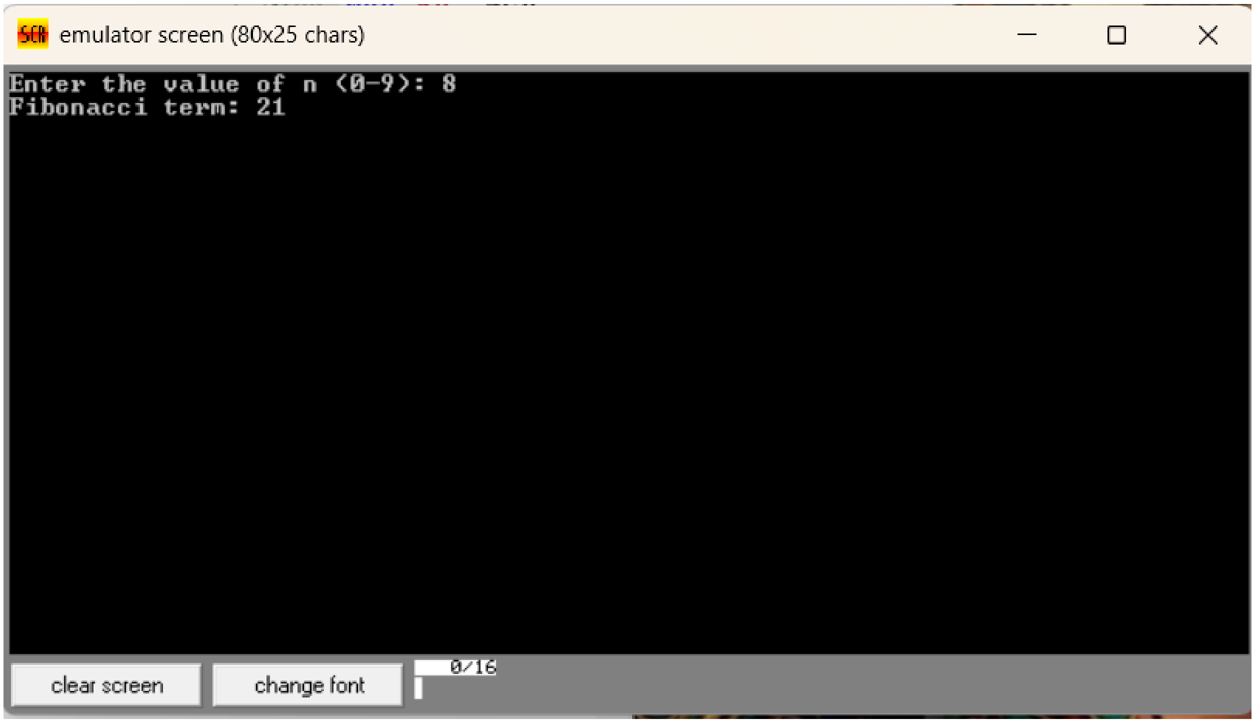
```
dec cl
jnz fib_loop ; Loop until CL becomes zero (reached nth term)
; Store the nth Fibonacci term in fib_res
mov fib_res, al
display_result:
; Display result message
mov ah, 09h

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
end main
```

OUTPUT:



```
Enter the value of n (0-9): 8
Fibonacci term: 21
```

Practice set:

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

CODE:

```
.MODEL SMALL ; Define memory model
.STACK 100H ; Define stack size (256 bytes)
.DATA
msg db 'Enter a single-digit number (0-9): $' ; Prompt message for user
input
result_msg db 0Dh, 0Ah, 'Factorial: $' ; Message to display before the result
```

```
result db '00000$', 0Dh, 0Ah ; Space to store the result
num db ? ; Variable to store the user input (single digit)
fact dw 1 ; Variable to store the factorial result (initial value 1)
.CODE
main:
; Initialize data segment
mov ax, @data ; Load the address of the data segment into AX
mov ds, ax ; Move the value of AX into DS (data segment register)
; Display prompt message
mov ah, 09h ; Set AH to 09h (DOS function to display string)
lea dx, msg ; Load the effective address of 'msg' into DX
int 21h ; Interrupt to call DOS function (display string)
; Take single-digit input from user
mov ah, 01h ; Set AH to 01h
int 21h ; Interrupt to call DOS function (get character input)
sub al, '0' ; Convert ASCII value of the input
mov num, al ; Store the converted value in 'num' variable

; Initialize factorial calculation
mov al, num ; Move the input number (in 'num') into AL register
mov ah, 0 ; Clear AH to extend AL to AX
mov cx, ax ; Move AX (the input number) into CX register
mov ax, 1 ; Initialize AX to 1 (this will hold the factorial result)
factorial_loop:
cmp cx, 1 ; Compare CX (counter) to 1
je end_factorial_loop ; If CX is 1, jump to the end of factorial loop
mul cx ; Multiply AX by CX (AX = AX * CX, result stored in AX)
loop factorial_loop ; Decrement CX and repeat the loop if CX is not zero
end_factorial_loop:
; Store the factorial result in 'fact'
mov fact, ax ; Store the final result of AX (factorial) in 'fact'
display_factorial:
```

