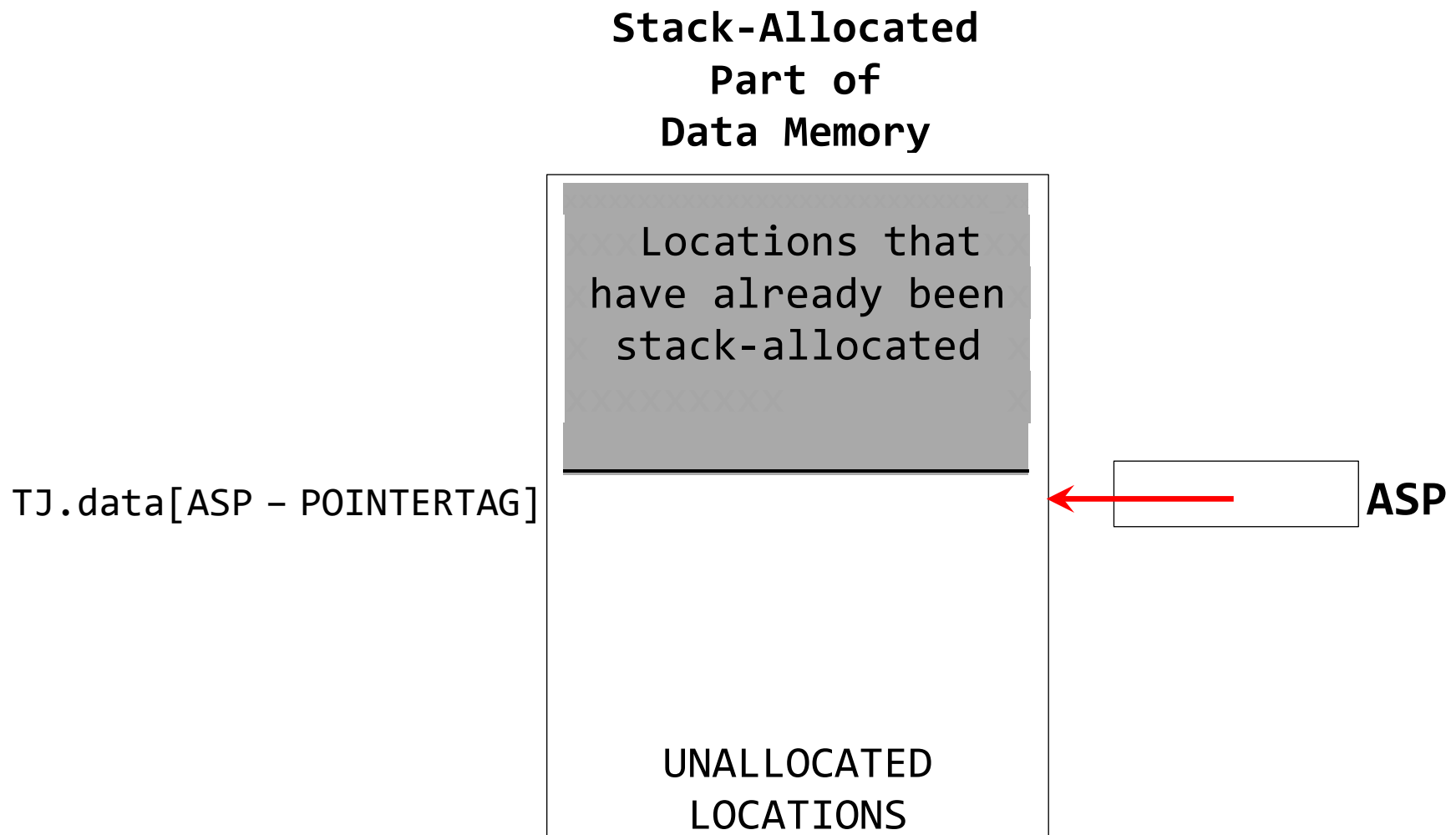


# **Execution of Method Call and Return**

S-PUSH  $y$  is equivalent to:

$TJ.data[ASP - POINTERTAG] = y; \quad ASP++;$

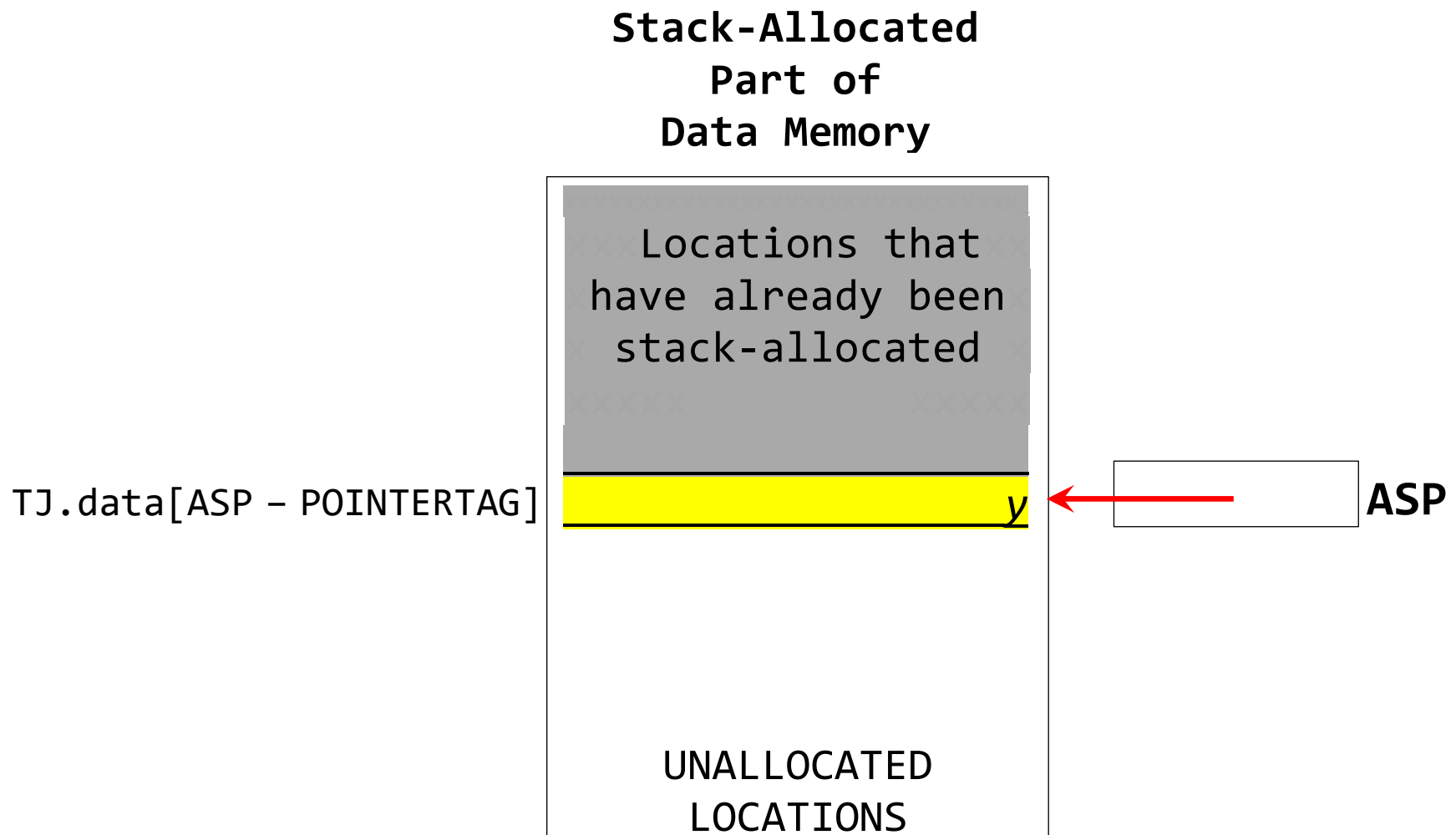
BEFORE execution of S-PUSH  $y$



S-PUSH *y* is equivalent to:

`TJ.data[ASP - POINTERTAG] = y; ASP++;`

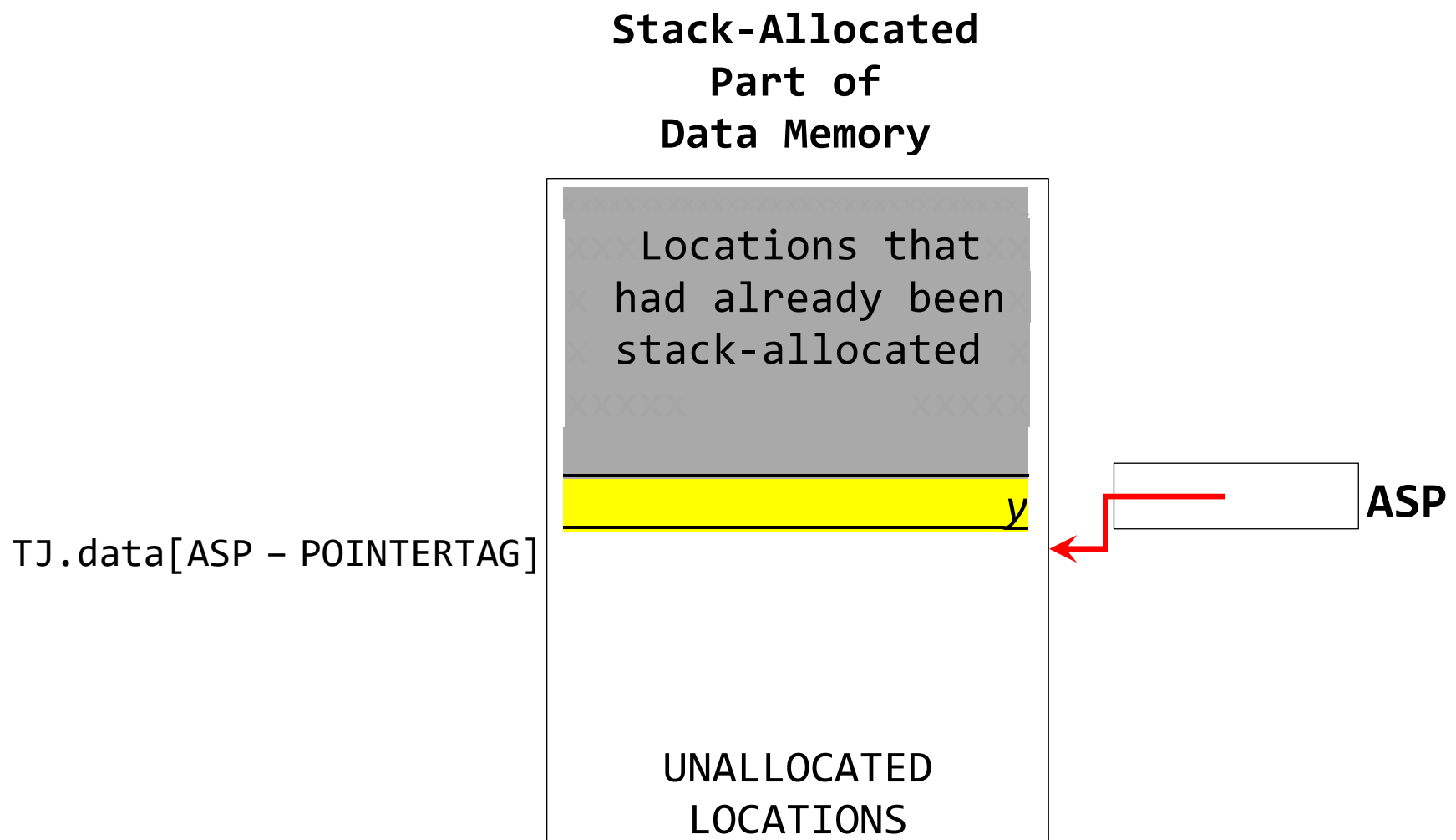
*AFTER execution of* `TJ.data[ASP - POINTERTAG] = y;`



S-PUSH  $y$  is equivalent to:

