Lecture # 25

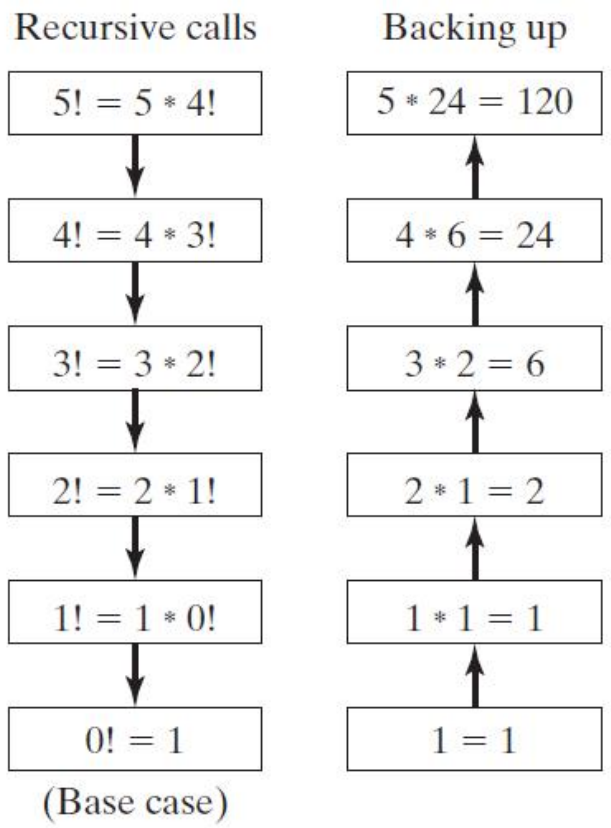
Factorial Example

```
; Sum of Integers                (RecursiveSum.asm)

INCLUDE Irvine32.inc
.code
main PROC
     mov  ecx,5                  ; count = 5
     mov  eax,0                  ; holds the sum
     call CalcSum                ; calculate sum
L1:  call WriteDec               ; display EAX
     call Crlf                   ; new line
     exit
main ENDP
;-----------------------------------------------------
CalcSum PROC
; Calculates the sum of a list of integers
; Receives: ECX = count
; Returns: EAX = sum
;-----------------------------------------------------
    cmp  ecx,0                   ; check counter value
    jz   L2                      ; quit if zero
    add  eax,ecx                 ; otherwise, add to sum
    dec  ecx                     ; decrement counter
    call CalcSum                 ; recursive call
L2: ret
CalcSum ENDP
end Main
```

Table 8-1 Stack Frame and Registers (CalcSum).

Pushed on Stack	Value in ECX	Value in EAX
L1	5	0
L2	4	5
L2	3	9
L2	2	12
L2	1	14
L2	0	15

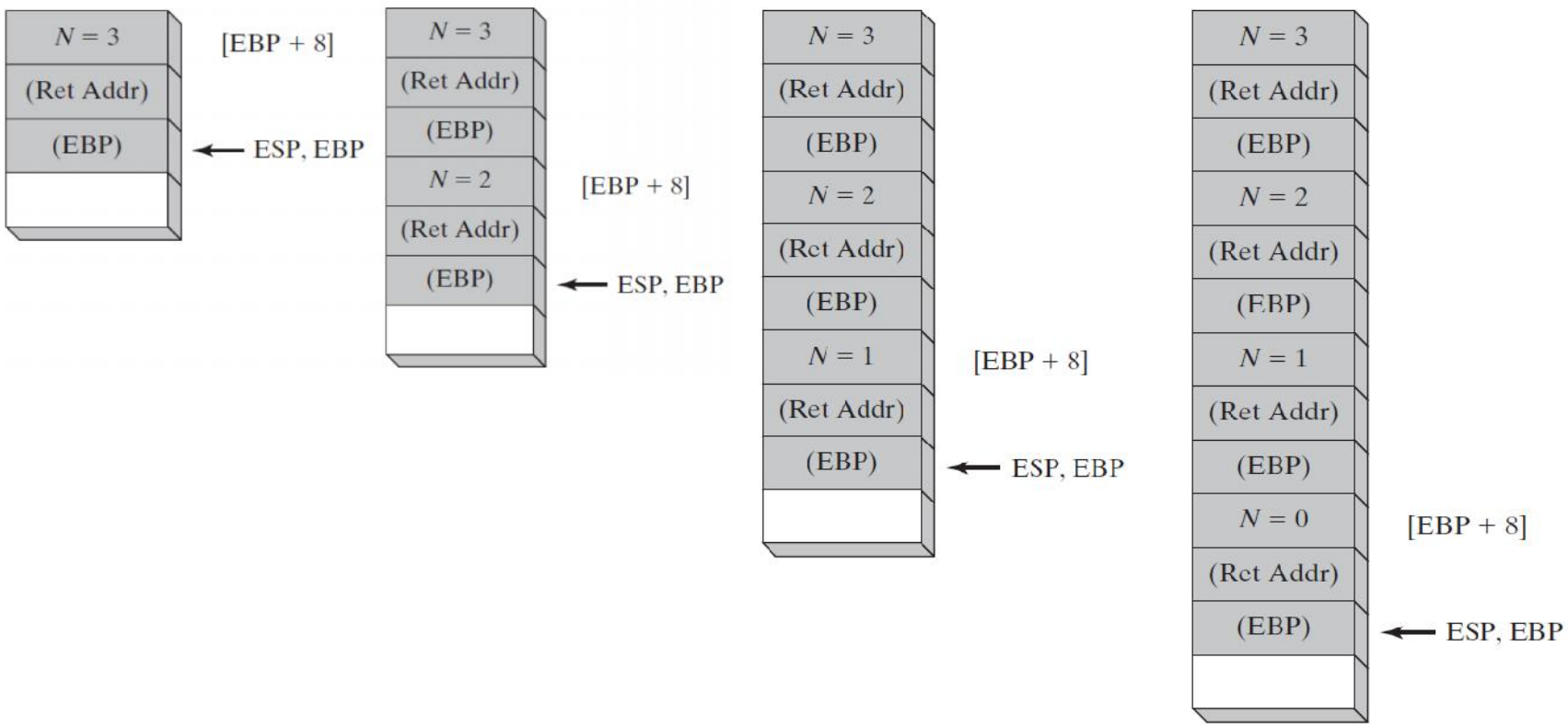
```
int function factorial(int n)
{
      if(n == 0)
        return 1;
      else
        return n * factorial(n−1);
}
```

