Parameter Passing & Recursion

Chapter 17
of our main text book
i.e.
Assembly language
programming and organization
of **IBM** PCs
By
Ytha **Yu** and Charles **Marut**

Overview

1 Parameter Passing

2 Recursion

Parameter passing to procedures

3

Methods

1. By Registers

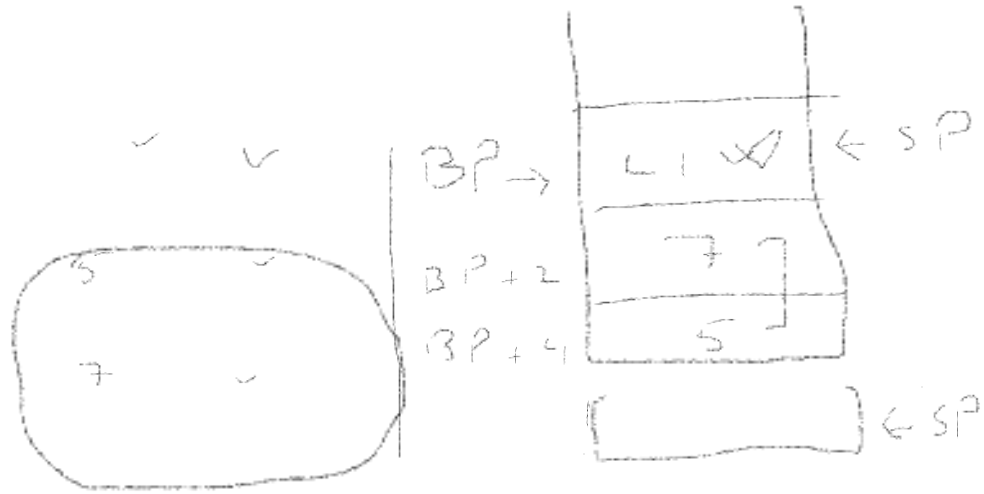
2. By Reference(Address)

3. By Stack

By Register

By **Address/Variable**

By **Stack**



By **Stack**

```
org 100h
.DATA
   WORD1 DW 2
   WORD2 DW 5
.CODE
MAIN PROC
   PUSH WORD1
   PUSH WORD2
   CALL ADD_WORDS
   RET
MAIN ENDP
```

```
ADD_WORDS PROC
 PUSH BP
 MOV BP,SP
 MOV AX,[BP+6]   ;AX GETS
WORD1
 ADD AX,[BP+4]   ;AX HAS SUM
 POP BP
 RET 4------> ?????
ADD_WORDS ENDP
```

Recursion

Recursion

- A process is said to be recursive if it is defined in terms of itself

- A recursive procedure is a procedure that calls itself.

Iteration

- keep repeating until a task is done
- *e.g.,* loop counter reaches limit

Recursion

- Solve a large problem by breaking it up into smaller and smaller pieces until you can solve it; combine the results.

Base **Properties**

- A recursive function must have **two** properties:
  - There must be a certain (base) criteria for which function doesn't call itself.
  - Each time function does call itself (directly or indirectly), it must get closer to the base criteria

Factorial example

- Non-Recursive

$$Factorial(N) = N * (N-1) * (N-2) * \ldots * 2 * 1$$

- Recursive

$$Factorial(N) = N * Factorial(N-1)$$

$$Factorial(1) = 1$$

Pseudo code

- Recursive

*Factorial(N) = N * Factorial(N-1)*

Factorial(1) = 1

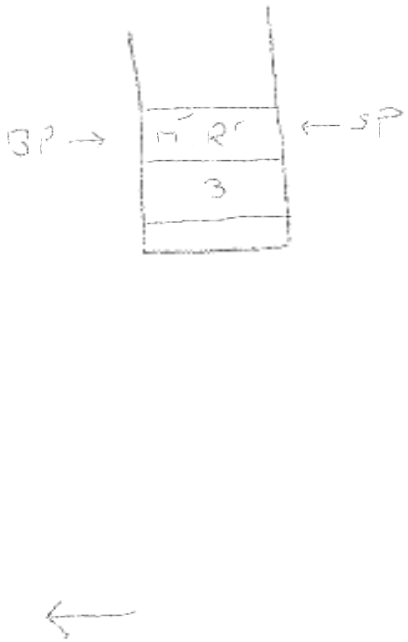
Pseudo code

```
FACTORIAL(N)
IF N = 1
    RESULT=1
    ret
ELSE
     CALL FACTORIAL(N-1)

     RESULT = N * RESULT
END_IF
RETURN
```

Code

```
FACTORIAL(N)
IF N = 1
    RESULT=1
   ret
ELSE
    CALL FACTORIAL(N-1)

        RESULT = N * RESULT
END_IF
RETURN
```

Procedure




```
FACTORIAL(N)
IF N = 1
    RESULT=1
   ret
ELSE
    CALL FACTORIAL(N-1)

        RESULT = N * RESULT
END_IF
RETURN
```

Procedure

```
FACTORIAL(N)
IF N = 1
    RESULT=1
   ret
ELSE
    CALL FACTORIAL(N-1)

        RESULT = N * RESULT
END_IF
RETURN
```

