Passing Parameters via the Stack

- You can pass a parameter to a procedure/function
  - ***by value***
  - *by reference*

- When passed ***by value***, the procedure receives a **copy** of the parameter.
  - If the parameter is modified by the procedure, the new value does not affect the value of the parameter elsewhere in the program.
  - Passing a parameter by value causes the *parameter to be cloned* and the *cloned version of the parameter* to be used by the procedure.

- When passed *by reference*, the procedure receives a **pointer**, (i.e., an **address**) to the parameter.
  - *There is only one copy of the parameter* and the procedure can access this value because it knows the address of the parameter.
  - If the procedure modifies the parameter, it is modified the original value.

Passing Parameters via the Stack

- Let's examine how parameters are passed to a function when we compile `swap(int a, int b)` that is *intended* to exchange two values.

```
void swap(int a, int b)  /* swaps the value of a and b */
{  int temp;
   temp = a;             /* copy a      to temp     */
   a    = b              /* copy b      to a,   and */
   b    = temp;          /* copy temp to b          */
}

void main(void)
{  int x = 2, y = 3;
   swap (x, y);          /* swap a and b */
}
```

Passing Parameters via the Stack

```
AREA SwapVal, CODE, READONLY

ENTRY
MOV  sp,#0x1000              ;set up stack pointer
MOV  fp,#0xFFFFFFFF          ;set up dummy fp for tracing
B    main                    ;jump to the function main

;  void swap (int a, int b)
;  Parameter a    is at [fp]+4
;  Parameter b    is at [fp]+8
;  Variable  temp is at [fp]-4
```

Passing Parameters via the Stack

```
;     {
swap  SUB   sp,sp,#4     ;Create stack frame: decrement sp
      STR   fp,[sp]      ;push the frame pointer on the stack
      MOV   fp,sp        ;frame pointer points at the base
;     int temp;
      SUB   sp,sp,#4     ;move sp up 4 bytes for temp
;     temp = a;
      LDR   r0,[fp,#4]   ;get parameter a from the stack
      STR   r0,[fp,#-4]  ;copy a to temp on the stack frame
;     a    = b;
      LDR   r0,[fp,#8]   ;get parameter b from the stack
      STR   r0,[fp,#4]   ;copy b to a
;     b    = temp;
      LDR   r0,[fp,#-4]  ;get temp from the stack frame
      STR   r0,[fp,#8]   ;copy temp to b
;     }
                         ;Collapse stack frame created for swap
      MOV   sp,fp        ;restore the stack pointer
      LDR   fp,[sp]      ;restore old frame pointer from stack
      ADD   sp,sp,#4     ;move stack pointer down 4 bytes
      MOV   pc,lr        ;return by loading LR into PC
```

Passing Parameters via the Stack

```
;     void main(void)
;     {
main                        ;Create stack frame in main for x, y
      SUB   sp,sp,#4        ;move the stack pointer up
      STR   fp,[sp]         ;push the frame pointer on the stack
      MOV   fp,sp           ;the frame pointer points at the base;
      int x = 2, y = 3;
SUB   sp,sp,#8        ;move sp up 8 bytes for 2 integers
      MOV   r0,#2             ;x = 2
      STR   r0,[fp,#-4]       ;put x in stack frame
      MOV   r0,#3             ;y = 3
      STR   r0,[fp,#-8]       ;put y in stack frame
;     swap(x, y);
      LDR   r0,[fp,#-8]       ;get y from stack frame
      STR   r0,[sp,#-4]!      ;push y on stack
      LDR   r0,[fp,#-4]       ;get x from stack frame
      STR   r0,[sp,#-4]!      ;push x on stack
      BL    swap              ;call swap, save return address in LR
      ADD   sp,sp,#8          ;Clean the stack from the parameters
;     }
MOV   sp,fp           ;restore the stack pointer
      LDR   fp,[sp]           ;restore old frame pointer from stack
      ADD   sp,sp,#4          ;move stack pointer down 4 bytes
      SWI   0x11              ;call O/S to terminate the program
      END
```

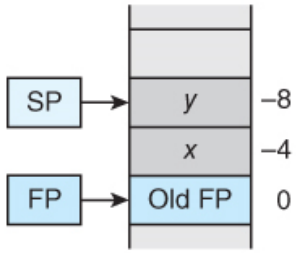
Bold is not correct in page 244

Passing Parameters via the Stack

- This code swaps the variables in the stack frame,
- When a return is made the stack frame is collapsed and the effect of the swap is lost.
- The variables in the calling environment are not affected.

Passing Parameters via the Stack

FIGURE 4.8 Passing values to a subroutine by value



(a) State of the stack in main after creating stack frame with:

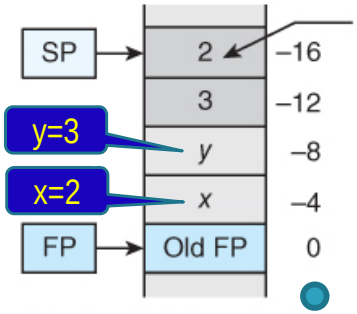
```
SUB sp,sp,#4
STR fp,[sp]
MOV fp,sp
SUB sp,sp,#8
```

The location of the addresses relative to the FP needs to be shifted down one cell. It is not correct in page 245

FP not SP. Not correct in page 245

Same thing

The values of the parameters are not correct in page 245



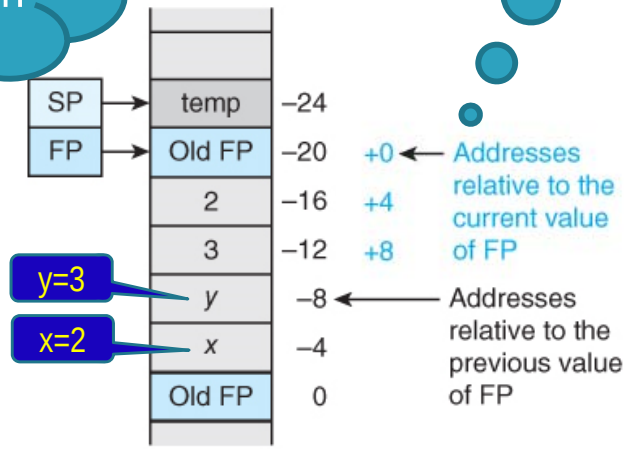
The values of parameters *x* and *y* are pushed o[illegible] stack

(b) The stack in main after putting two parameters in the stack frame with:

```
MOV r0,#2
STR r0,[fp,#-4]
MOV r0,#3
STR r0,[fp,#-8]
```

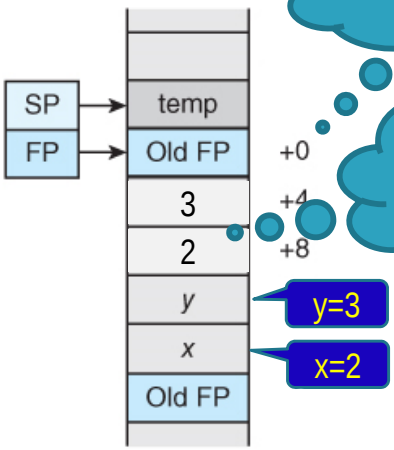
Then pushing two parameters on the stack

```
LDR r0,[fp,#-8]
STR r0,[sp,#-4]!
LDR r0,[fp,#-4]
STR r0,[sp,#-4]!
```



(c) The stack after the creation of a stack frame in `swap`. The new stack frame is four bytes deep and holds the variable temp. The frame is created by:

```
SUB sp,sp,#4
STR fp,[sp]
MOV fp,sp
SUB sp,sp,#4
```