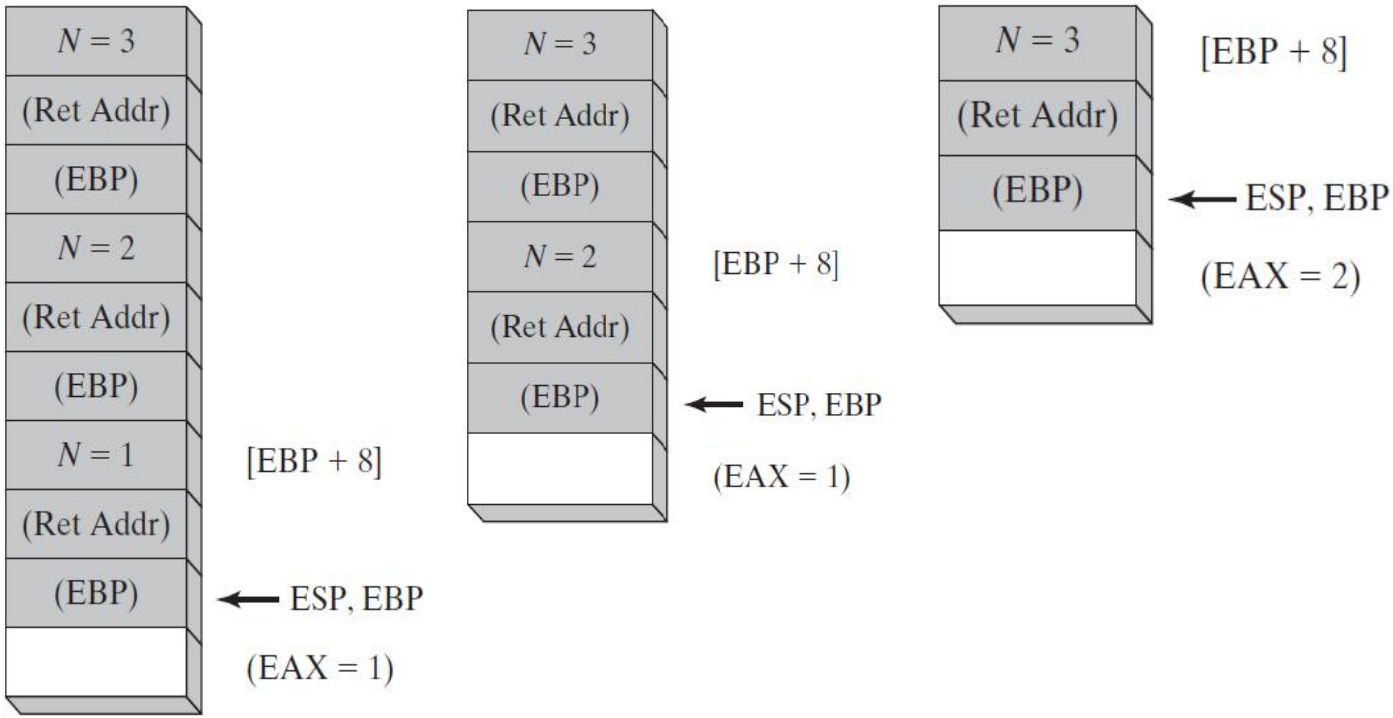


```
; Calculating a Factorial (Fact.asm)

INCLUDE Irvine32.inc
.code
main PROC
     push 5                    ; calc 5!
     call Factorial            ; calculate factorial (EAX)
     call WriteDec             ; display it
     call Crlf
     exit
main ENDP
```

```
;-----------------------------------------------------
Factorial PROC
; Calculates a factorial.
; Receives: [ebp+8] = n, the number to calculate
; Returns: eax = the factorial of n
;-----------------------------------------------------
    push  ebp
    mov   ebp,esp
    mov   eax,[ebp+8]          ; get n
    cmp   eax,0                ; n > 0?
    ja    L1                   ; yes: continue
    mov   eax,1                ; no: return 1 as the value of 0!
    jmp   L2                   ; and return to the caller
L1: dec   eax
    push  eax                  ; Factorial(n-1)
    call  Factorial

; Instructions from this point on execute when each
; recursive call returns.

ReturnFact:
    mov   ebx,[ebp+8]          ; get n
    mul   ebx                  ; EDX:EAX = EAX * EBX

L2: pop   ebp                  ; return EAX
    ret   4                    ; clean up stack
Factorial ENDP
END main
```

```
push 3
call Factorial                    ; EAX = 3!
```




N = 3
(Ret Addr)
(EBP)
N = 2
(Ret Addr)
(EBP)
N = 1
(Ret Addr)
(EBP)
[EBP + 8]
ESP, EBP
(EAX = 1)
N = 3
(Ret Addr)
(EBP)
N = 2
(Ret Addr)
(EBP)
[EBP + 8]
ESP, EBP
(EAX = 1)
N = 3
(Ret Addr)
(EBP)
[EBP + 8]
ESP, EBP
(EAX = 2)