$TJ.data[ASP - POINTERTAG] = y; \quad ASP++;$

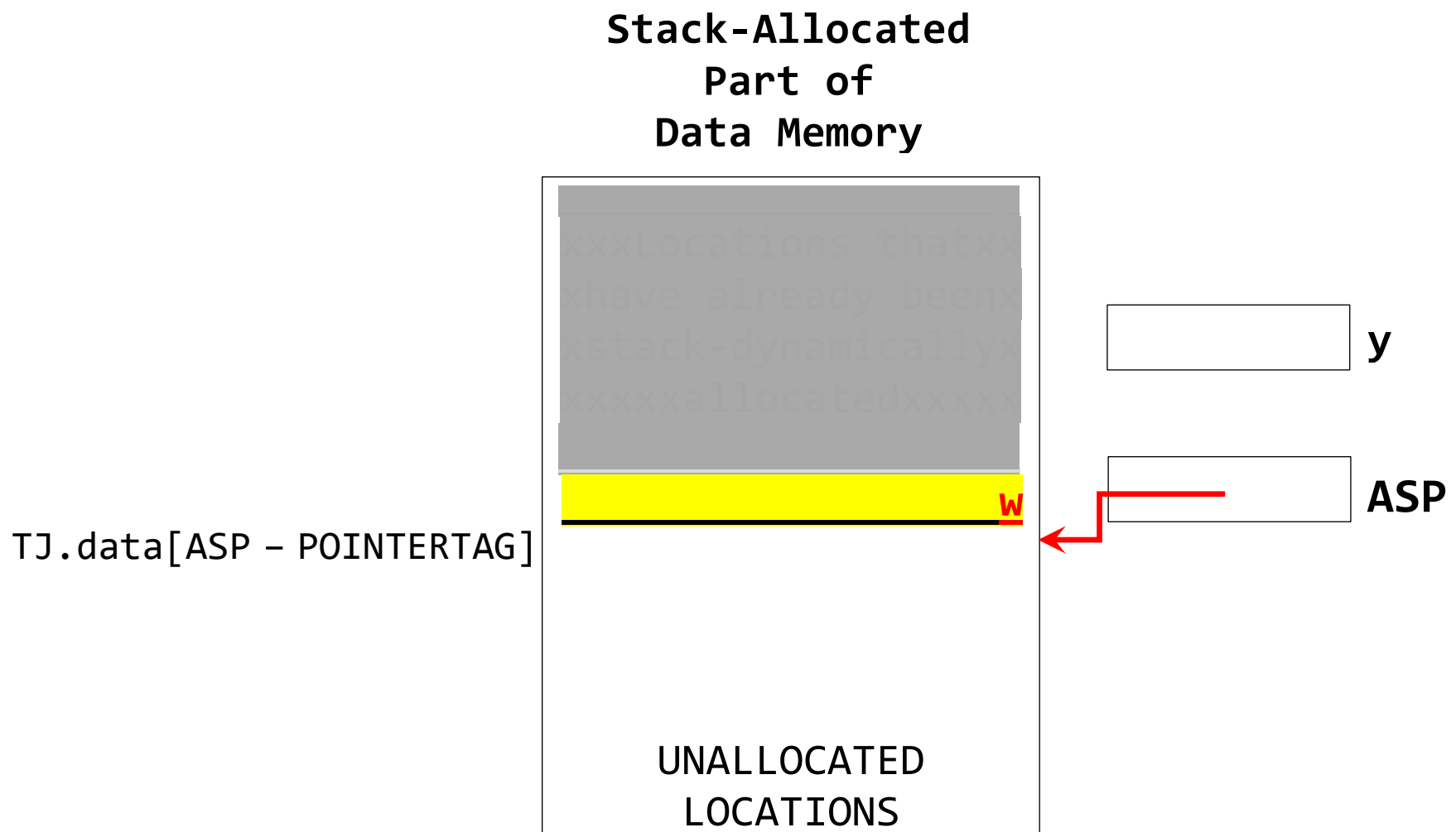
AFTER execution of S-PUSH  $y$



S-POP  $y$  is equivalent to:

--ASP;  $y = \text{TJ.data}[\text{ASP} - \text{POINTERTAG}];$

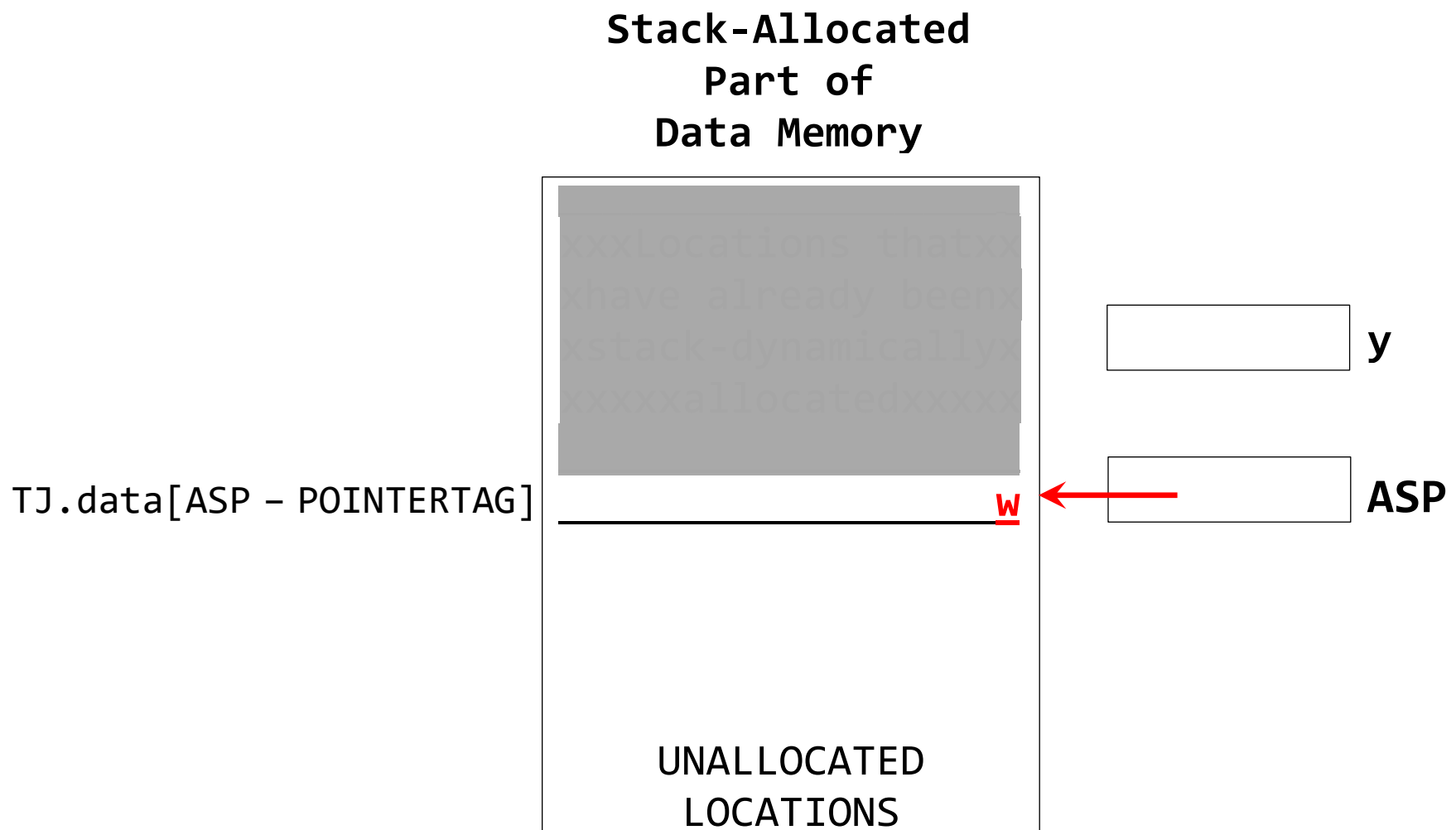
**BEFORE** execution of S-POP  $y$



S-POP  $y$  is equivalent to:

--ASP;  $y = TJ.data[ASP - POINTERTAG];$

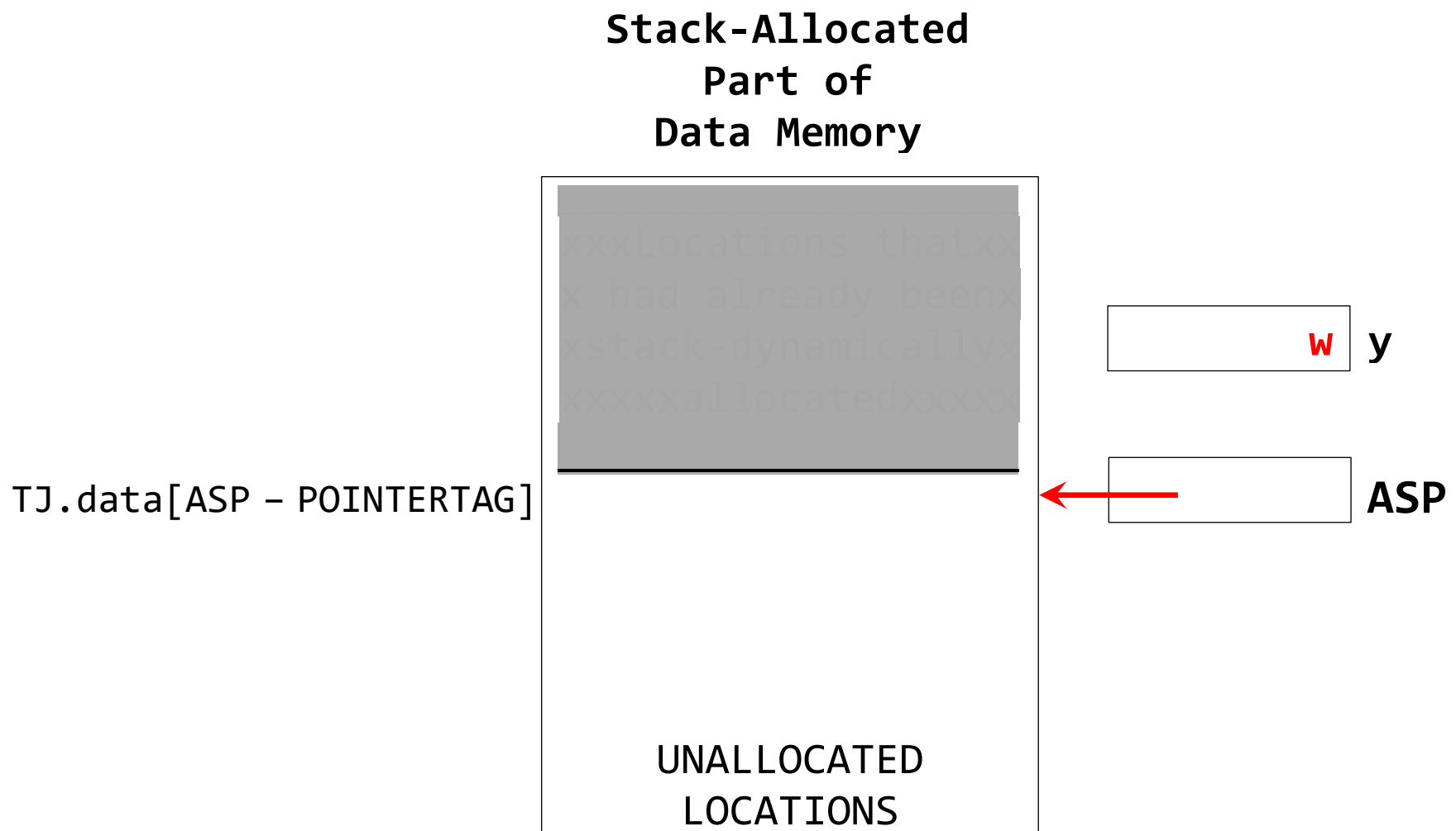
AFTER execution of --ASP;



S-POP  $y$  is equivalent to:

```
--ASP;   $y$  = TJ.data[ASP - POINTERTAG];
```

AFTER execution of S-POP  $y$



## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

--e.g., within: `System.out.print(g(17,23,7,3));`

Suppose further that:

## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

--e.g., within: `System.out.print(g(17,23,7,3));`

Suppose further that:

1. **7** stackframe locations are allocated for local variables declared in **g**'s body.

## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

--e.g., within: `System.out.print(g(17,23,7,3));`

Suppose further that:

1. **7** stackframe locations are allocated for local variables declared in **g**'s body.
2. The code memory address of the first VM instruction generated for method **g** is **671**.

## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

--e.g., within: `System.out.print(g(17,23,7,3));`

Suppose further that:

1. **7** stackframe locations are allocated for local variables declared in **g**'s body.
2. The code memory address of the first VM instruction generated for method **g** is **671**.
3. Method **g** returns control to its caller by executing:  
**713: RETURN 4**

## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

--e.g., within: `System.out.print(g(17,23,7,3));`

Suppose further that:

1. **7** stackframe locations are allocated for local variables declared in **g**'s body.
2. The code memory address of the first VM instruction generated for method **g** is **671**.
3. Method **g** returns control to its caller by executing:  
**713: RETURN 4**
4. The code memory address of the first VM instruction generated for **g(17,23,7,3)** is **44**.