Procedure

```
FACTORIAL(N)
IF N = 1
    RESULT=1
   ret
ELSE
    CALL FACTORIAL(N-1)
        RESULT = N * RESULT
END_IF
RETURN
```

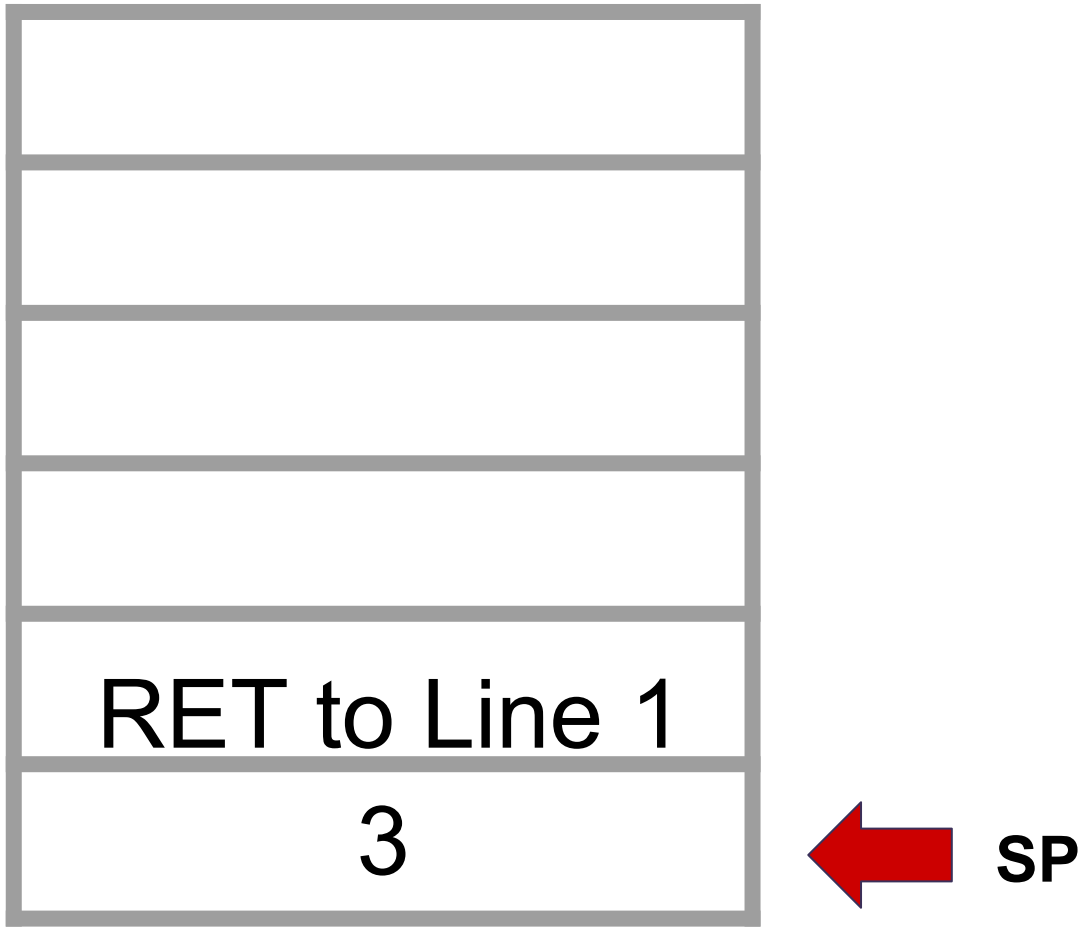
Code

```
org 100h

; add your code here
.data
N dw 0
;n1 dw 0
result dw 0
.code

main proc
mov ax,3
push ax
call factorial

Ret □LINE 1
main endp


factorial proc

mov bp,sp

mov ax,[bp+2]
mov N,ax

cmp N,1
jne again
```