(d) The stack after executing the body of swap. Note that all data is referenced to FP.

```
LDR r0,[fp,#4]
STR r0,[fp,#-4]
LDR r0,[fp,#8]
STR r0,[fp,#4]
LDR r0,[fp,#-4]
STR r0,[fp,#8]
```

Passing Parameters via the Stack

- In the next example, we pass parameters by reference

```
void swap(int *a, int *b)  /* A function to swap two parameters
                              in calling program      */
{ int temp;                /* copy *a     to temp     */
  temp = *a;               /* copy *b     to *a,  and */
  *a   = *b;               /* copy temp to  *b        */
  *b   = temp;
}

void main(void)
{ int x = 2, y = 3;
  swap(&x, &y);            /* call swap and pass
                              addresses of parameters  */
}
```

Passing Parameters via the Stack

```
AREA SwapVal, CODE, READONLY

ENTRY
MOV  sp,#0x1000          ;set up stack pointer
MOV  fp,#0xFFFFFFFF      ;set up dummy fp for tracing
B    main                ;jump to main function

;  void swap (int *a, int *b)
;  Parameter *a   is at [fp]+4
;  Parameter *b   is at [fp]+8
;  Variable  temp is at [fp]-4
```

Passing Parameters via the Stack

```
;       {
swap    SUB   sp,sp,#4      ;Create stack frame: decrement sp
        STR   fp,[sp]       ;push the frame pointer on the stack
        MOV   fp,sp         ;frame pointer points at the base
;       int temp;
        SUB   sp,sp,#4      ;move sp up 4 bytes for temp
;       temp = *a;
        LDR   r1,[fp,#4]    ;get address of parameter a
        LDR   r2,[r1]       ;get value of parameter a (i.e., *a)
        STR   r2,[fp,#-4]   ;store *a in temp in stack frame
;       *a = *b;
        LDR   r0,[fp,#8]    ;get address of parameter b
        LDR   r3,[r0]       ;get value of parameter b (i.e., *b)
        STR   r3,[r1]       ;store *b in *a
;       *b = temp;
        LDR   r3,[fp,#-4]    ;get temp
        STR   r3,[r0]        ;store temp in *b
;       }
                            ;Collapse stack frame created for swap
        MOV   sp,fp         ;restore the stack pointer
        LDR   fp,[sp]       ;restore old frame pointer from stack
        ADD   sp,sp,#4      ;move stack pointer down 4 bytes
        MOV   pc,lr         ;return by loading LR into PC
```

Missing the * in page 247

Passing Parameters via the Stack

```
;     void main(void)
;     {
main                        ;Create stack frame in main for x, y
      SUB   sp,sp,#4        ;move the stack pointer up
      STR   fp,[sp]         ;push the frame pointer on the stack
      MOV   fp,sp           ;the frame pointer points at the base;
      int x = 2, y = 3;
      SUB   sp,sp,#8        ;move sp up 8 bytes for 2 integers
      MOV   r0,#2           ;x = 2
      STR   r0,[fp,#-4]     ;put x in stack frame
      MOV   r0,#3           ;y = 3
      STR   r0,[fp,#-8]     ;put y in stack frame
;     swap(&x, &y);
      SUB   r0,fp,#8        ;get  address of y in stack frame
      STR   r0,[sp,#-4]!    ;push address of y on stack
      SUB   r0,fp,#4        ;get  address of x in stack frame
      STR   r0,[sp,#-4]!    ;push address of x on stack
      BL    swap            ;call swap, save return address in LR
      ADD   sp,sp,#8        ;Clean the stack from the parameters
;     }
      MOV   sp,fp           ;restore the stack pointer
      LDR   fp,[sp]         ;restore old frame pointer from stack
      ADD   sp,sp,#4        ;move stack pointer down 4 bytes
      SWI   0x11            ;call O/S to terminate the program
      END
```

Bold is not correct in page 244

Passing Parameters via the Stack

- In the function main, the addresses of the *parameters are pushed on the stack* by means of the following instructions:

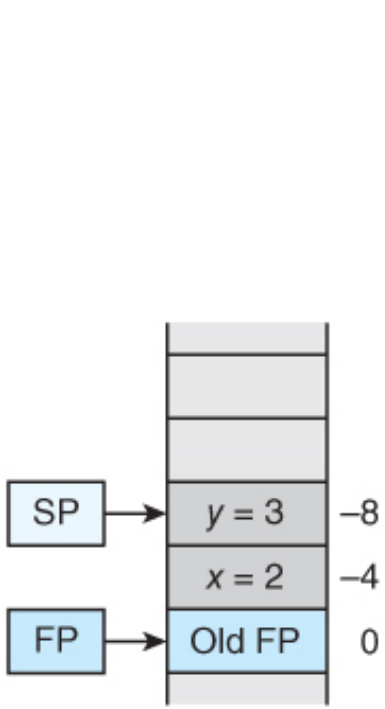
```
SUB  r0,fp,#8      ;get  address of y in stack frame
STR  r0,[sp,#-4]!  ;push address of y on stack
SUB  r0,fp,#4      ;get  address of x in stack frame
STR  r0,[sp,#-4]!  ;push address of x on stack
```

- In the function swap, the addresses of *parameters are read from the stack* by means of

```
;    temp = *a;
     LDR  r1,[fp,#4]  ;get address of parameter a
     LDR  r2,[r1]     ;get value of parameter a (i.e., *a)
     STR  r2,[fp,#-4] ;store *a in temp in stack frame
;    *a = *b;
     LDR  r0,[fp,#8]  ;get address of parameter b
     LDR  r3,[r0]     ;get value of parameter b (i.e., *b)
     STR  r3,[r1]     ;store *b in *a
;    *b = temp;
     LDR  r3,[fp,#-4]  ;get temp
     STR  r3,[r0]      ;store temp in *b
```

Passing Parameters via the Stack

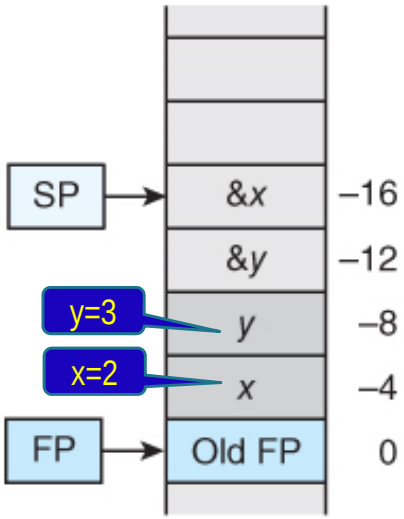
FIGURE 4.9 Passing values to a subroutine by reference



(a) State of the stack after

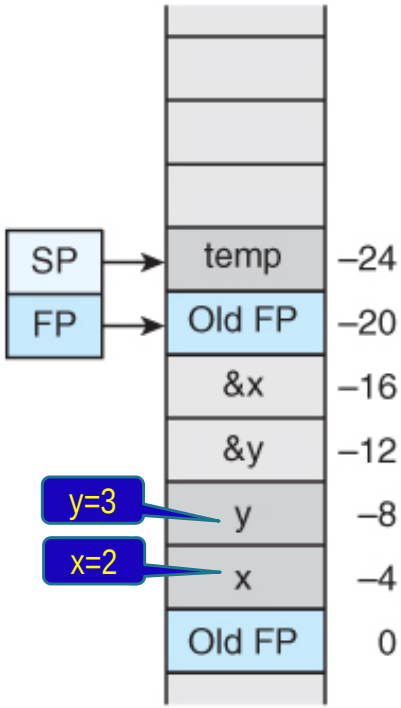
```
SUB sp,sp,#4
STR fp,[sp]
MOV fp,sp
SUB sp,sp,#8
MOV r0,#2
STR r0,[fp,#-4]
MOV r0,#3
STR r0,[fp,#-8]
```

in function main



(b) State of the stack after pushing parameter addresses by

```
SUB r0,fp,#8
STR r0,[sp,#-4]!
SUB r0,fp,#4
STR r0,[sp,#-4]!
```