## Diagrams Relating to Sec. 4 on Page 7 of the [TinyJ Assignment 3 Document](#)

Suppose a method **f** calls a method **g** as follows

**g(17,23,7,3)**

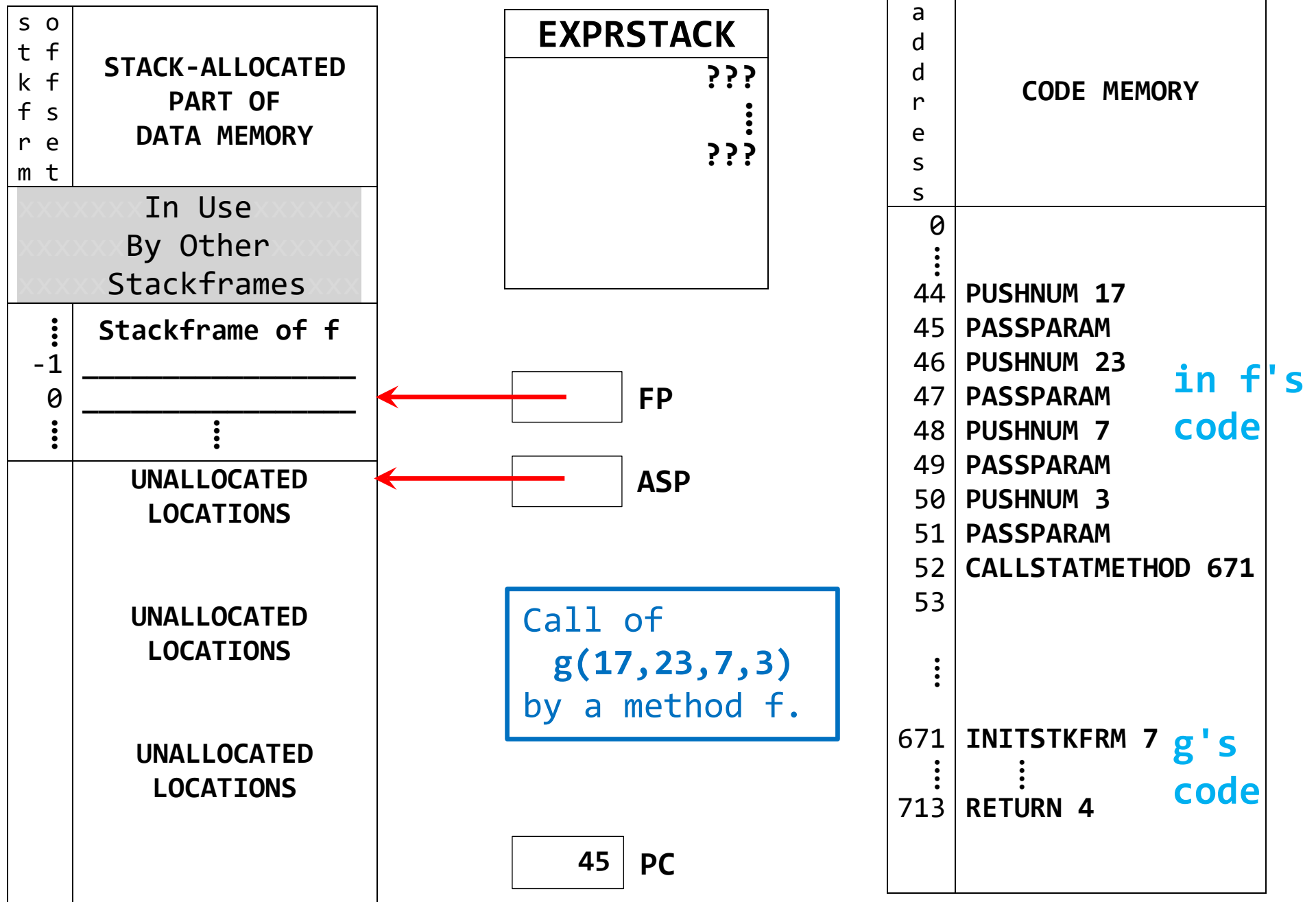
--e.g., within: **System.out.print(g(17,23,7,3));**

Suppose further that:

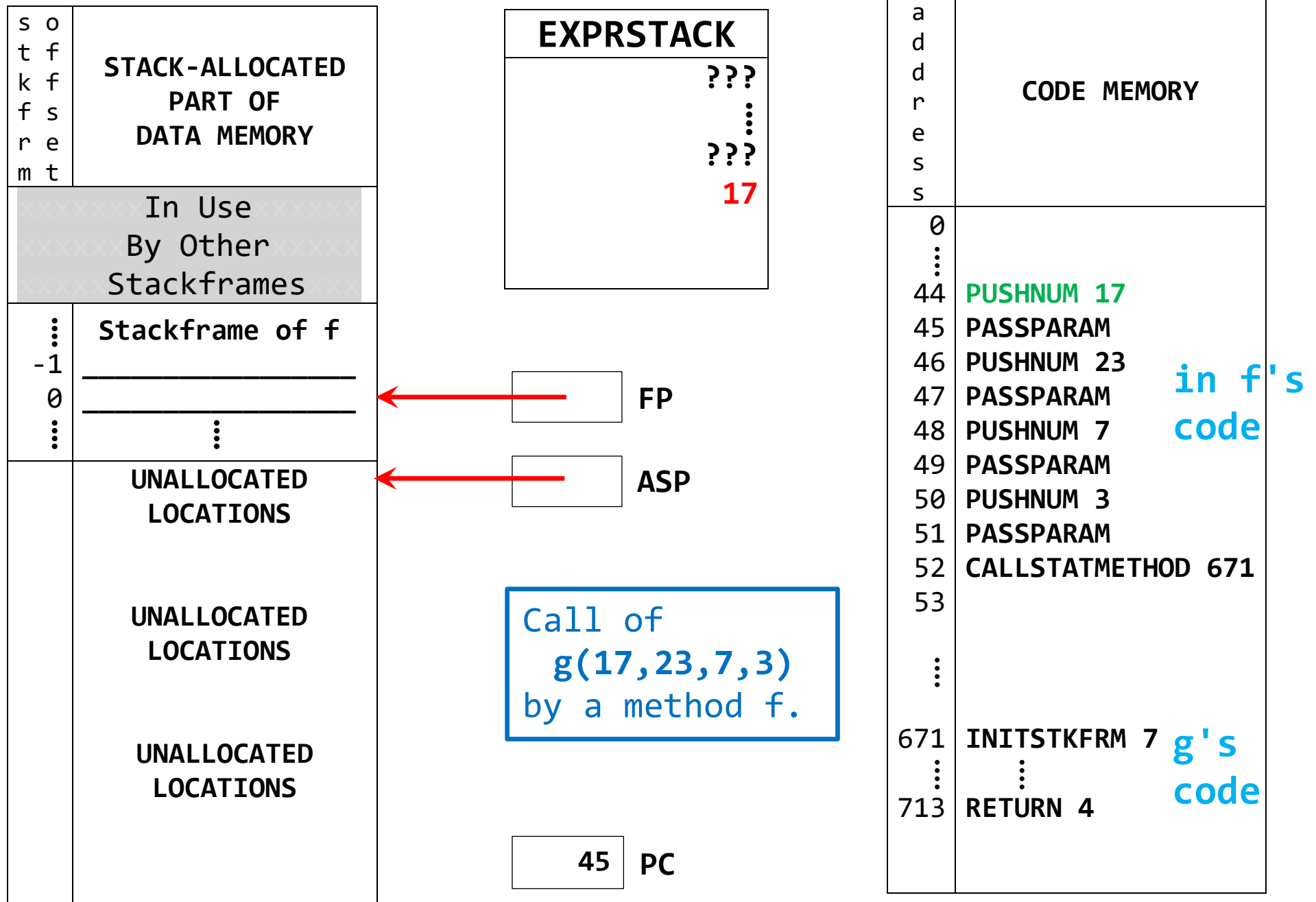
1. **7** stackframe locations are allocated for local variables declared in **g**'s body.
2. The code memory address of the first VM instruction generated for method **g** is **671**.
3. Method **g** returns control to its caller by executing:  
**713: RETURN 4**
4. The code memory address of the first VM instruction generated for **g(17,23,7,3)** is **44**.

The following slides show how the call **g(17,23,7,3)** would be executed, and how **713: RETURN 4** would be executed.

After fetch of (but BEFORE execution of): 44: PUSHNUM 17



## AFTER Execution of 44: PUSHNUM 17

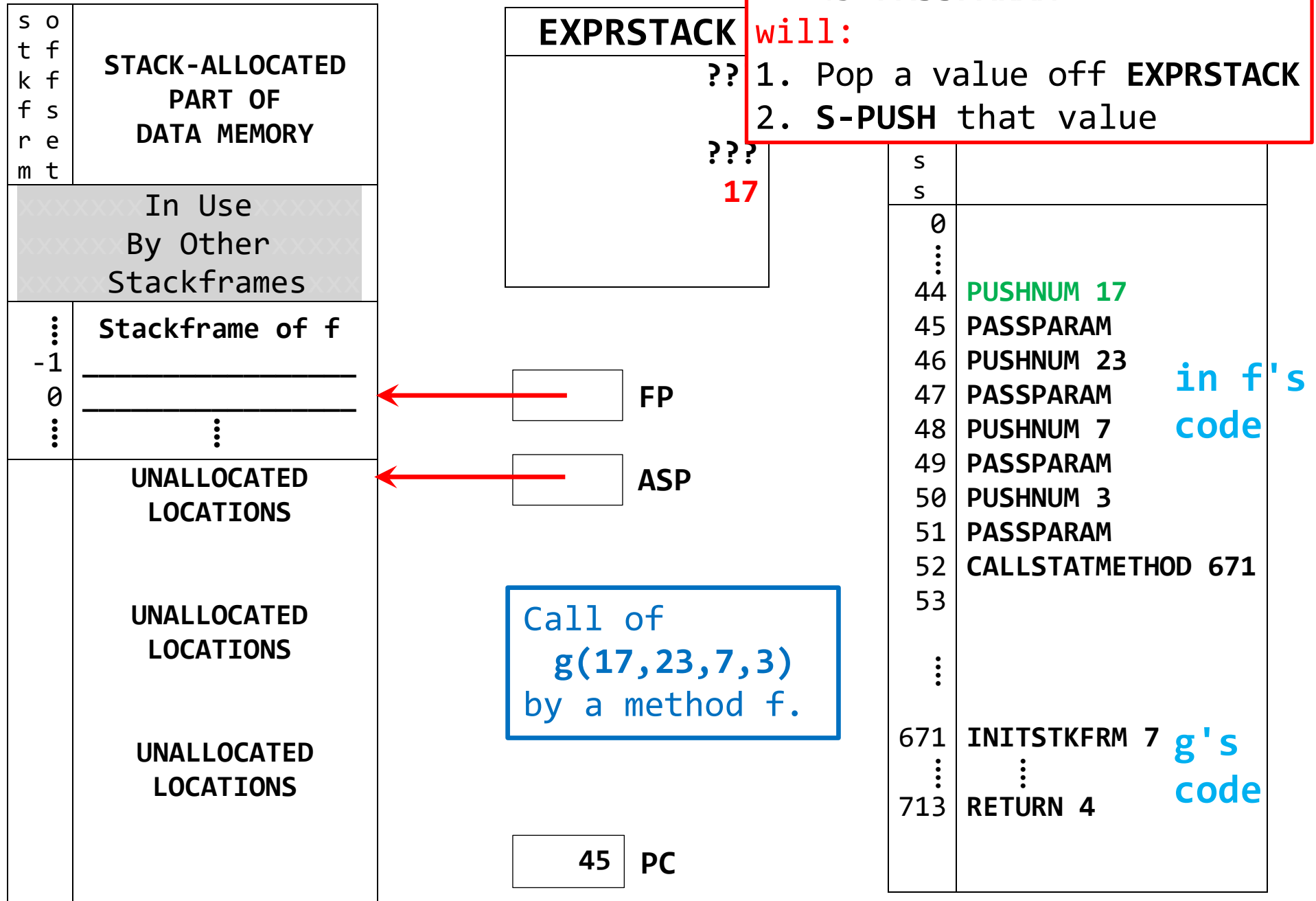


AFTER Execution of 44: PUSHNUM 17

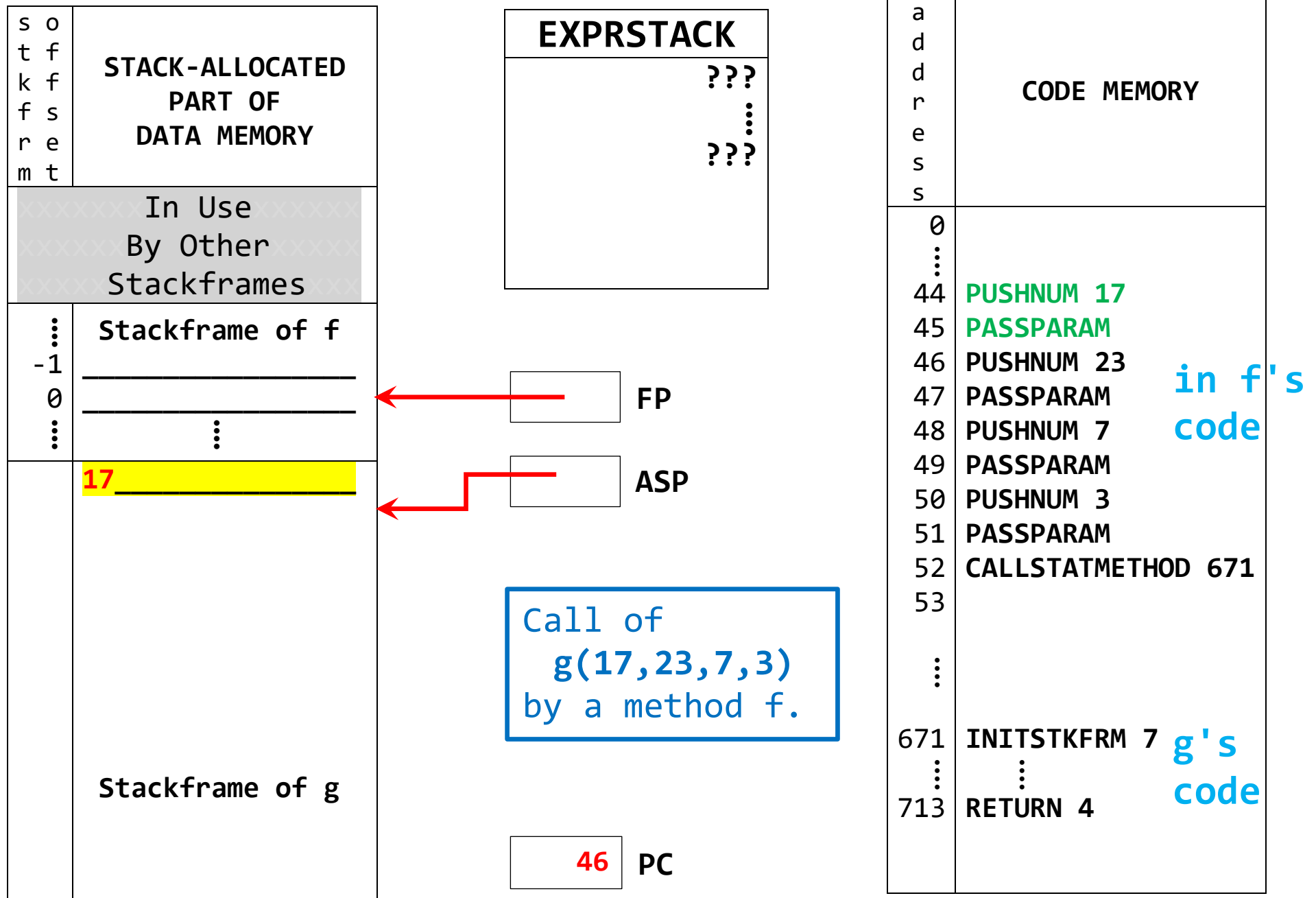
Next, execution of  
45 **PASSPARAM**

will:

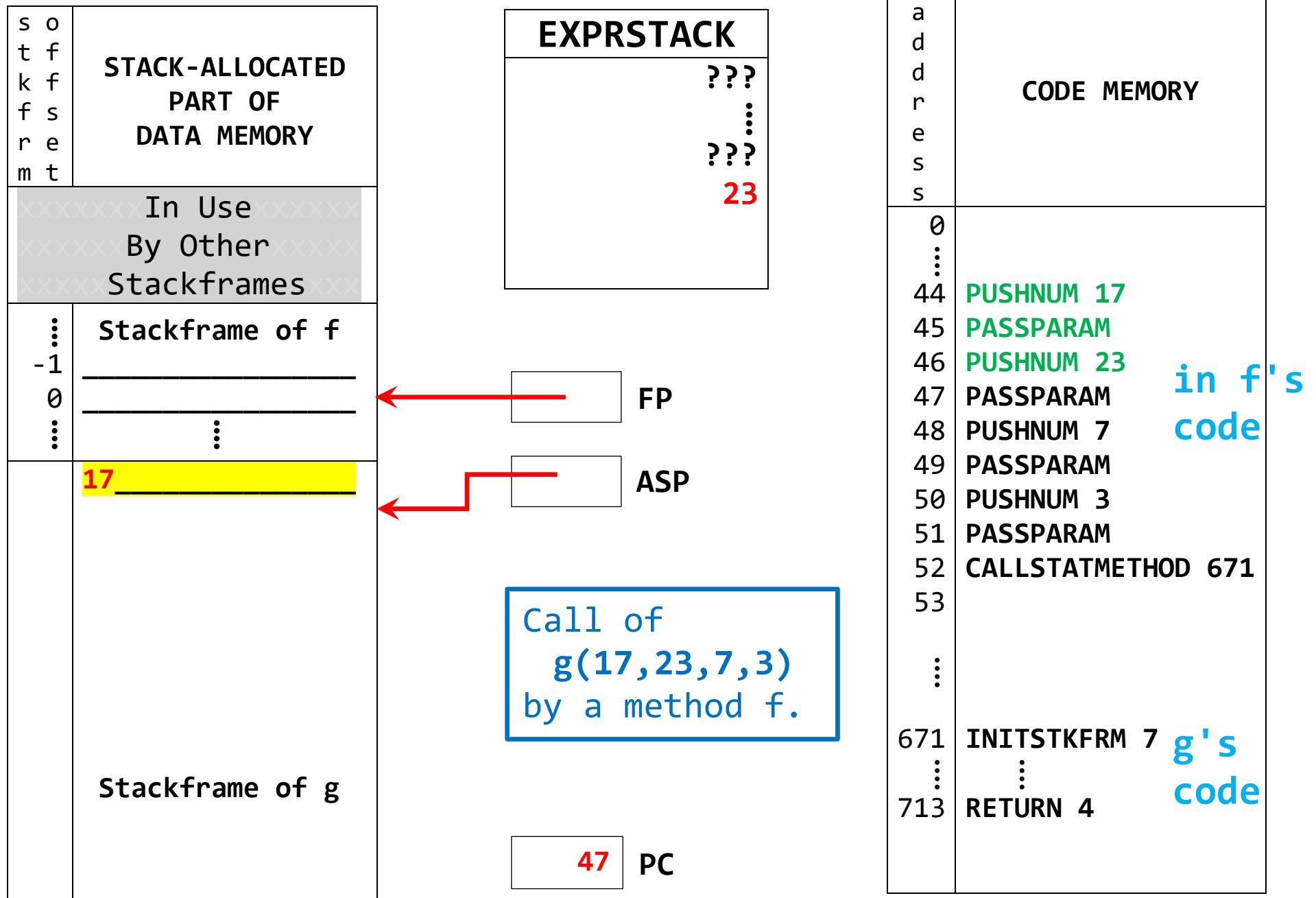
1. Pop a value off **EXPRSTACK**
2. **S-PUSH** that value