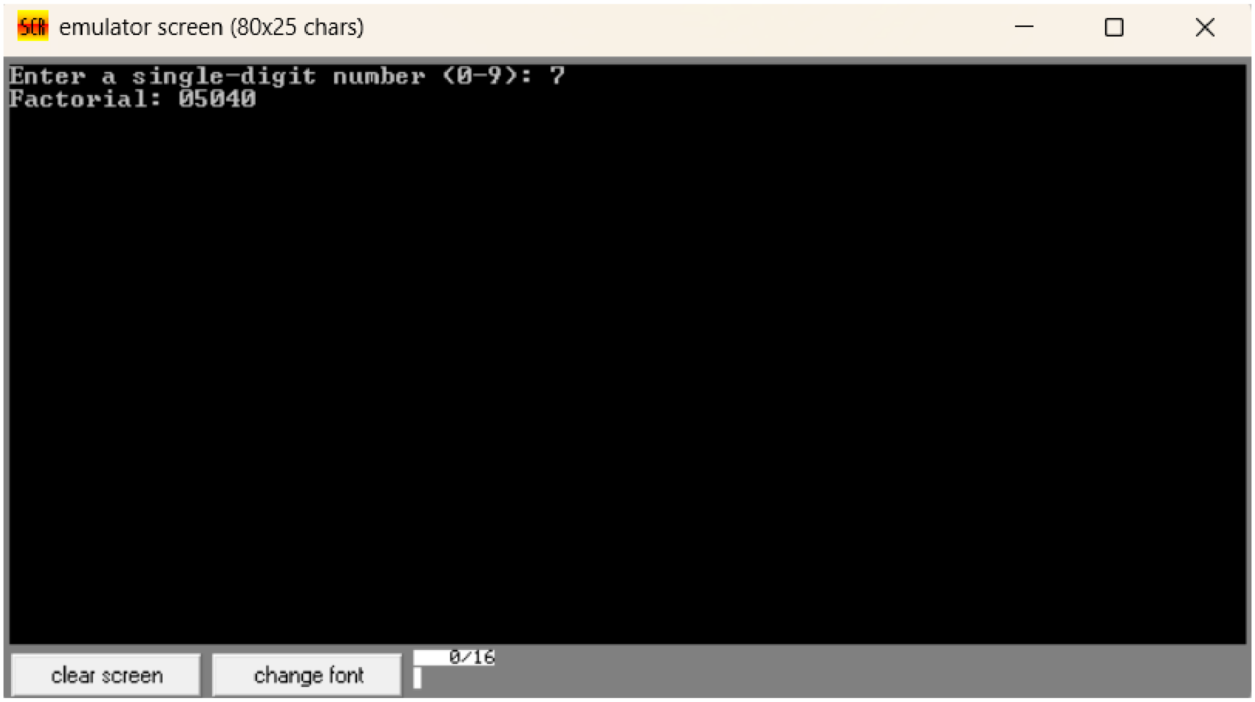
```
; Display result message
mov ah, 09h ; Set AH to 09h (DOS function to display string)
lea dx, result_msg ; Load the effective address of result_msg into DX
int 21h ; Interrupt to call DOS function (display string)
; Convert the factorial result to ASCII
mov ax, fact ; Load the factorial result from 'fact' into AX
mov cx, 10 ; Prepare divisor 10 for unpacking digits
lea di, result + 4 ; Load the address of the last position of the result

convert_to_ascii:
xor dx, dx ; Clear DX (DX will hold the remainder during division)
div cx ; Divide AX by CX (AX / 10) - quotient in AX, remainder in

add dl, '0' ; Convert the remainder (last digit) to ASCII by adding the

mov [di], dl ; Store the ASCII character in the result string
dec di ; Move DI to the next character position
cmp ax, 0 ; Compare the quotient (AX) with 0
jne convert_to_ascii ; If quotient is not 0, repeat the conversion
; Display the factorial result
lea dx, result ; Load the address of the result string into DX
mov ah, 09h ; Set AH to 09h (DOS function to display string)
int 21h ; Interrupt to call DOS function (display string)
; End the program
mov ah, 4Ch ; Set AH to 4Ch
int 21h ; Interrupt to call DOS function (terminate the program)
end main
```

OUTPUT:



```
Enter a single-digit number (0-9): 7
Factorial: 05040
```