(c) State of the stack after subroutine call and stack frame created by

```
SUB sp,sp,#4
STR fp,[sp]
MOV fp,sp
SUB sp,sp,#4
```

Addresses with respect to new FP

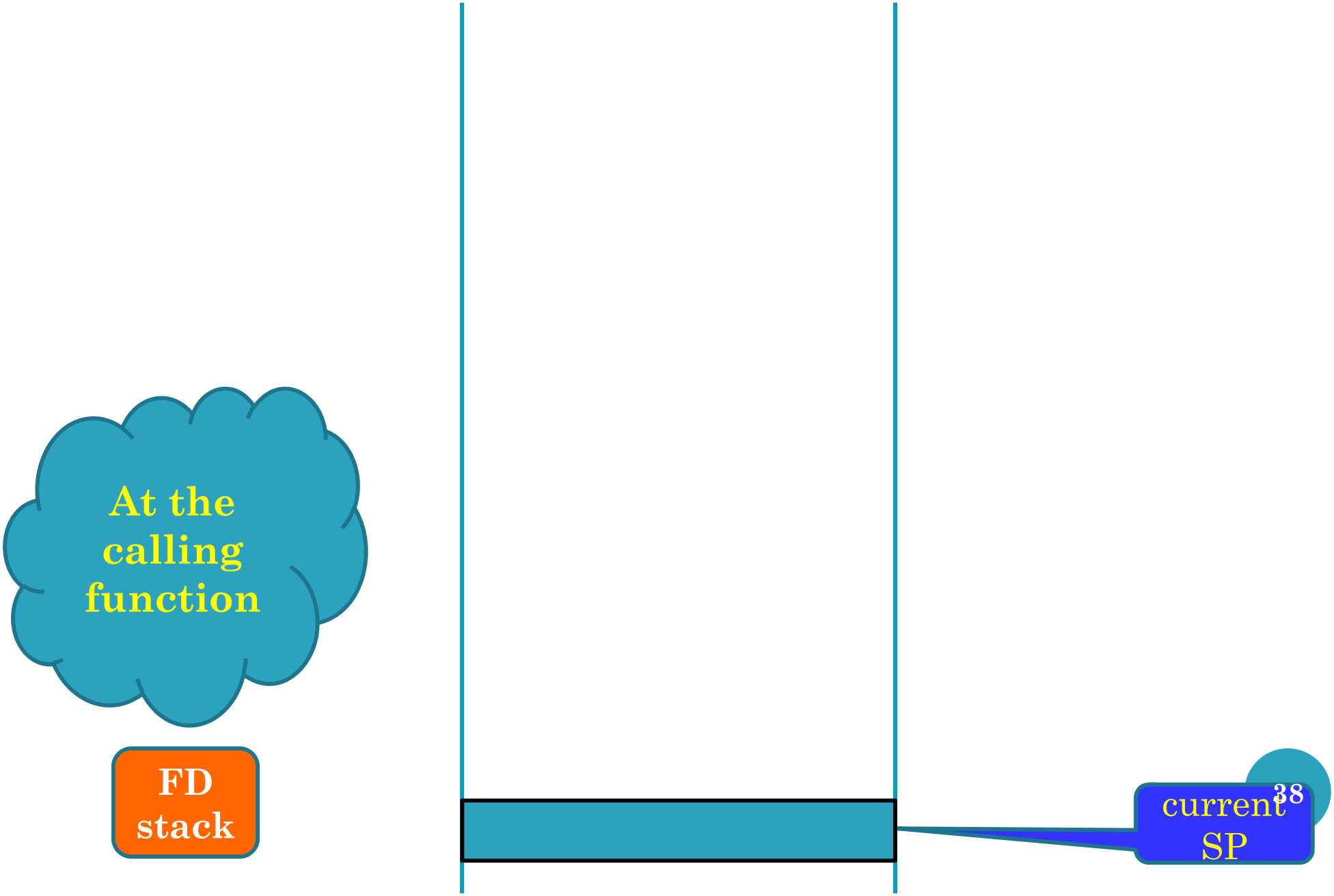
–4
+0
+4
+8

Local variables