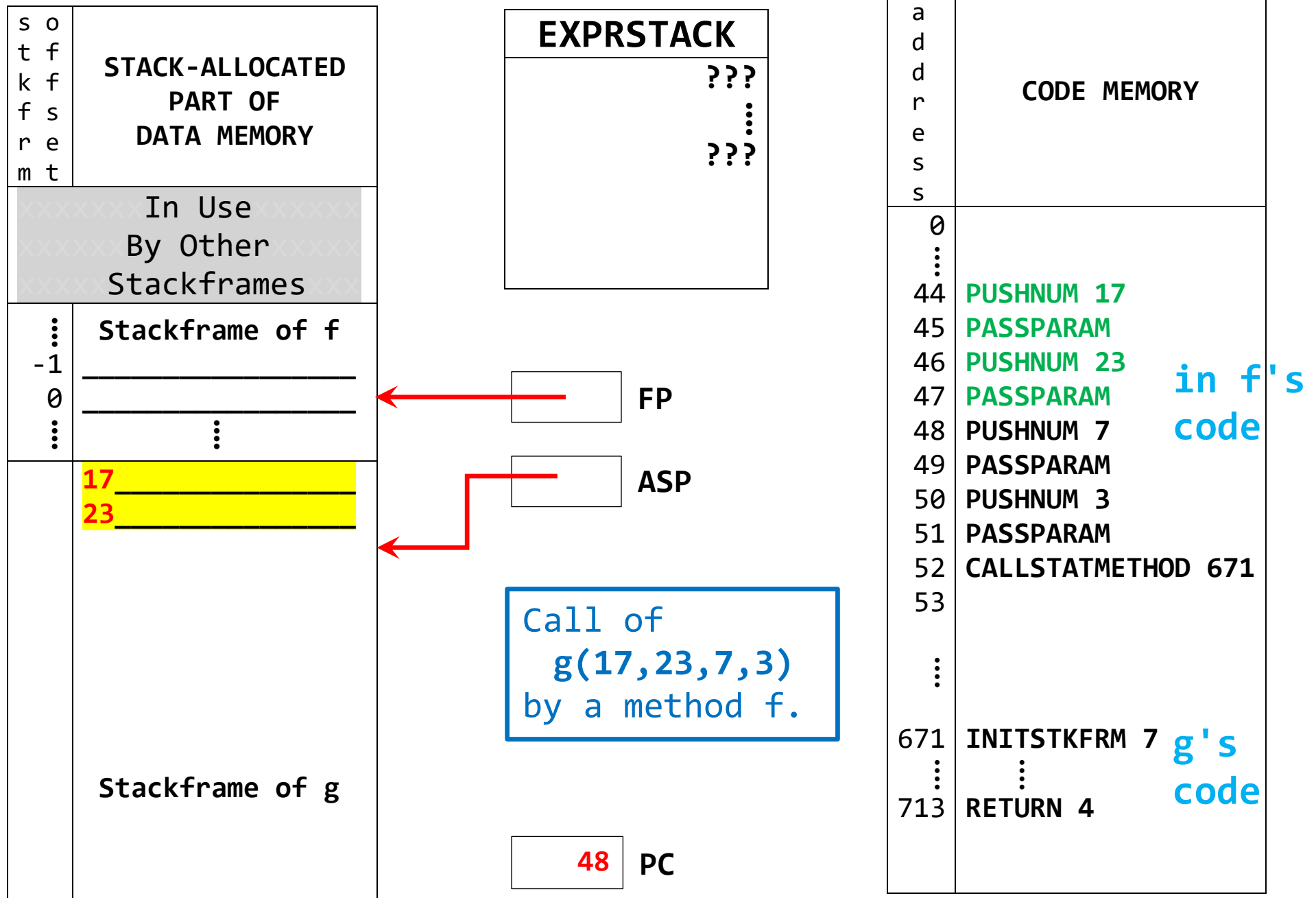
## AFTER Execution of 45: PASSPARAM



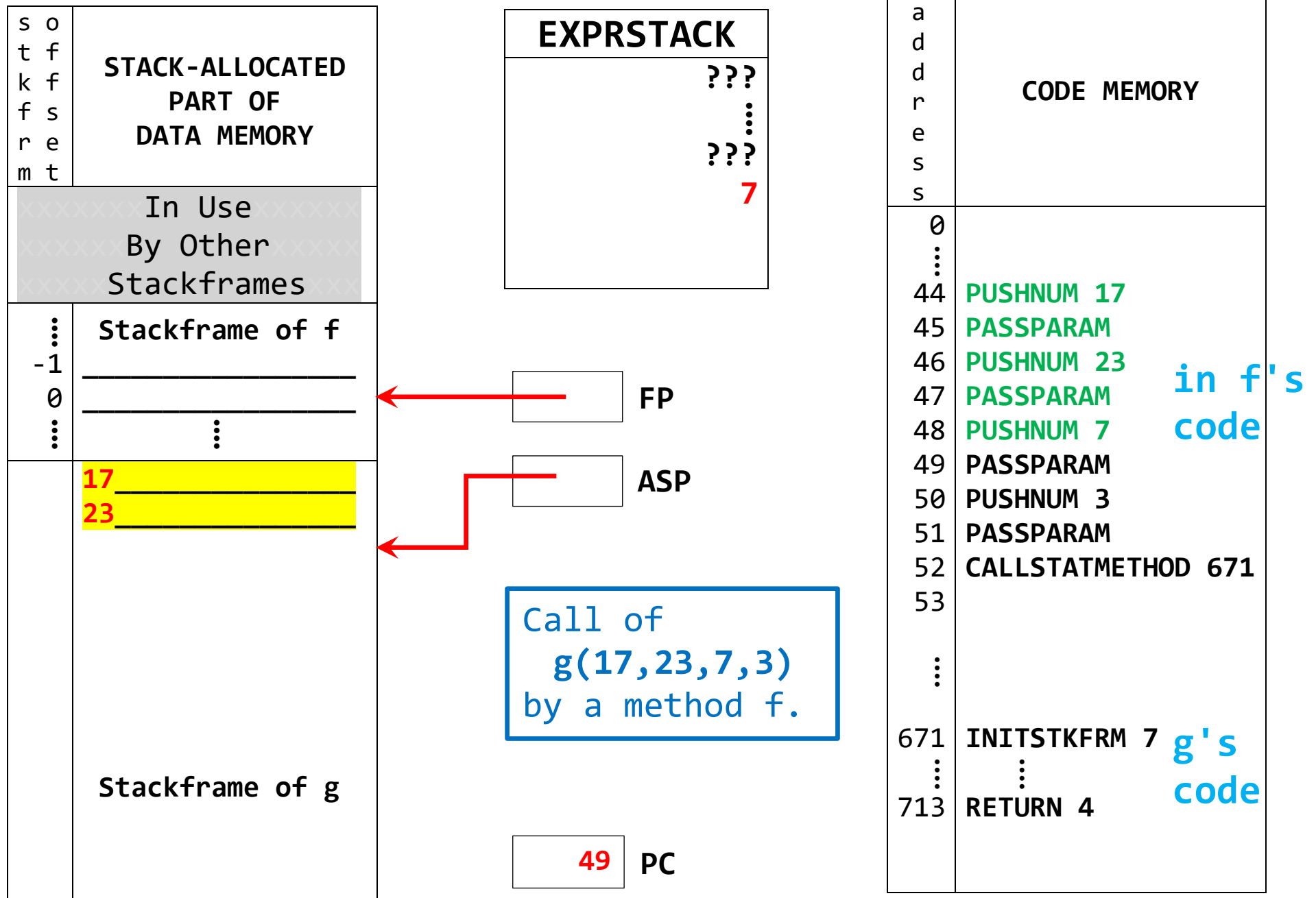
## AFTER Execution of 46: PUSHNUM 23



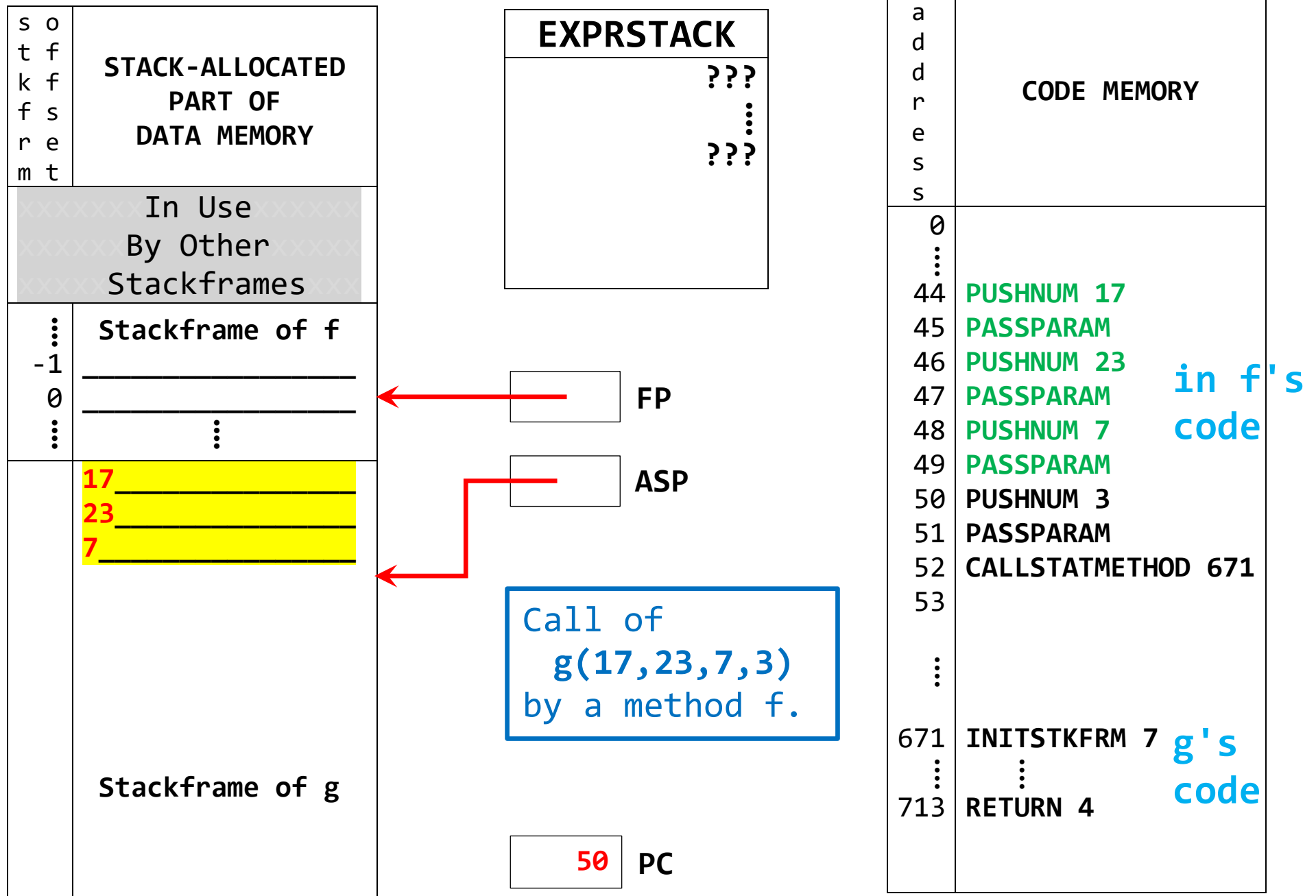
## AFTER Execution of 47: PASSPARAM



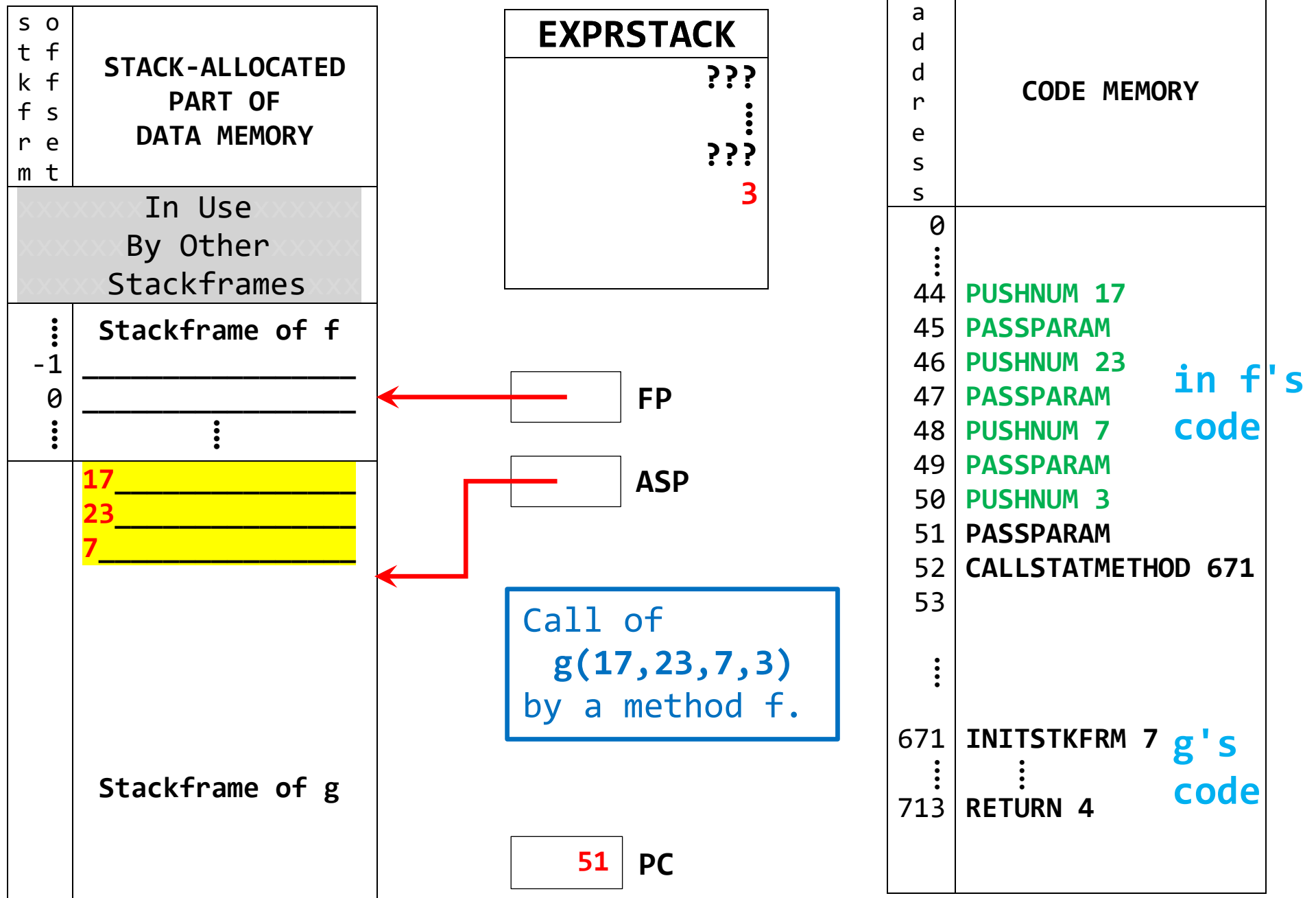
## AFTER Execution of 48: PUSHNUM 7



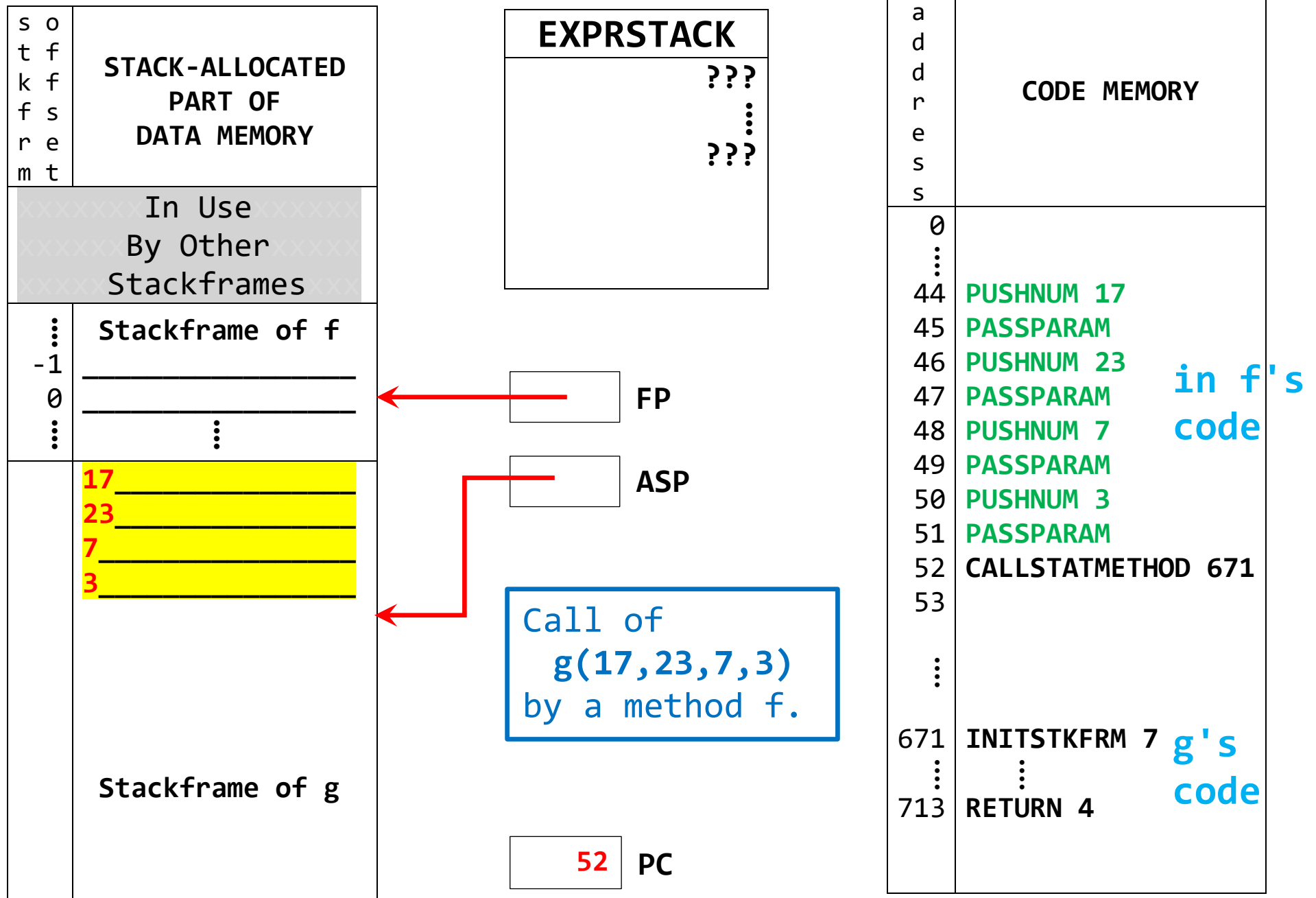
## AFTER Execution of 49: PASSPARAM



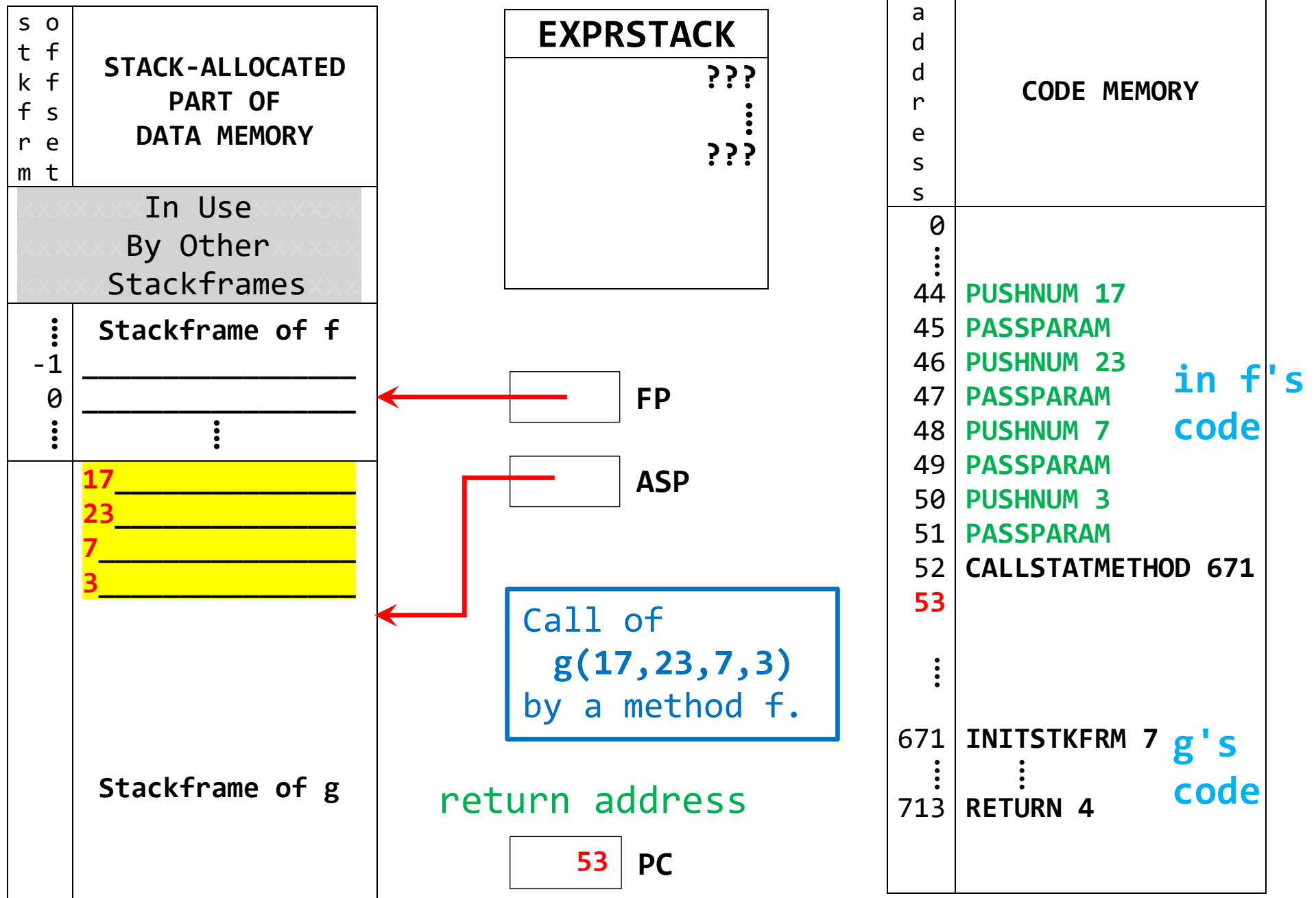
## AFTER Execution of 50: PUSHNUM 3



## AFTER Execution of 51: PASSPARAM



## After FETCH (BEFORE Execution) of: 52: CALLSTATMETHOD 671

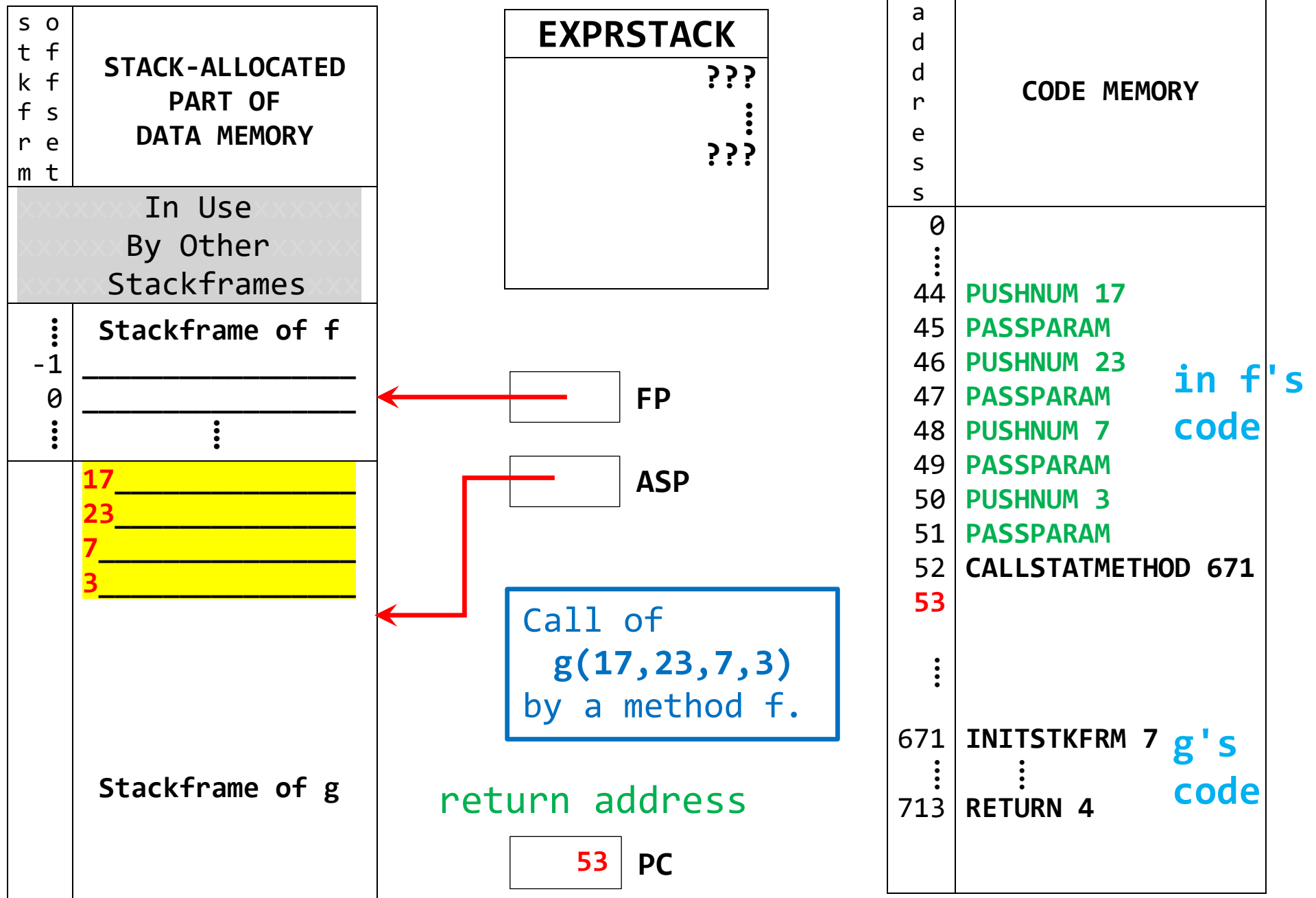