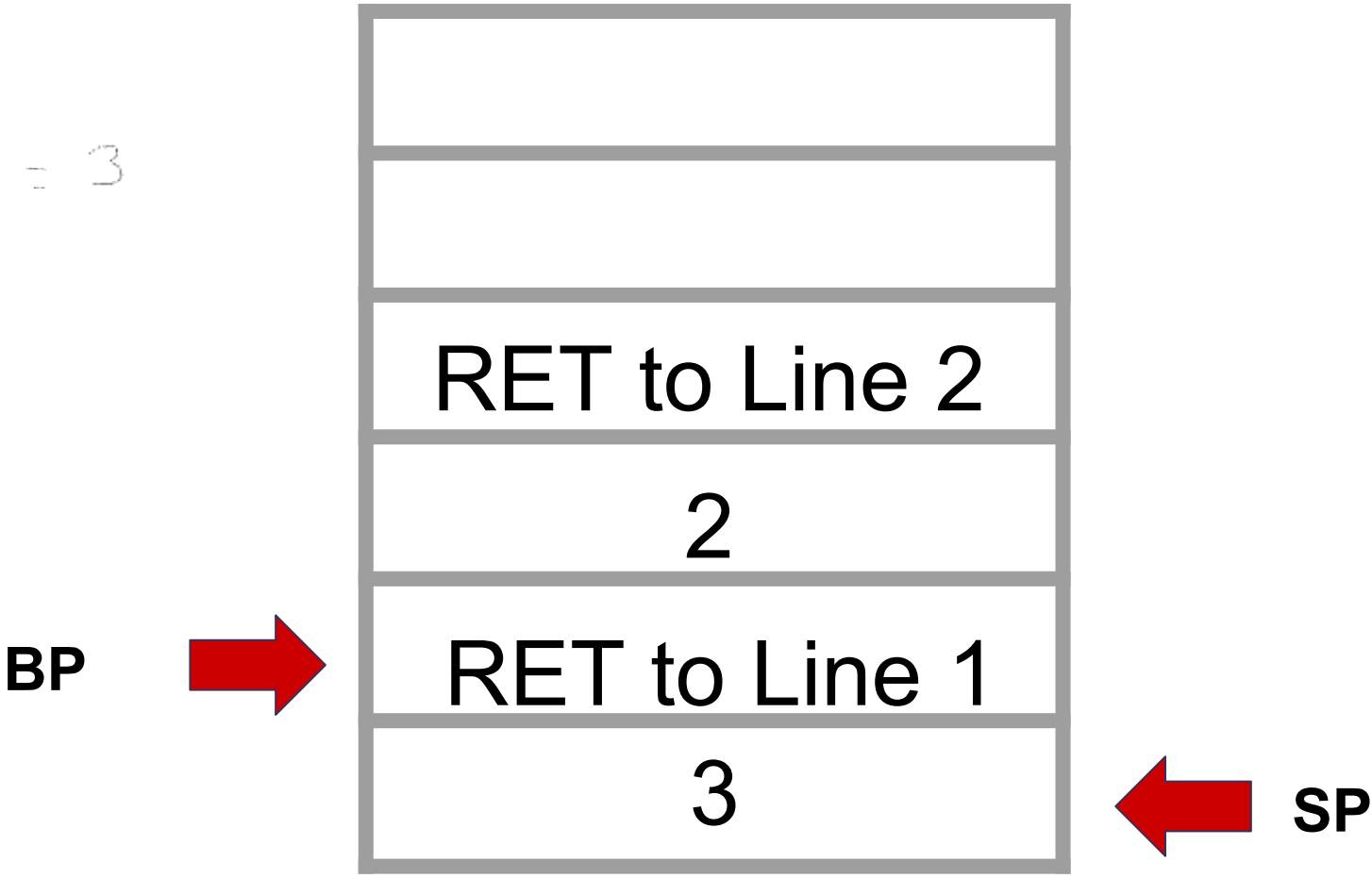
```
FACTORIAL(N)
IF N = 1
    RESULT=1
   ret
ELSE
    CALL FACTORIAL(N-1)


        RESULT = N * RESULT
END_IF
RETURN
```

Working Of Code

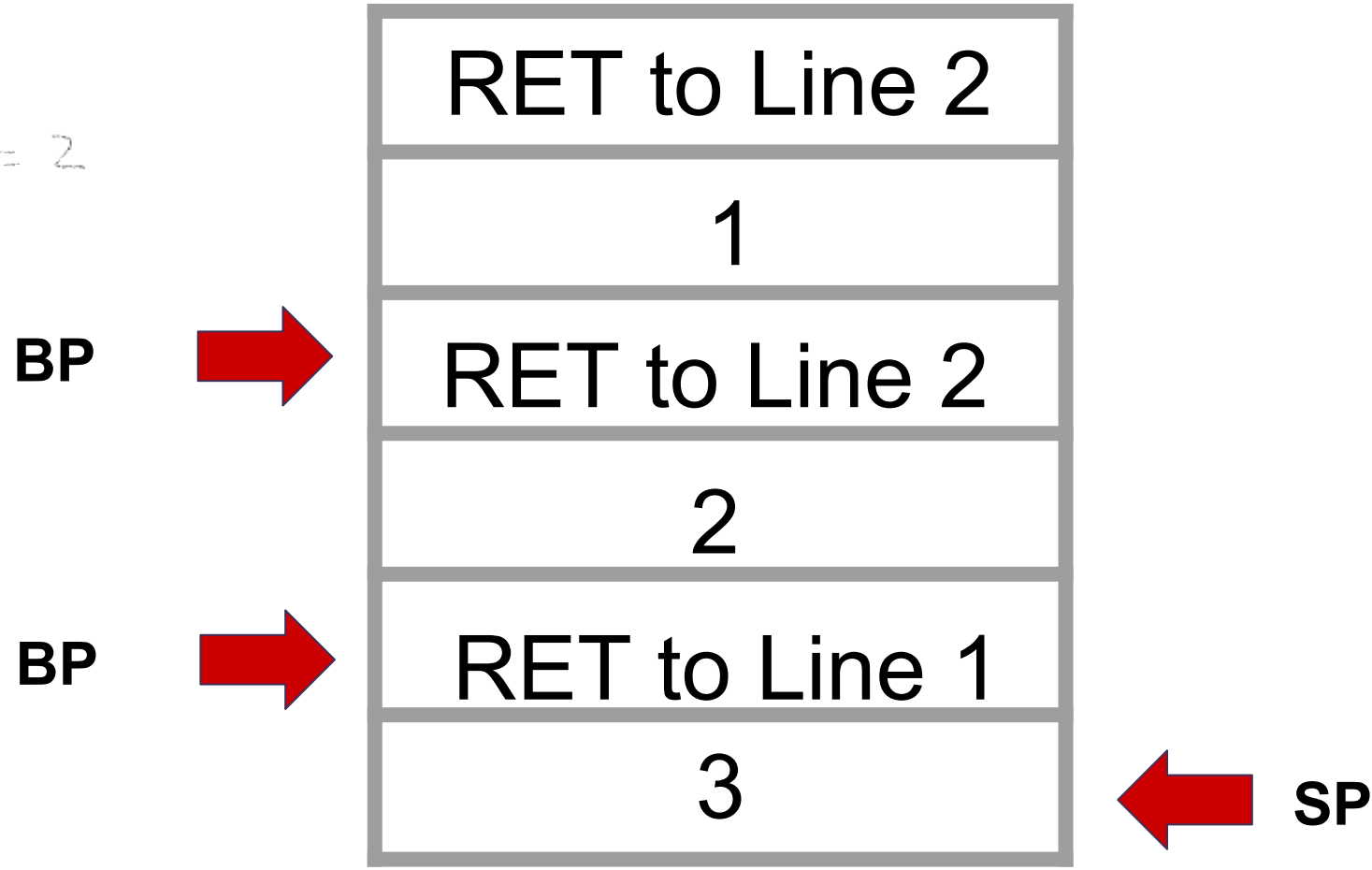
```
main proc
mov ax,3
push ax
call factorial

Ret <--LINE 1
main endp
```