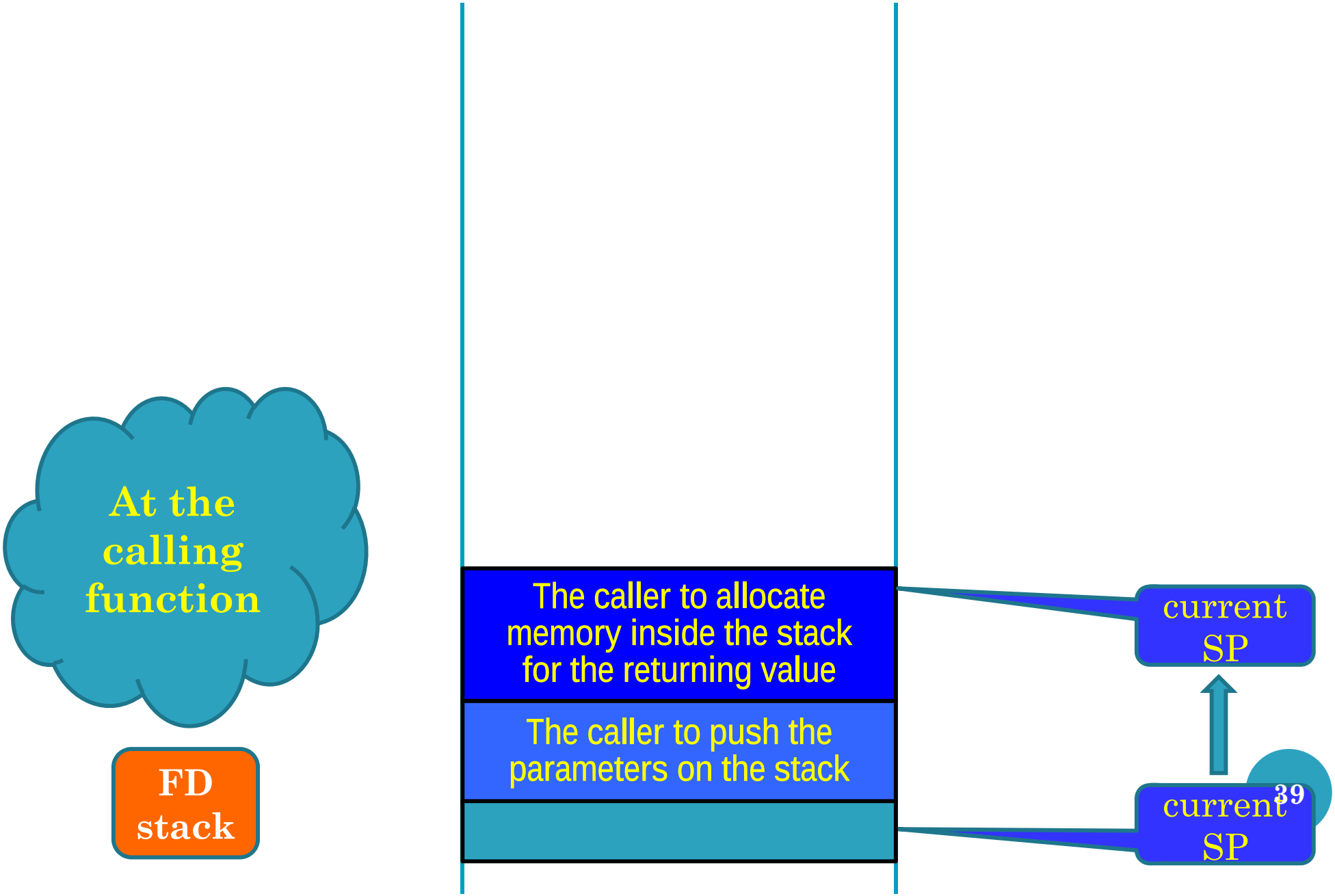
parameters

The swap function should not have a *direct* access to x and y

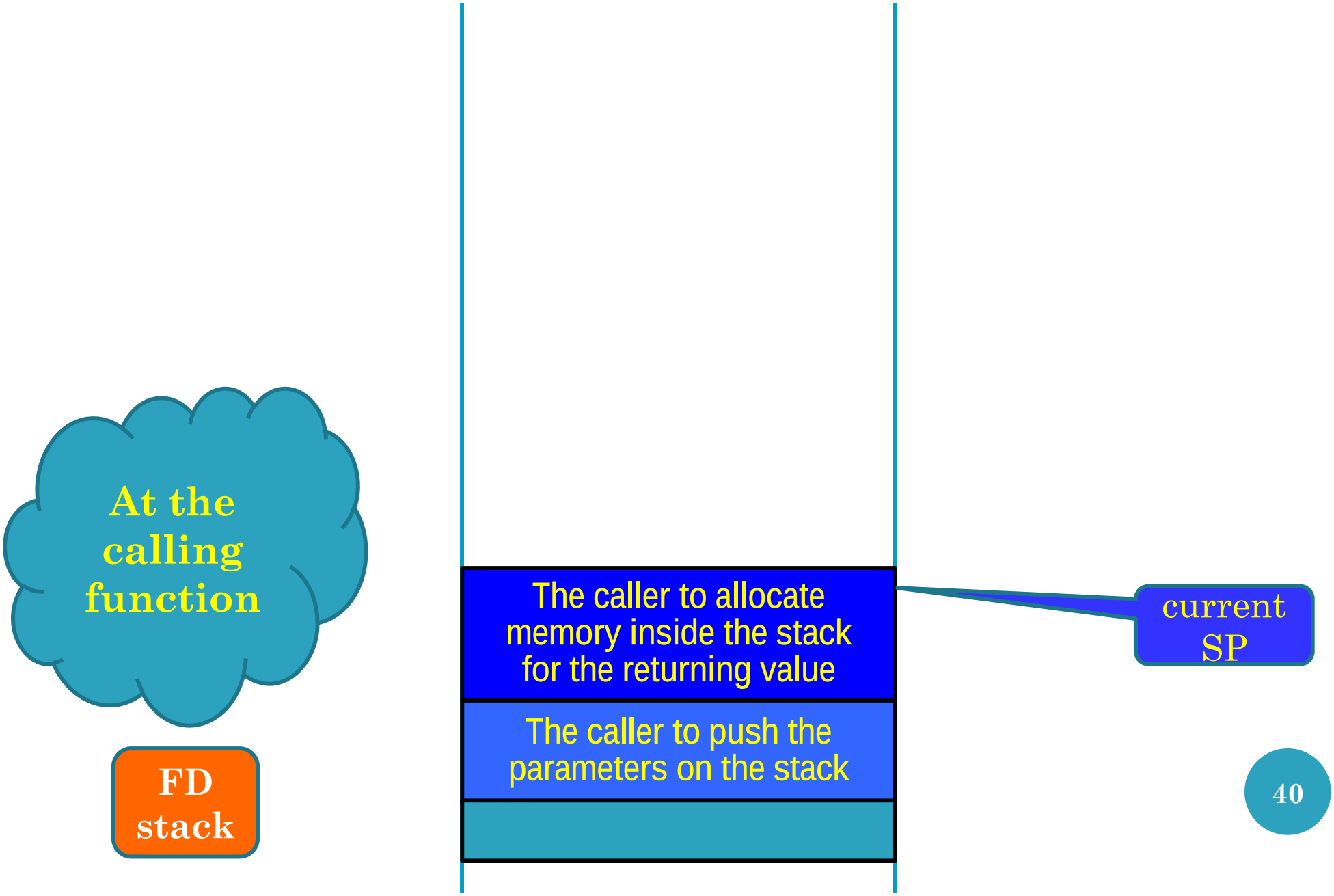
The Traditional Call/Return Mechanism



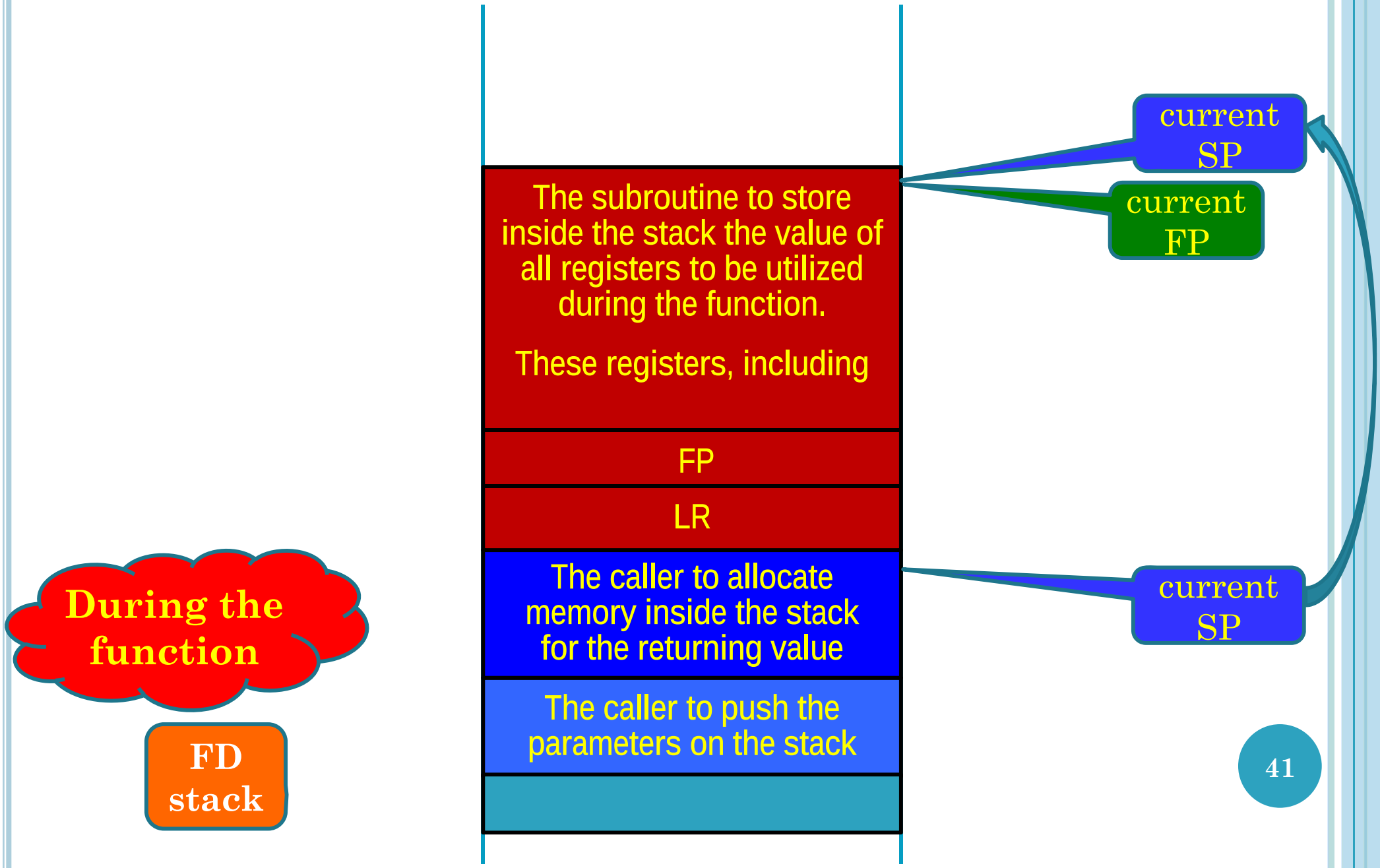
