**PRACTICAL # 10**

**OBJECT:**

Recursive subroutine in assembly.

**THEORY:**

Recursion, as opposed to iteration, is a way of a function calling itself again based on some condition.

Recursive functions are the functions calling themselves from within their body. There must be some terminal condition that stops further calling of the same function/procedure (called base condition). If there is no such condition, or the condition is written incorrectly, the function may go to infinite execution since it would call it again and again. If in this case, the stack overflows, the program will crash with the stack overflow error.

All recursive algorithms must obey three important laws:

1. A recursive algorithm must have a base case.
2. A recursive algorithm must change its state and move toward the base case.
3. A recursive algorithm must call itself, recursively.

**Program:**

The following is a recursive function that computes the factorial of a number using the recursive method.

.model small

.stack 100d

.data

.code

fact proc

cmp bl, 1 ;terminal condition

je leave

rec:

push bx

dec bl

call fact

pop bx

mul bl

leave:

ret

fact endp

main proc

mov bx, 3

mov ax, 1

push bx

call fact

mov dx, ax

add dx, 48

mov ah, 2

int 21h

mov ah, 4ch

int 21h

ret

main endp

**ACTIVITIES**

**Activity 1**

Write a recursive function that returns the sum of the digits of an integer.

int sumOfDigits(int x);

If x is 234, the function should return 2 + 3 + 4, that is, 9.

Hint: computing the values x/10 and x%10 is useful.

**Activity 2**

Find the sum of the integers from 1 through n using recursion.

**Activity 3**

Write a function to print a multidigit number.

**REVIEW QUESTIONS:**

1. How does recursion work?
2. What is a disadvantage of recursive function?
3. What is the base or terminal condition?

1. What will happen if the recursive function does not change itself and move towards the termainal condition?