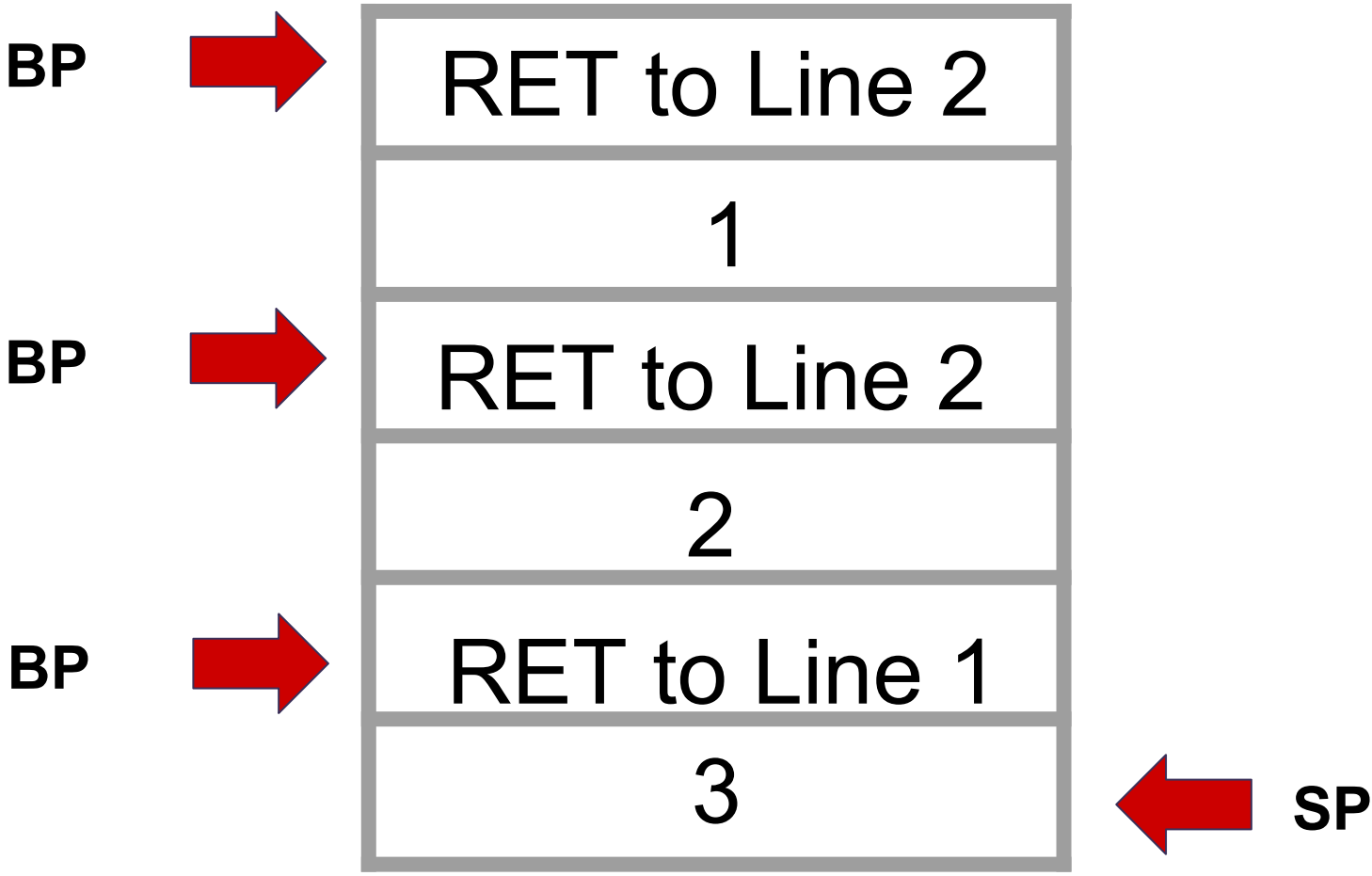
```
factorial proc

mov bp,sp

mov ax,[bp+2]
mov N,ax        — N = 3

cmp N,1
jne again
;---------
;n==1
mov result,1
jmp exit
;----------

again:
dec N           — N = 2
push N
call factorial

;get the parameter
again of this call
from stack
mov bp,sp <--LINE 2
```