The Traditional Call/Return Mechanism



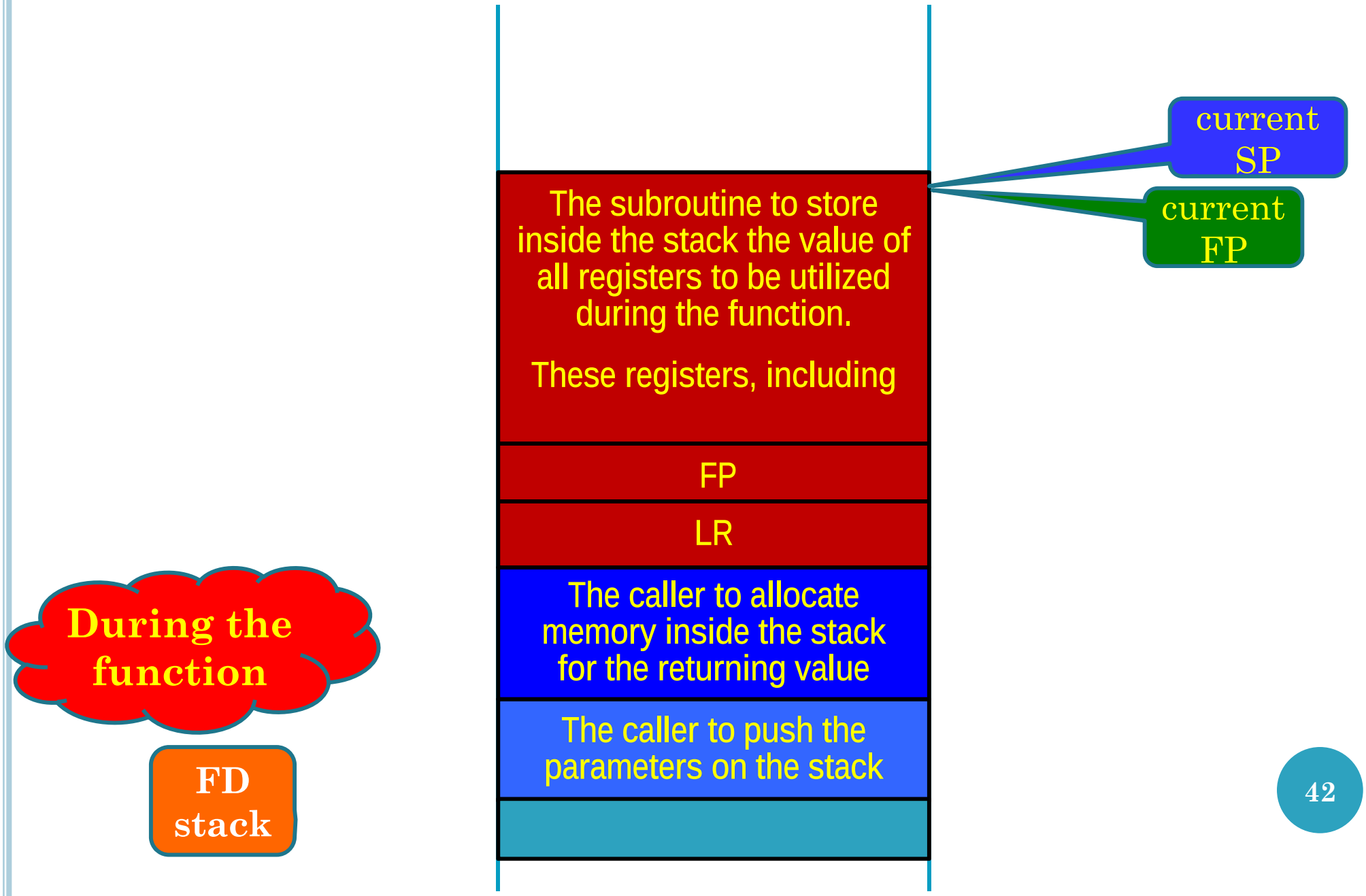
The Traditional Call/Return Mechanism



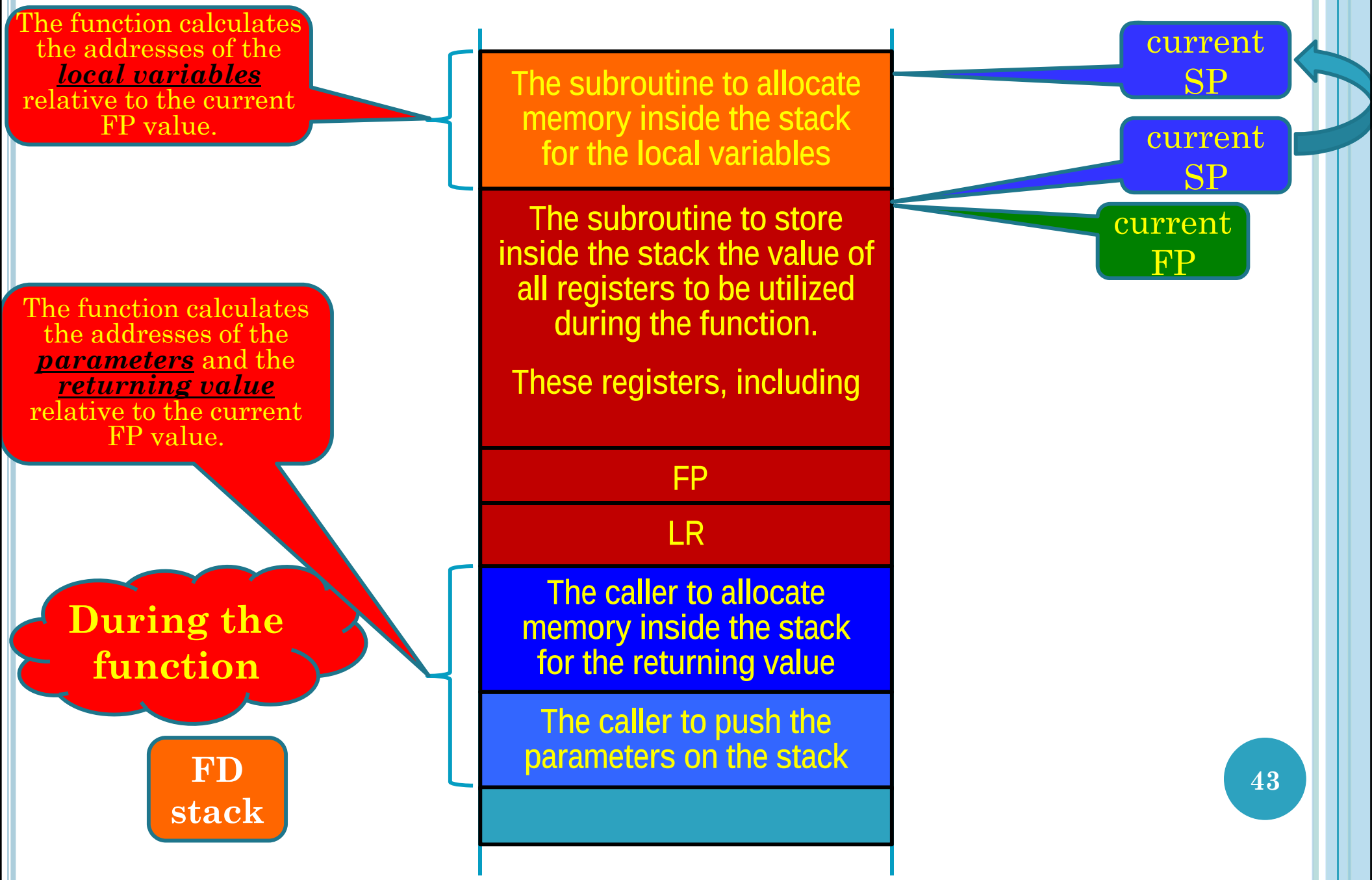
The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism

