ASSEMBLY - RECURSION

http://www.tutorialspoint.com/assembly_programming/assembly_recursion.htm

A recursive procedure is one that calls itself. There are two kind of recursion: direct and indirect. In direct recursion, the procedure calls itself and in indirect recursion, the first procedure calls a second procedure, which in turn calls the first procedure.

Recursion could be observed in numerous mathematical algorithms. For example, consider the case of calculating the factorial of a number. Factorial of a number is given by the equation −

```
Fact (n) = n * fact (n-1) for n > 0
```

For example: factorial of 5 is 1 x 2 x 3 x 4 x 5 = 5 x factorial of 4 and this can be a good example of showing a recursive procedure. Every recursive algorithm must have an ending condition, i.e., the recursive calling of the program should be stopped when a condition is fulfilled. In the case of factorial algorithm, the end condition is reached when n is 0.

The following program shows how factorial n is implemented in assembly language. To keep the program simple, we will calculate factorial 3.

```
section .text
   global _start         ;must be declared for using gcc

_start:                  ;tell linker entry point

   mov bx, 3              ;for calculating factorial 3
   call  proc_fact
   add   ax, 30h
   mov  [fact], ax

   mov   edx,len        ;message length
   mov   ecx,msg        ;message to write
   mov   ebx,1          ;file descriptor (stdout)
   mov   eax,4          ;system call number (sys_write)
   int   0x80           ;call kernel

   mov   edx,1            ;message length
   mov   ecx,fact       ;message to write
   mov   ebx,1          ;file descriptor (stdout)
   mov   eax,4          ;system call number (sys_write)
   int   0x80           ;call kernel

   mov   eax,1          ;system call number (sys_exit)
   int   0x80           ;call kernel

proc_fact:
   cmp   bl, 1
   jg    do_calculation
   mov   ax, 1
   ret

do_calculation:
   dec   bl
   call  proc_fact
   inc   bl
   mul   bl        ;ax = al * bl
   ret

section .data
msg db 'Factorial 3 is:',0xa
len equ $ - msg

section .bss
fact resb 1
```

When the above code is compiled and executed, it produces the following result −

```
Factorial 3 is:
6
```