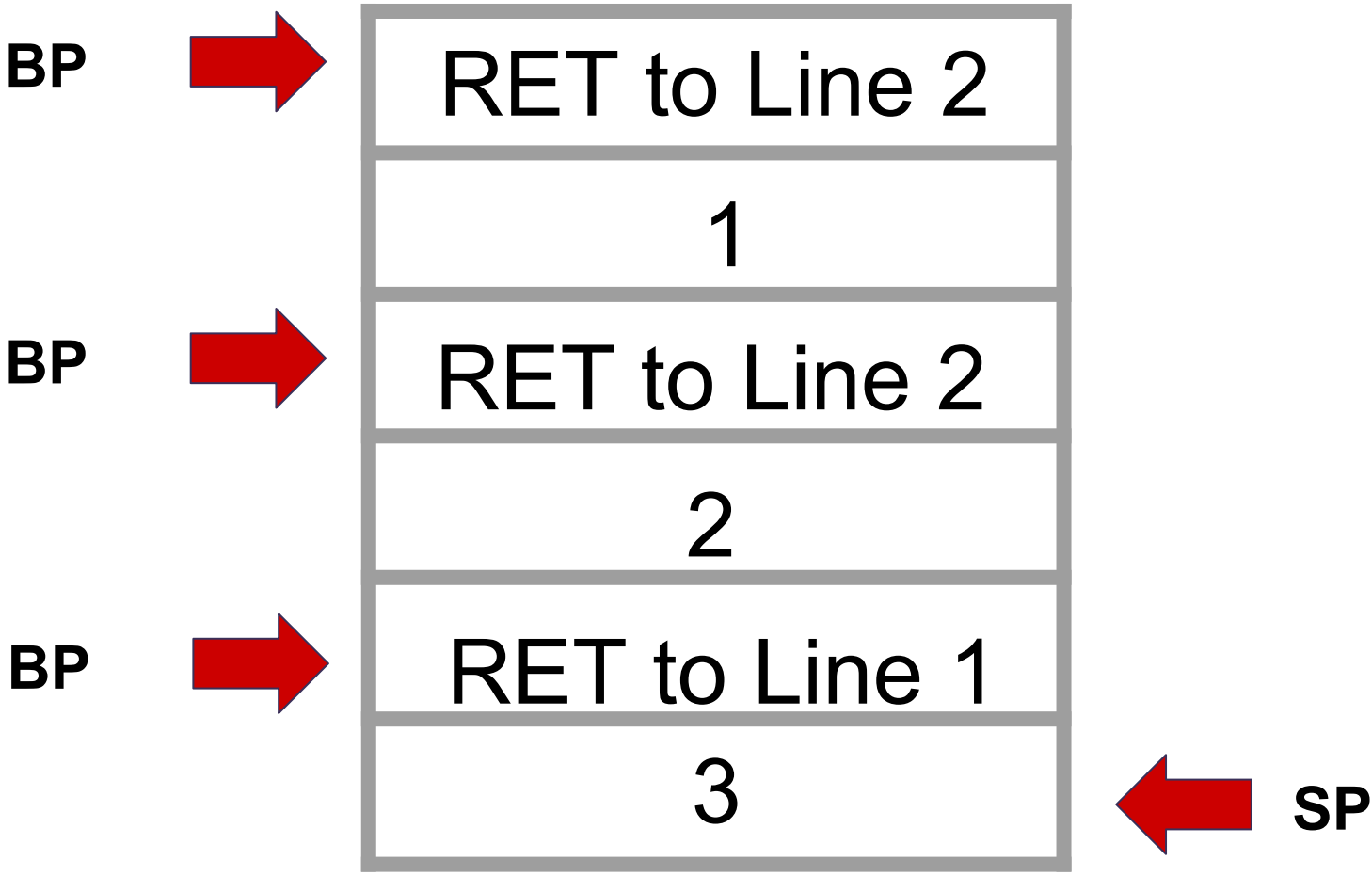
```
factorial proc

mov bp,sp

mov ax,[bp+2]
mov N,ax        ——— N = 2

cmp N,1
jne again
;---------
;n==1
mov result,1
jmp exit
;----------

again:
dec N           — N = 1
push N
call factorial

;get the parameter
again of this call
from stack
mov bp,sp <--LINE 2
```