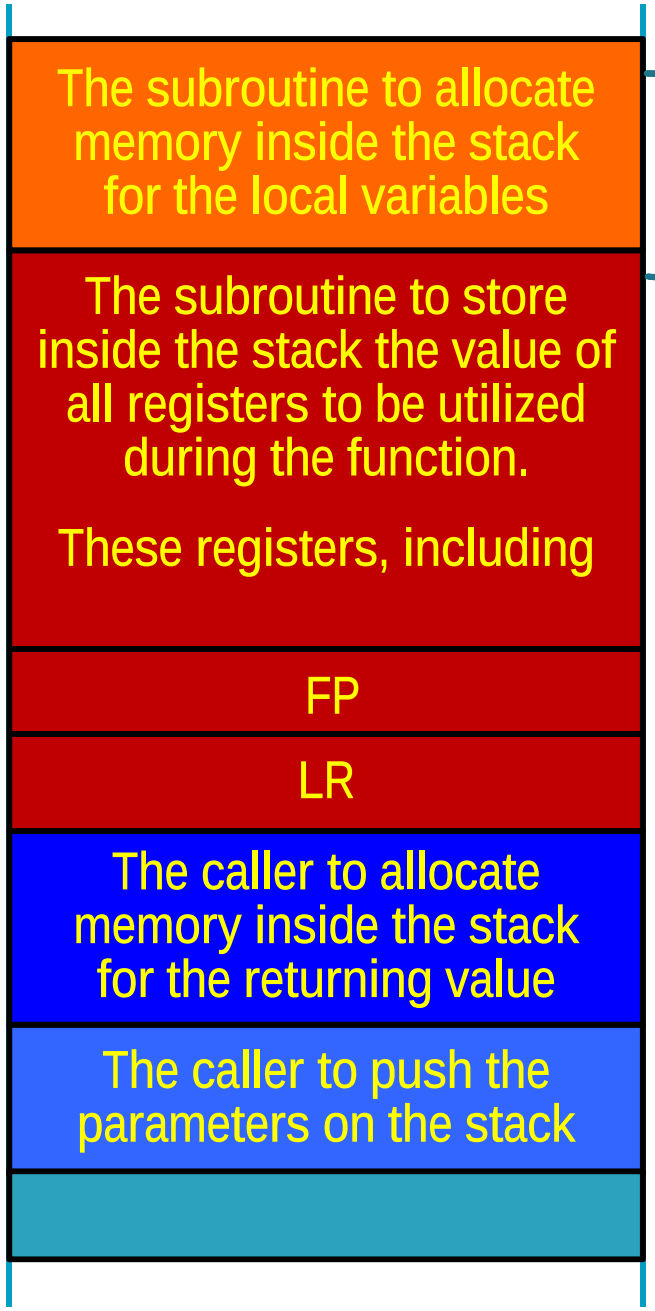
The function calculates the addresses of the ***local variables*** relative to the current FP value.

call by value
vs
call by reference

The function calculates the addresses of the ***parameters*** and the ***returning value*** relative to the current FP value.