## 52: CALLSTATMETHOD 671

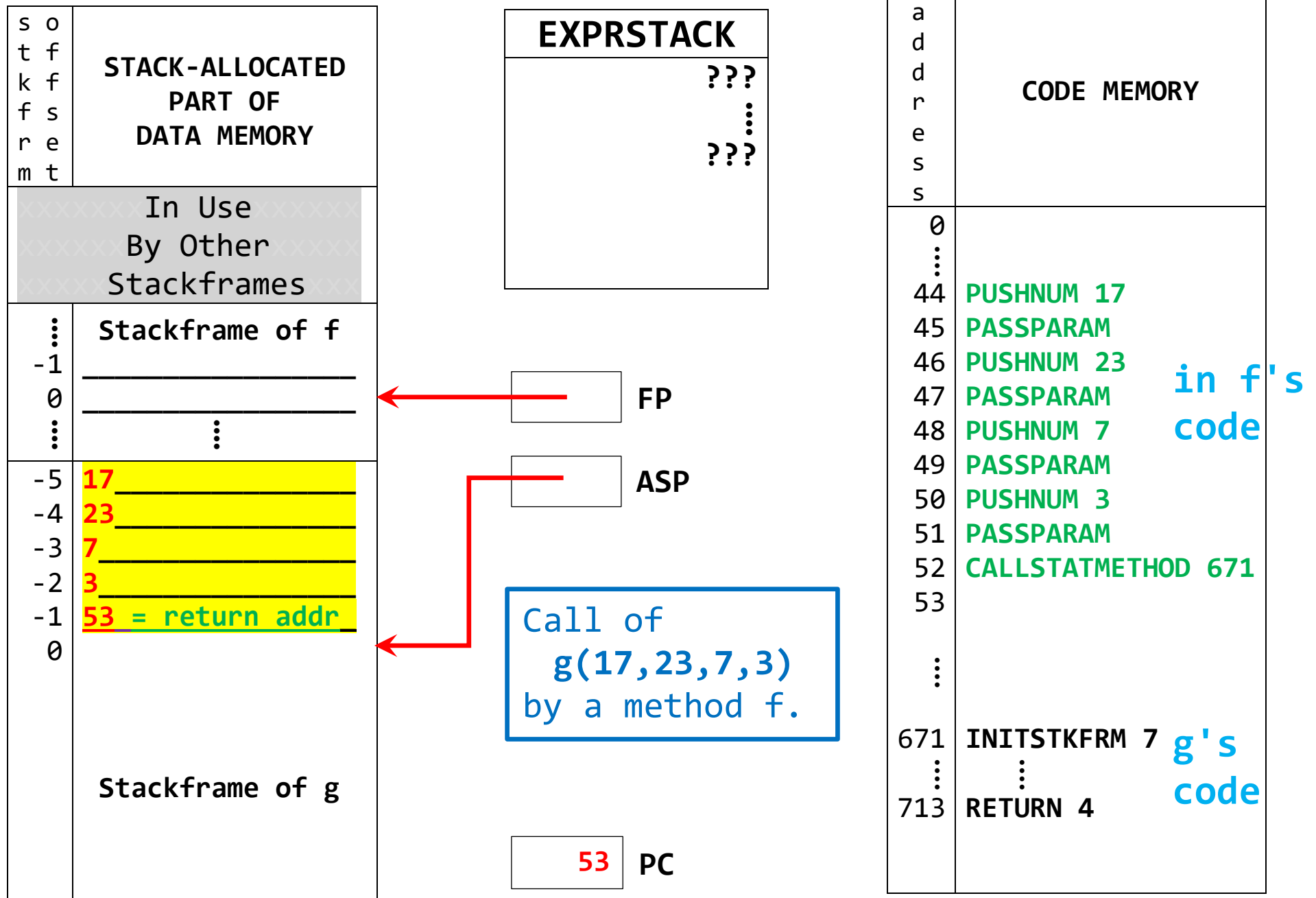
should     **S-PUSH PC** [saves return addr (here, **53**) into new frame\*]  
and then    **set PC to 671** [transfers control to g's code]

\*into the loc. at offset -1 of the new frame--see p. 4 of  
the following document (which is on Brightspace):  
**Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf**

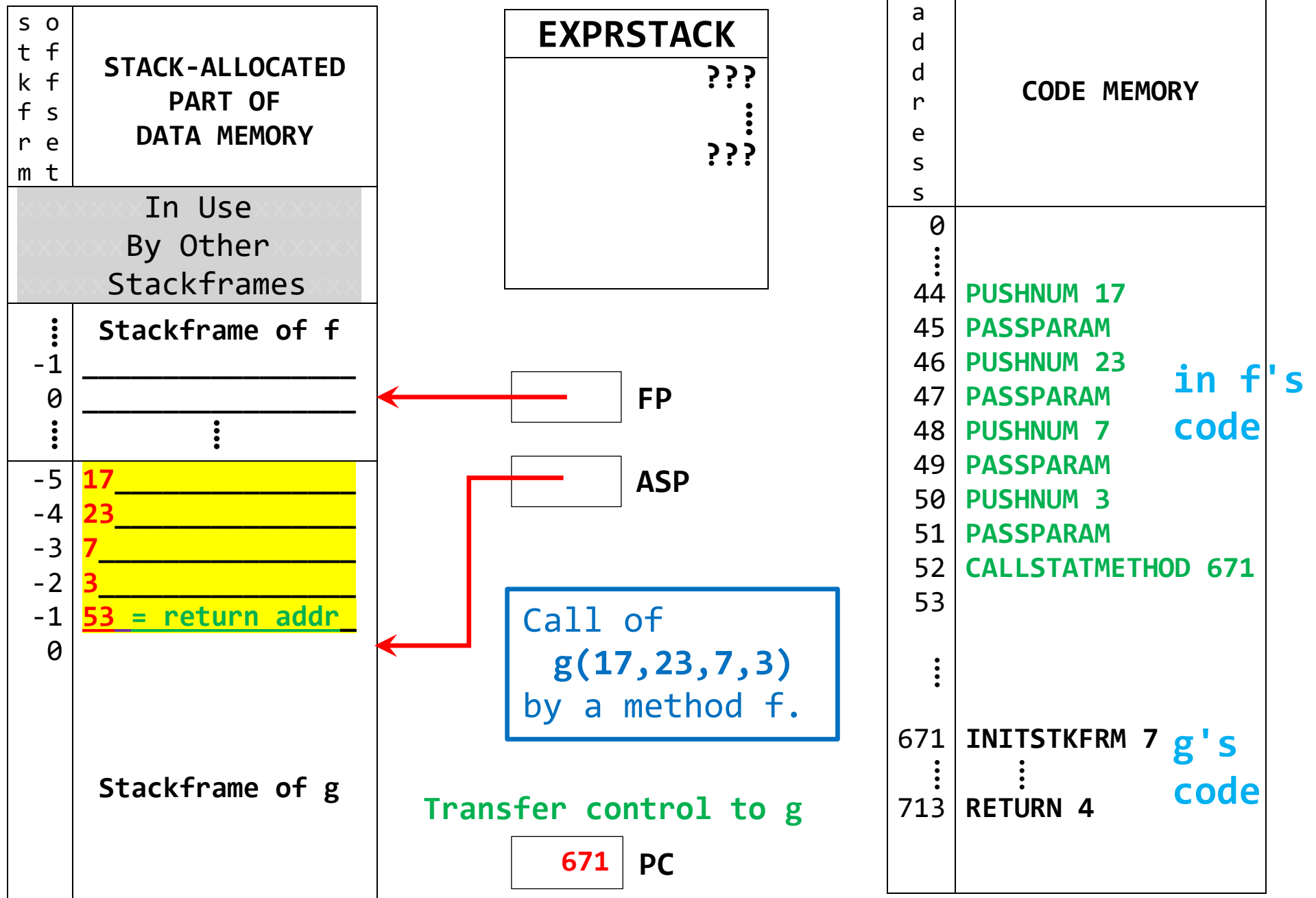
# After FETCH (BEFORE Execution) of: 52: CALLSTATMETHOD 671