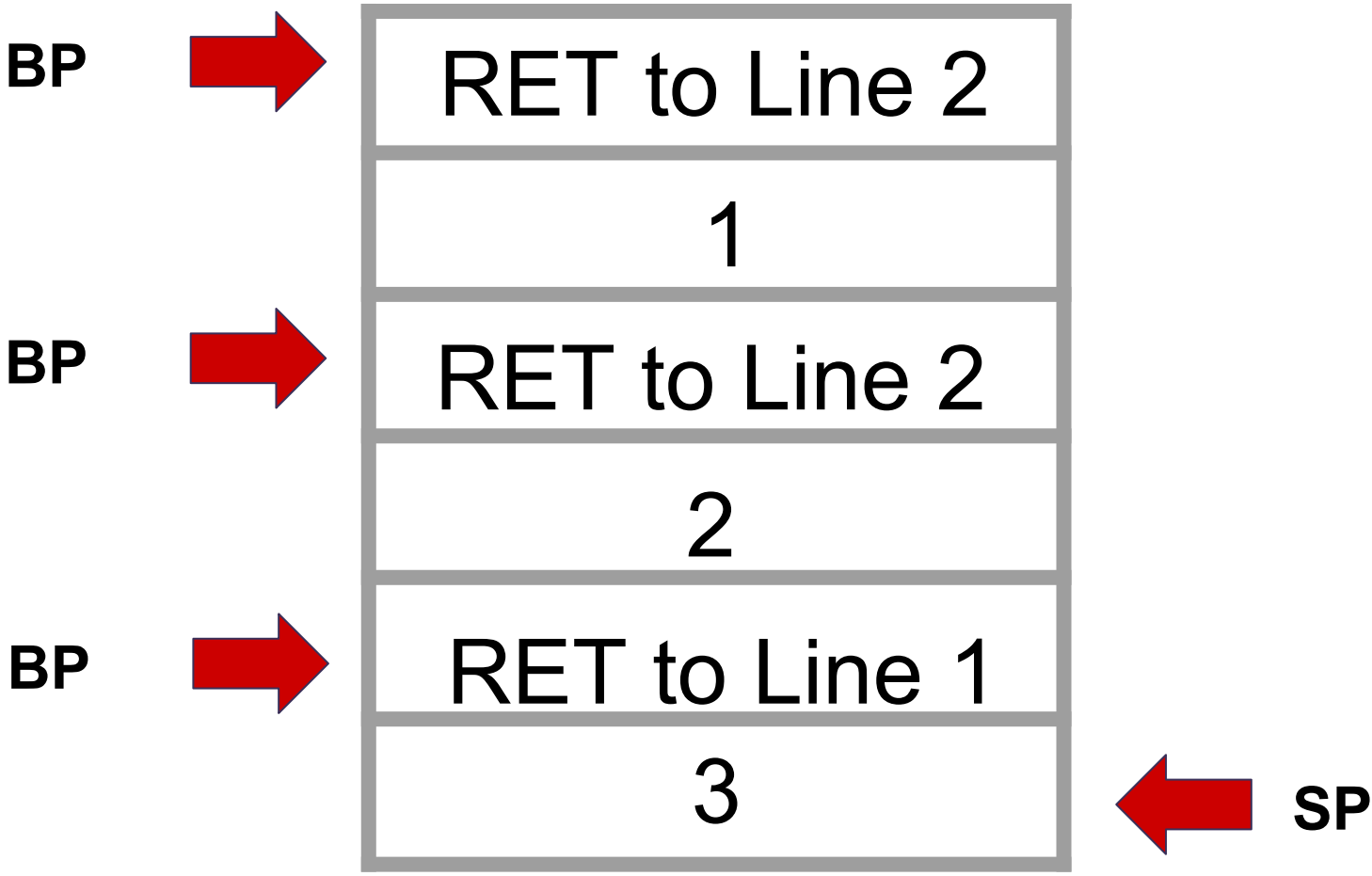
```
factorial proc

mov bp,sp

mov ax,[bp+2]
mov N,ax

cmp N,1
jne again
;---------
;n==1
mov result,1
jmp exit
;----------
again:
dec N
push N
call factorial

;get the parameter
again of this call
from stack
mov bp,sp <--LINE 2
…
exit:
RET 2
```