During the function

FD stack

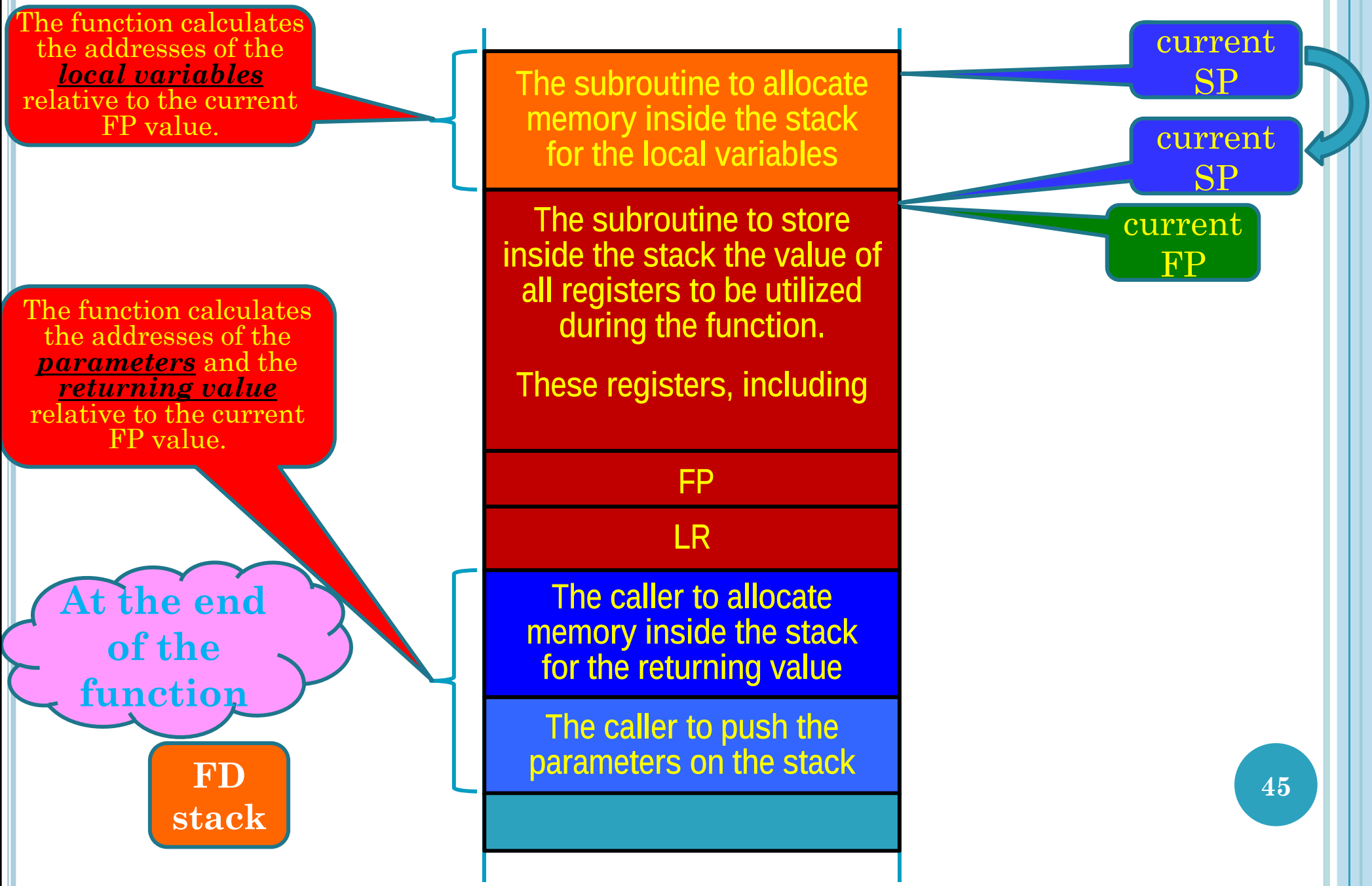


current SP

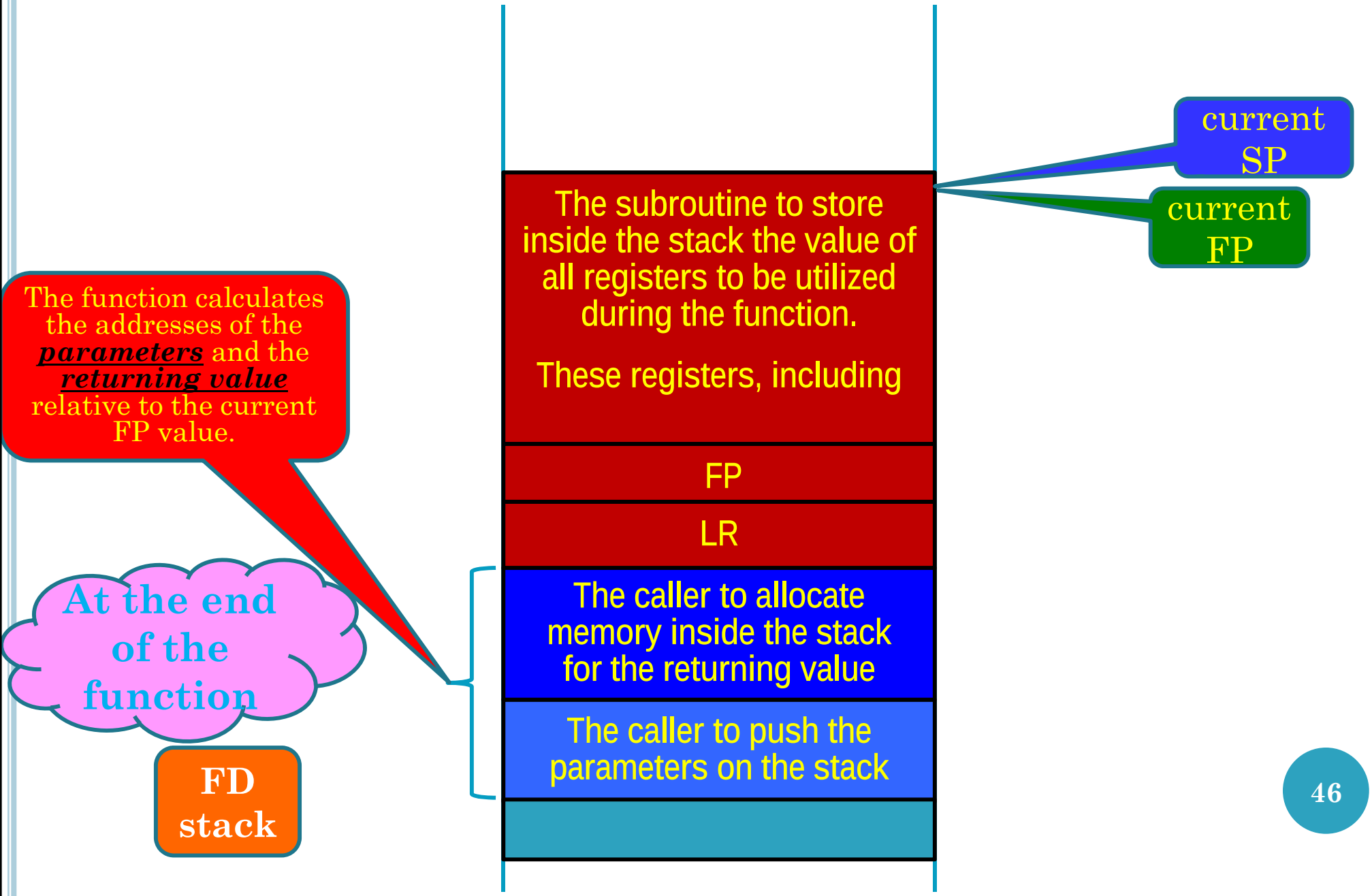
current FP

What will happen if the function called itself? i.e., recursion

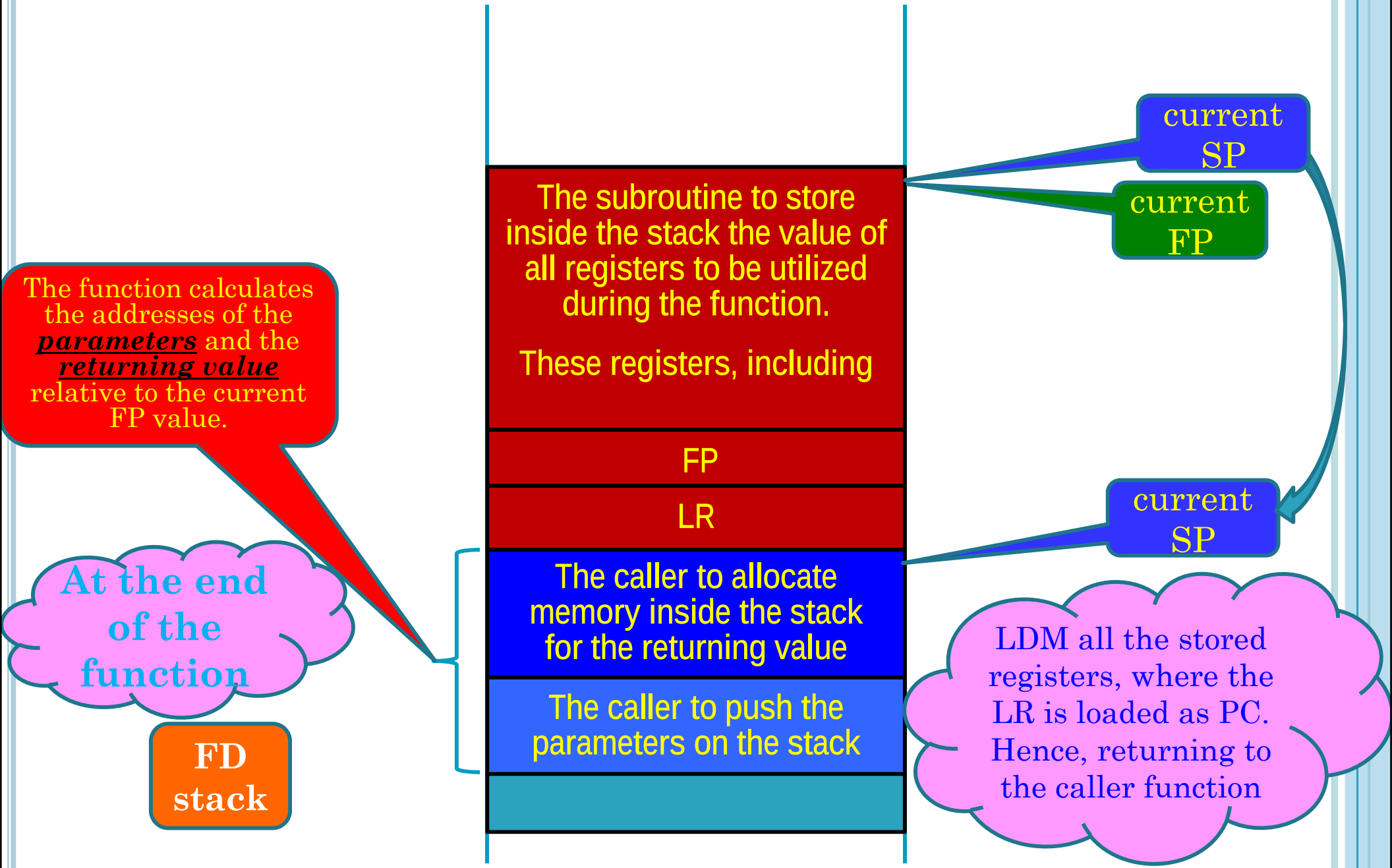
