ENTER and LEAVE Instructions

The ENTER and LEAVE instructions are used to manage stack frames in assembly language.

The **ENTER instruction** creates a stack frame for a called procedure, while the **LEAVE instruction** destroys the stack frame for the current procedure.

The ENTER instruction takes two operands: the number of bytes of stack space to reserve for local variables and the lexical nesting level of the procedure.

The lexical nesting level is the number of nested function calls that have occurred to reach the current function. In most cases, the lexical nesting level is zero.

The ENTER instruction performs the following actions:

Pushes the value of the EBP register onto the stack.

Sets the EBP register to the address of the current stack frame.

Reserves the specified number of bytes of stack space for local variables.

The LEAVE instruction performs the following actions:

Pops the value of the EBP register from the stack.

Restores the ESP register to its value before the ENTER instruction was executed.

The following example shows how to use the ENTER and LEAVE instructions to create and destroy a stack frame for a procedure:

```
416 MySub PROC
417     enter 8, 0 ; Reserve 8 bytes of stack space for local variables.
418     ; ...
419     leave ; Destroy the stack frame.
420     ret
421 MySub ENDP
```

It is important to note that the ENTER and LEAVE instructions should be used together. If you use the ENTER instruction to create a stack frame, you must also use the LEAVE instruction to destroy the stack frame. Otherwise, the stack space that you reserved for local variables will not be released.

The image that you provided shows the stack before and after the ENTER instruction has executed. The ENTER instruction has pushed the value of the EBP register onto the stack and set the EBP register to the address of the current stack frame. The ENTER instruction has also reserved 8 bytes of stack space for local variables.

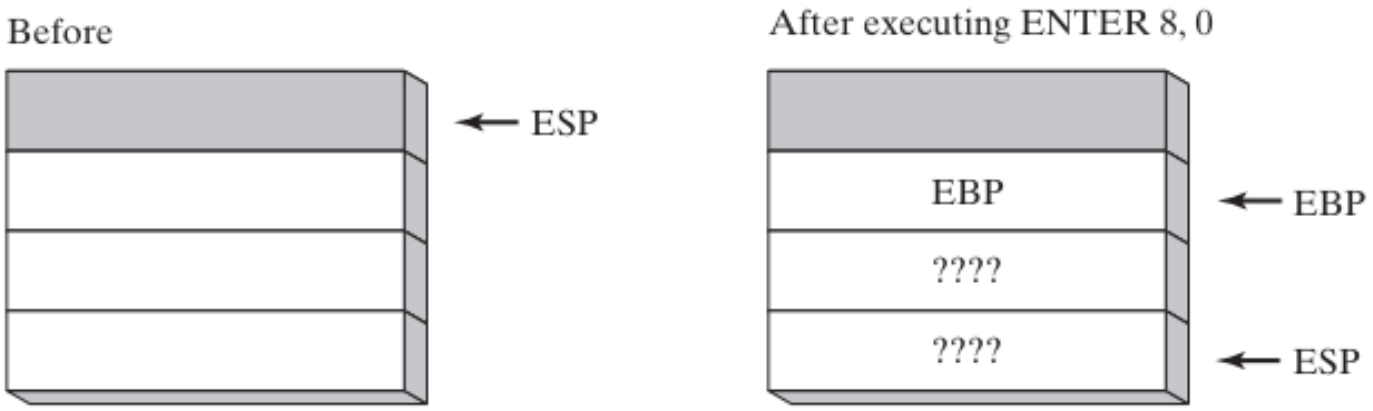
Why is it important to use the ENTER and LEAVE instructions together?

The ENTER and LEAVE instructions should be used together because they work together to manage the stack.

The ENTER instruction creates a stack frame for a called procedure, while the LEAVE instruction destroys the stack frame for the current procedure.

If you use the ENTER instruction to create a stack frame but do not use the LEAVE instruction to destroy the stack frame, **the stack space that you reserved for local variables will not be released.**

This will cause the stack to grow larger and larger, and it could eventually cause the program to crash.



The image above shows the stack before and after the ENTER instruction has executed. The ENTER instruction has pushed the value of the EBP register onto the stack and set the EBP register to the address of the current stack frame. The ENTER instruction has also reserved 8 bytes of stack space for local variables.