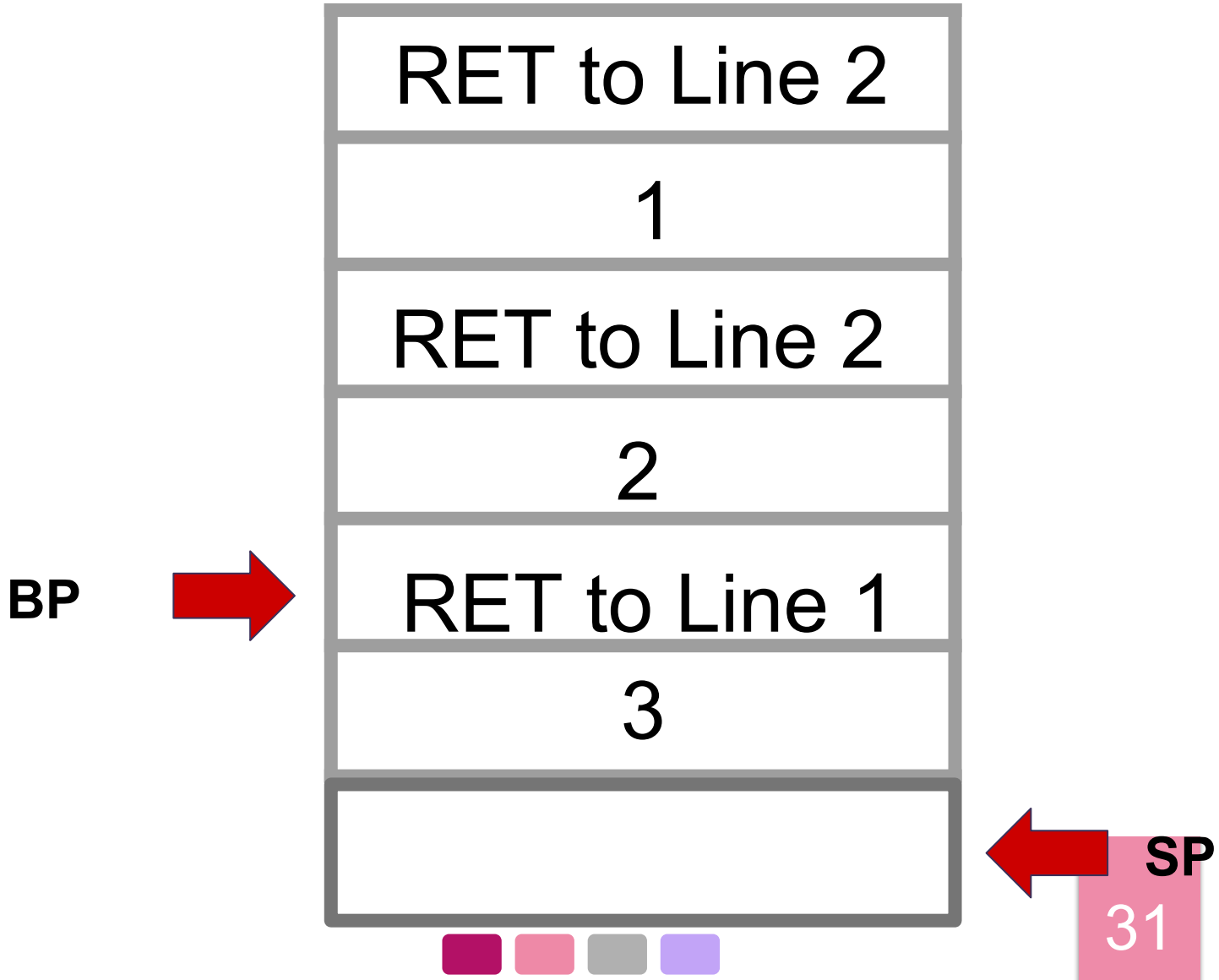
```
call factorial

;get the parameter
again of this call
from stack
mov bp,sp <--LINE 2
mov ax,[bp+2]
;-----------

mul result
mov result,ax

exit:
ret 2

factorial endp
```