The Traditional Call/Return Mechanism



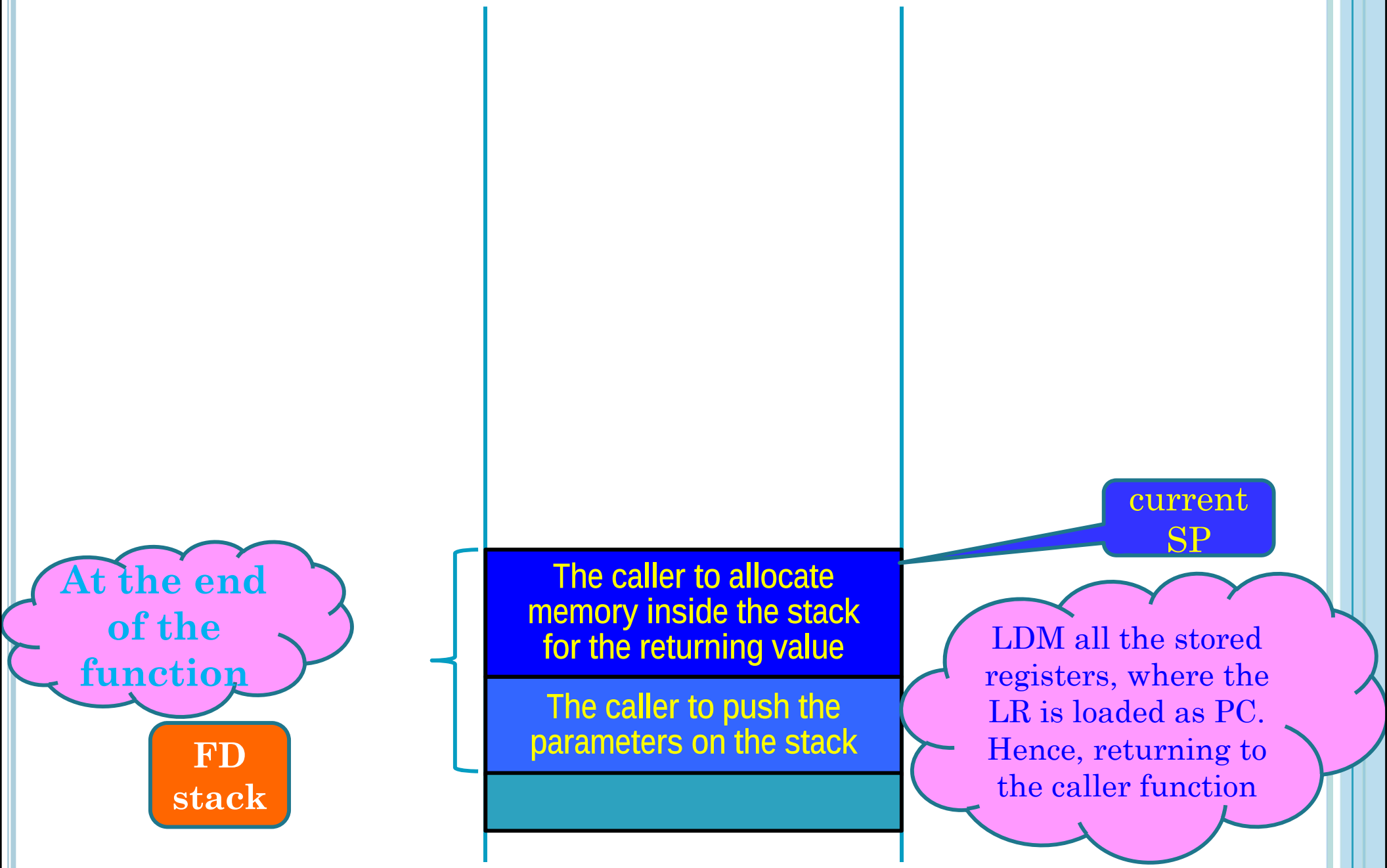
The Traditional Call/Return Mechanism



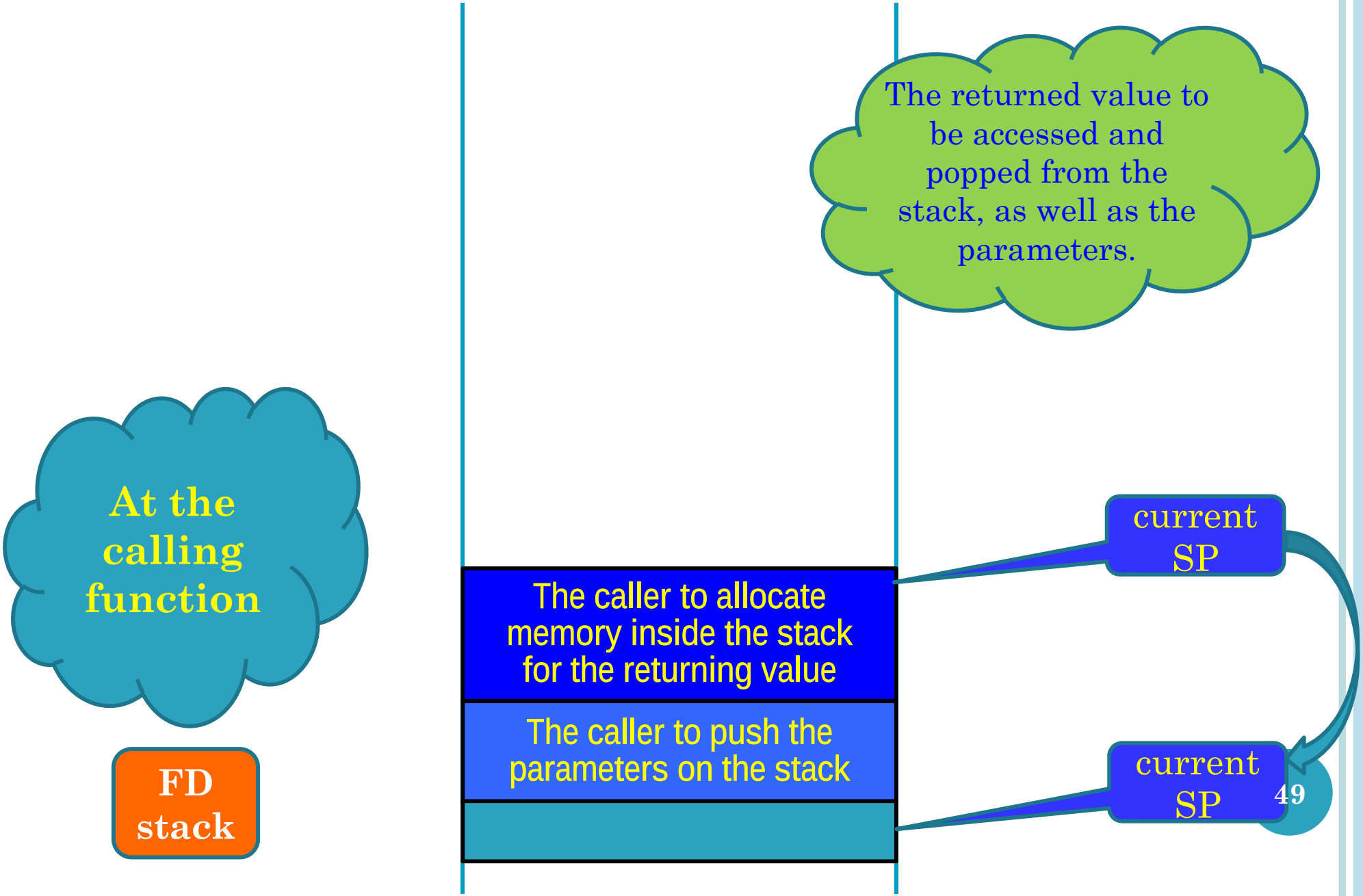
The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism



The Traditional Call/Return Mechanism