# AFTER S-PUSH PC during Execution of 52: CALLSTATMETHOD 671



## AFTER Execution of 52: CALLSTATMETHOD 671



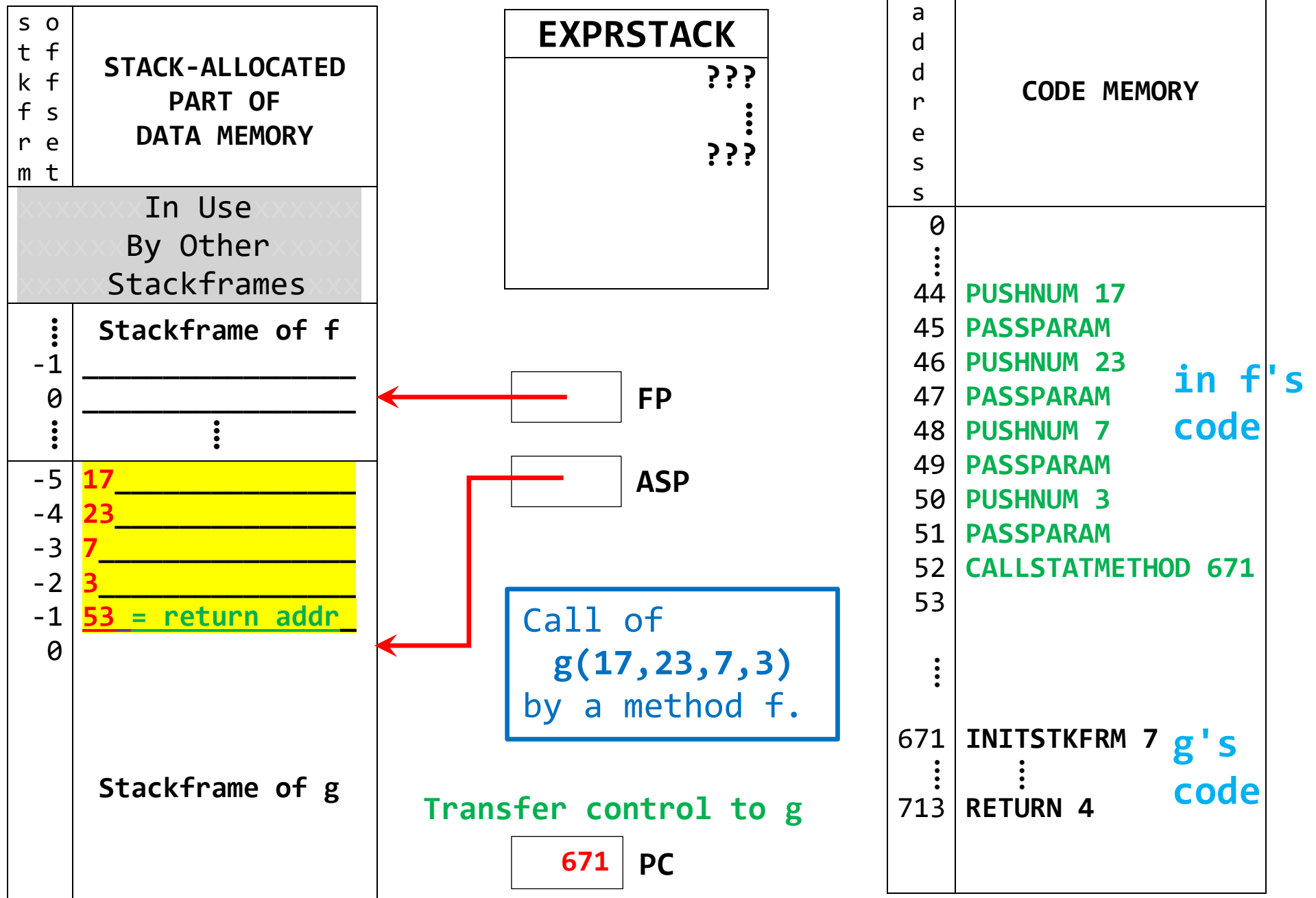
## 671: INITSTKFRM 7

should **S-PUSH FP** [saves caller's FP at offset 0\* in the new frame]  
and then **set FP to ASP - 1** [makes FP point to offset 0\* in the new frame]  
and then **increase ASP by 7** [allocates space for callee's local variables]

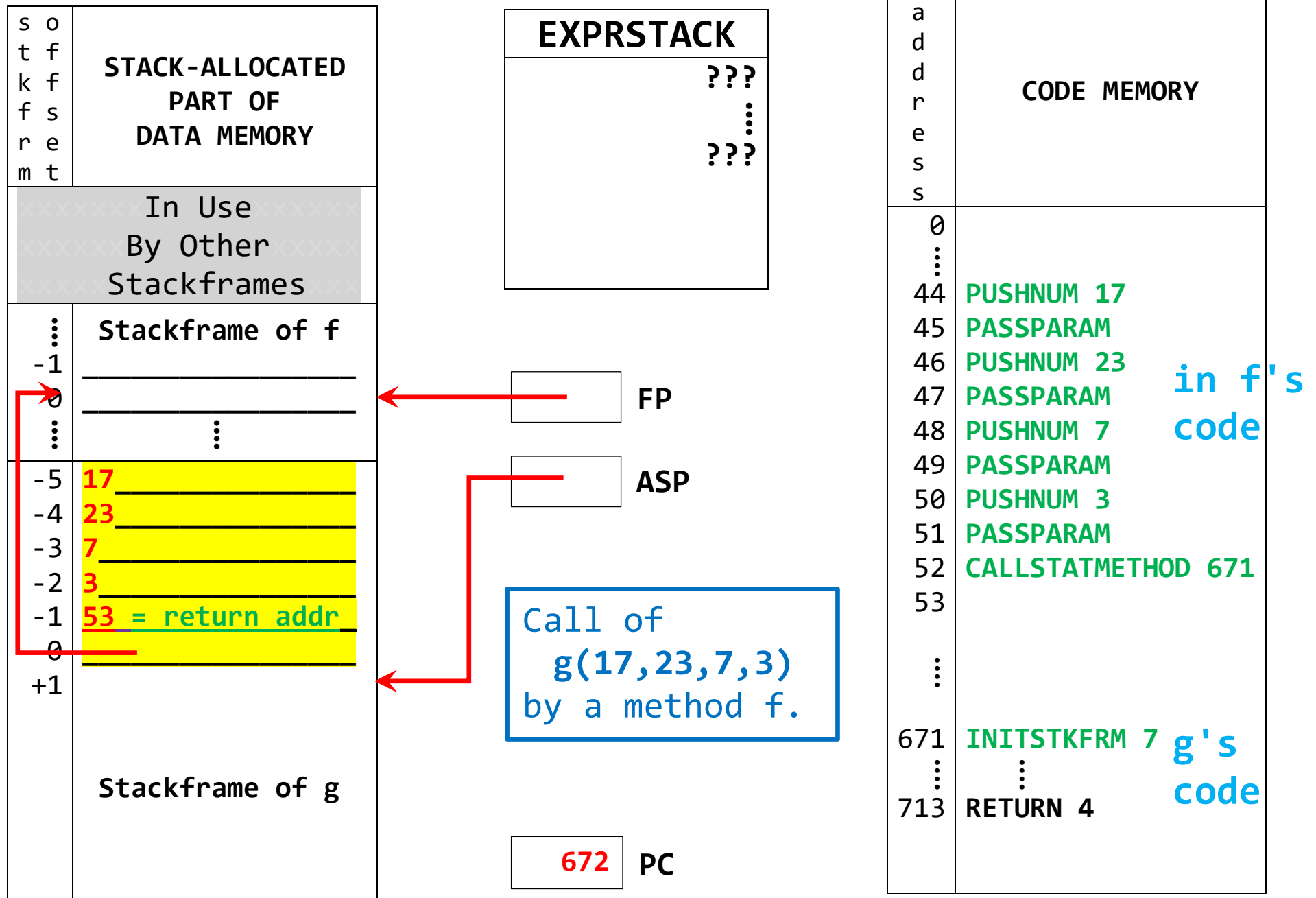
\*After the caller's FP is stored at offset 0 in the new frame, the stored pointer is called the **dynamic Link**--see page 4 of the following document (which is on Brightspace):

**Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf**

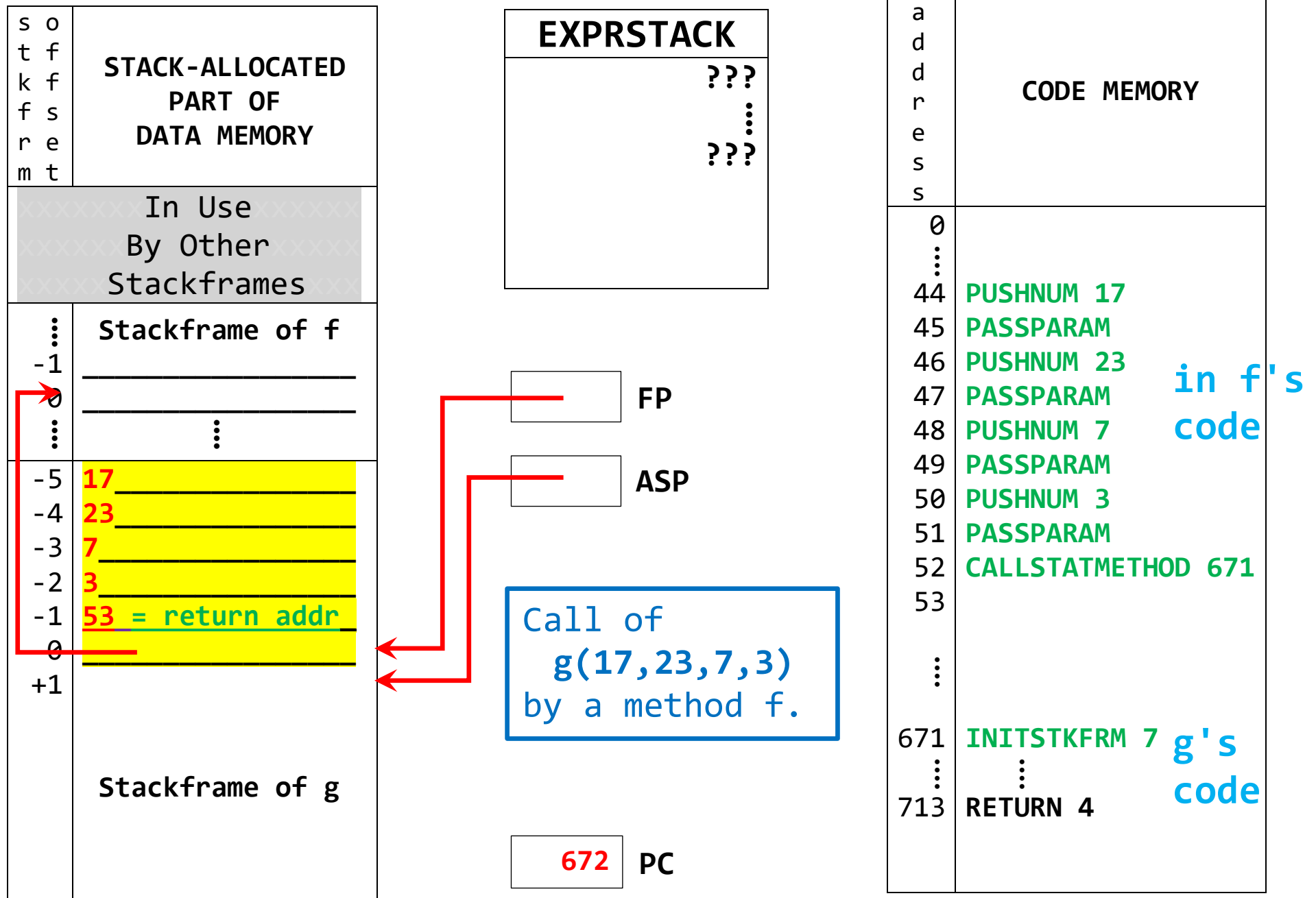
## AFTER Execution of 52: CALLSTATMETHOD 671



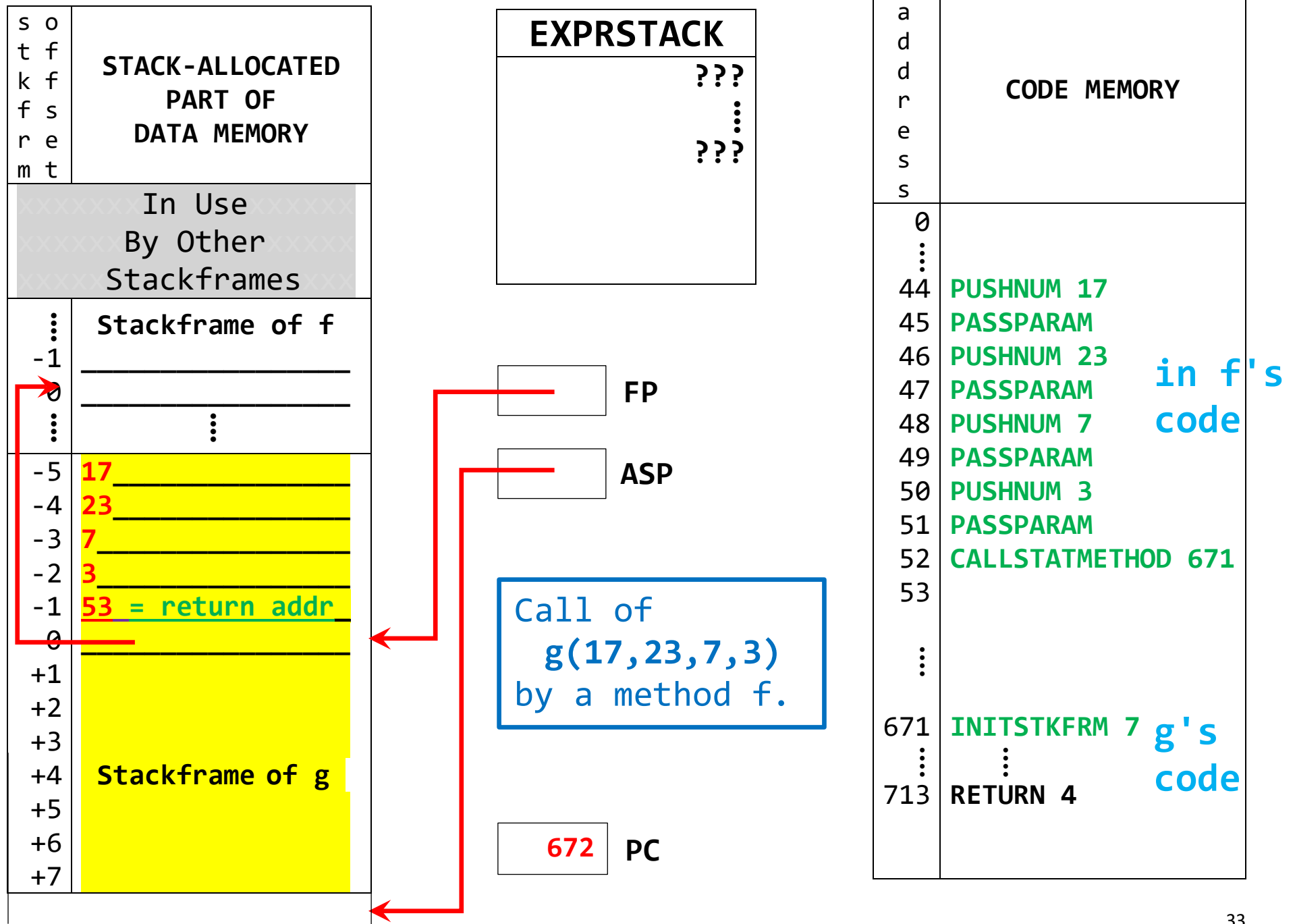
## After S-PUSH FP during Execution of 671: INITSTKFRM 7