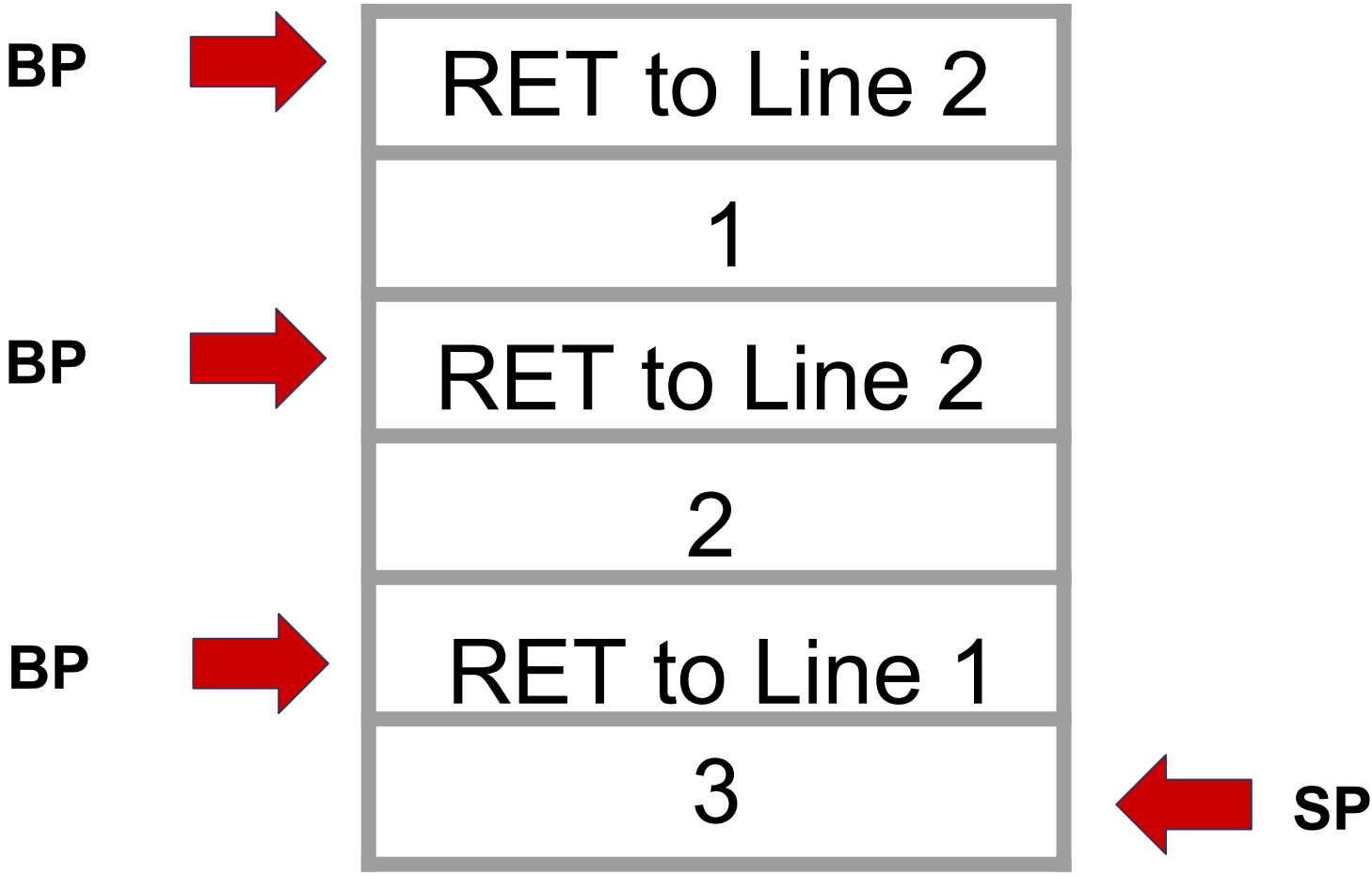


```
call factorial

;get the parameter
again of this call
from stack
mov bp,sp <--LINE 2
mov ax,[bp+2]
;-----------

mul result
mov result,ax

exit:
ret 2

factorial endp
```



```
main proc
mov ax,3
push ax
call factorial

Ret <--LINE 1
main endp
```




BP
RET to Line 2
1
BP
RET to Line 2
2
BP
RET to Line 1
3
SP

Code from Book

```
org 100h
.CODE
MAIN PROC
1:   MOV AX,3
2:  PUSH AX
3:   CALL FACTORIAL
4:  RET
MAIN ENDP
FACTORIAL PROC
5:   PUSH BP
6:   MOV BP, SP
7:  CMP WORD PTR[BP+4], 1
8:   JG END_IF
6: MOV AX, 1
7:    JMP RETURN
END_IF:
8:    MOV BX, [BP+4]
9:    DEC BX
10:  PUSH BX
11:    CALL FACTORIAL
12:    MUL WORD PTR[BP+4]
RETURN:
13:    POP BP
14:    RET 2
FACTORIAL ENDP
```

Code Pattern

Fortunately

All Recursion programs
Follow
Same
pattern of steps

Let's Analyze

Pattern of **Steps** :1

1. Pass parameters from main
2. Call procedure
3. Load parameters in procedure

Pattern of **Steps** :2

4 Check Base Criteria

5 If fulfilled then RET

6 Else prepare parameters for next call

Pattern of **Steps** :3

7 Call again

8 Load previous call parameters

9 Process and RET

Mark

Steps on

Factorial

example program

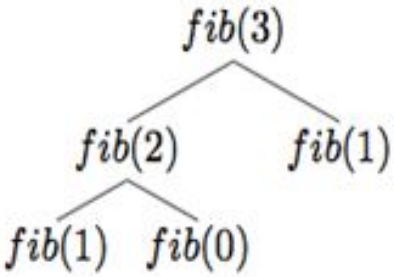
Fibonacci Sequence

- Following is the Fibonacci Series
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89……..
- Fibonacci series start with 1 and 1 as first two numbers and any **subsequent number is sum of previous two numbers**
- 2 = 1+1
- 3 = 2+1
- 5 = 3+2
- 8 = 5+3
- 21 = 13+8

Nth Fibonacci Number

- Base Case **: if n = 1 or n=0 then $F_n = 1$**
- Recursive Call**: if n > 1 then $F_n = F_{n-2} + F_{n-1}$**

```
int fib(int  n)
{
  if ( n == 0 )
    return 1;
  else if ( n == 1 )
    return 1;
  else
    return (fib(n-1) + fib (n-2) );
}
```




The End