## After Step 2 of Execution of 671: INITSTKFRM 7



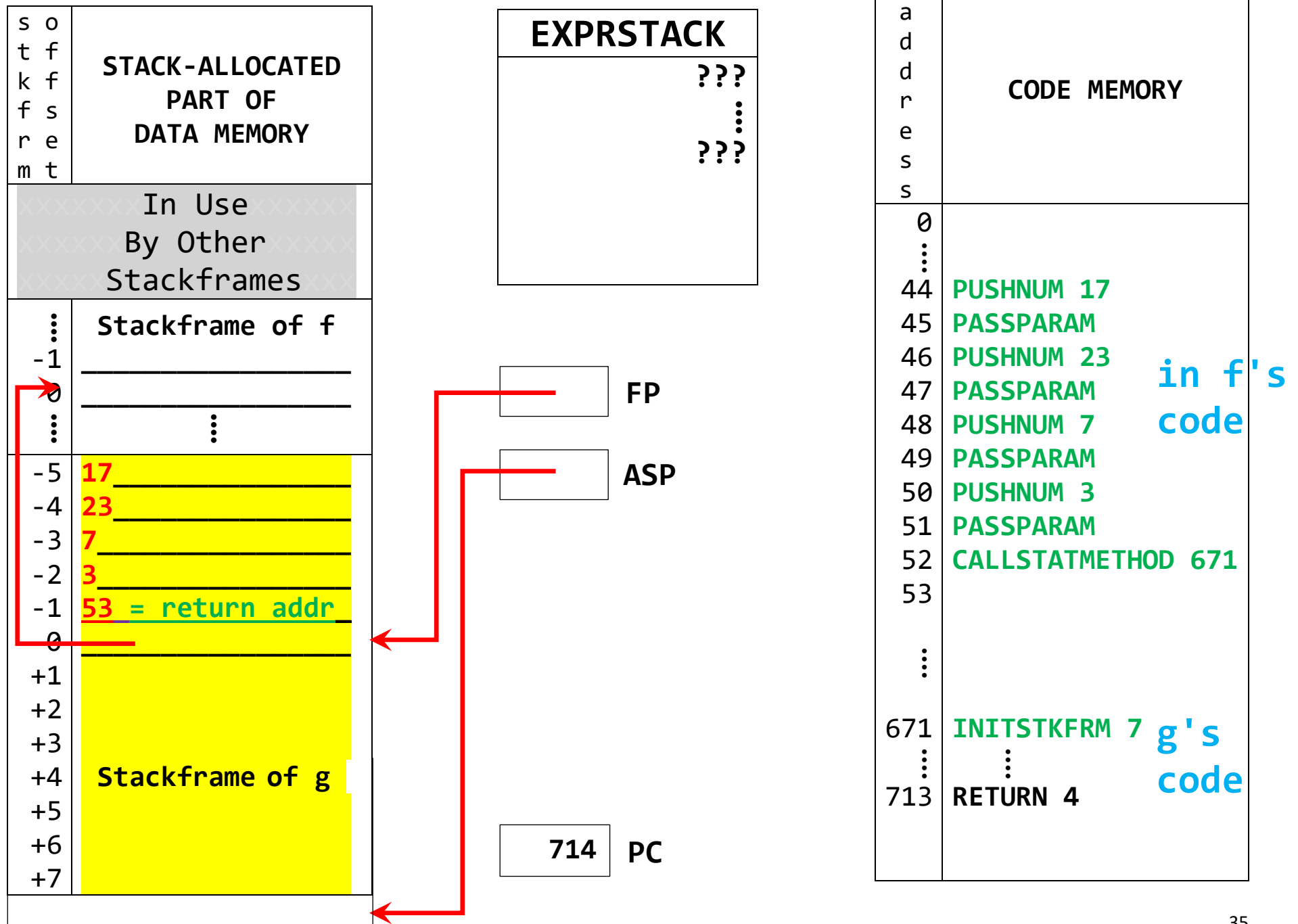
## After Final Step of Execution of 671: INITSTKFRM 7



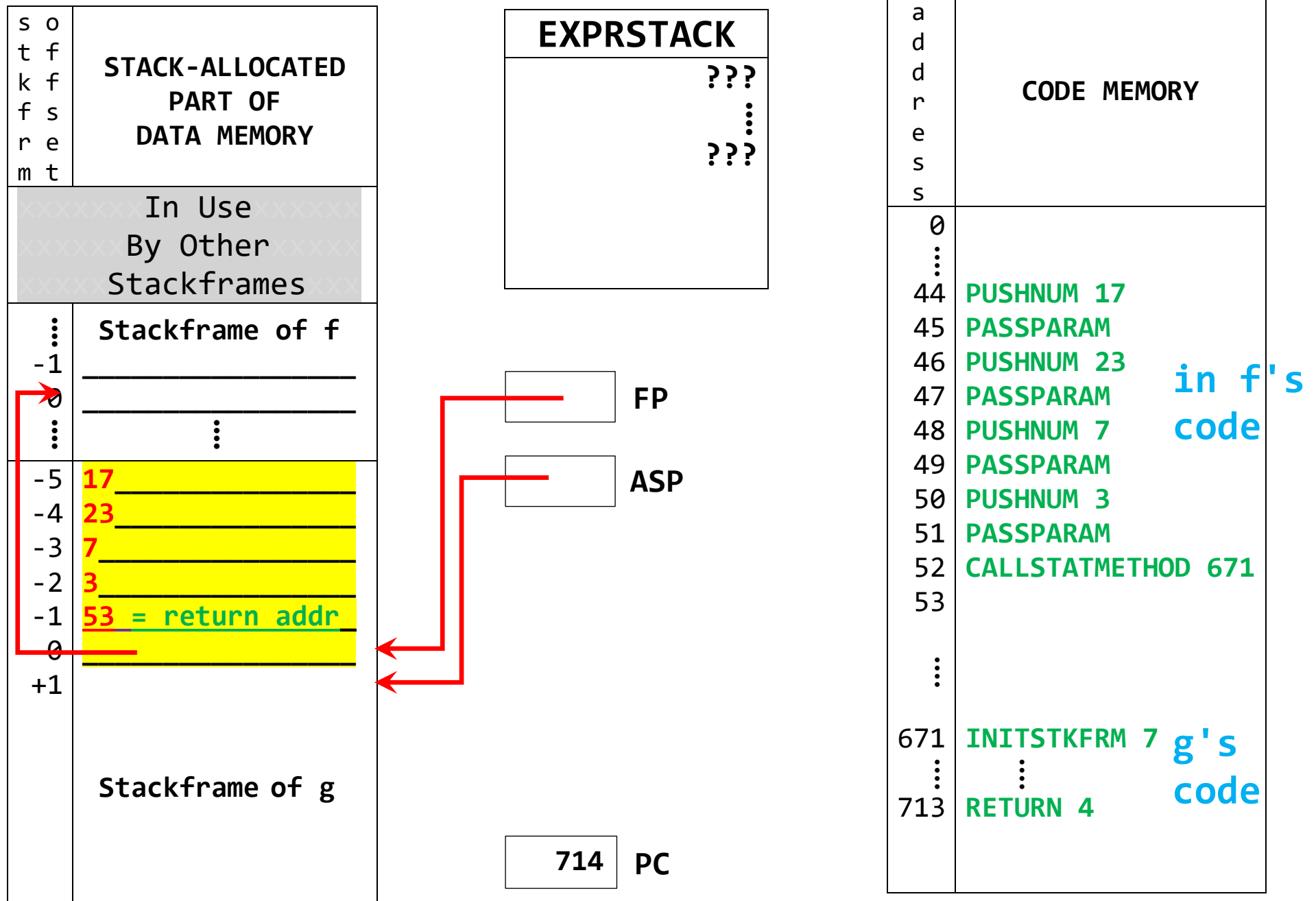
## 713: RETURN 4

should    **set ASP to FP+1**    [deallocates space used by callee's variables]  
and then **S-POP FP**    [restores caller's FP]  
and then **S-POP PC**    [puts the saved return address into PC]  
and then **decrease ASP by 4** [deallocates space used by formal parameters]

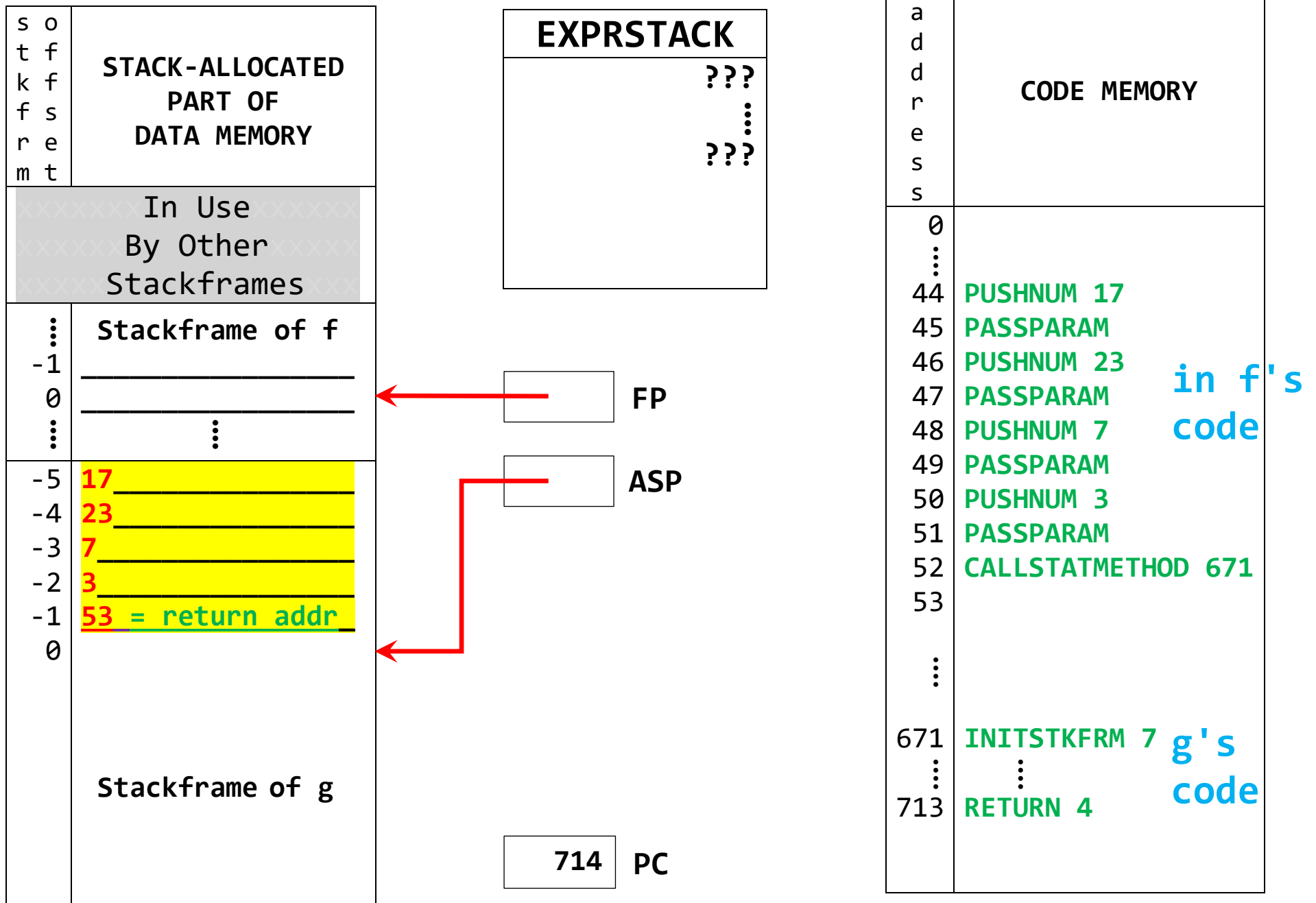
After fetch of (but BEFORE Execution of): 713: RETURN 4



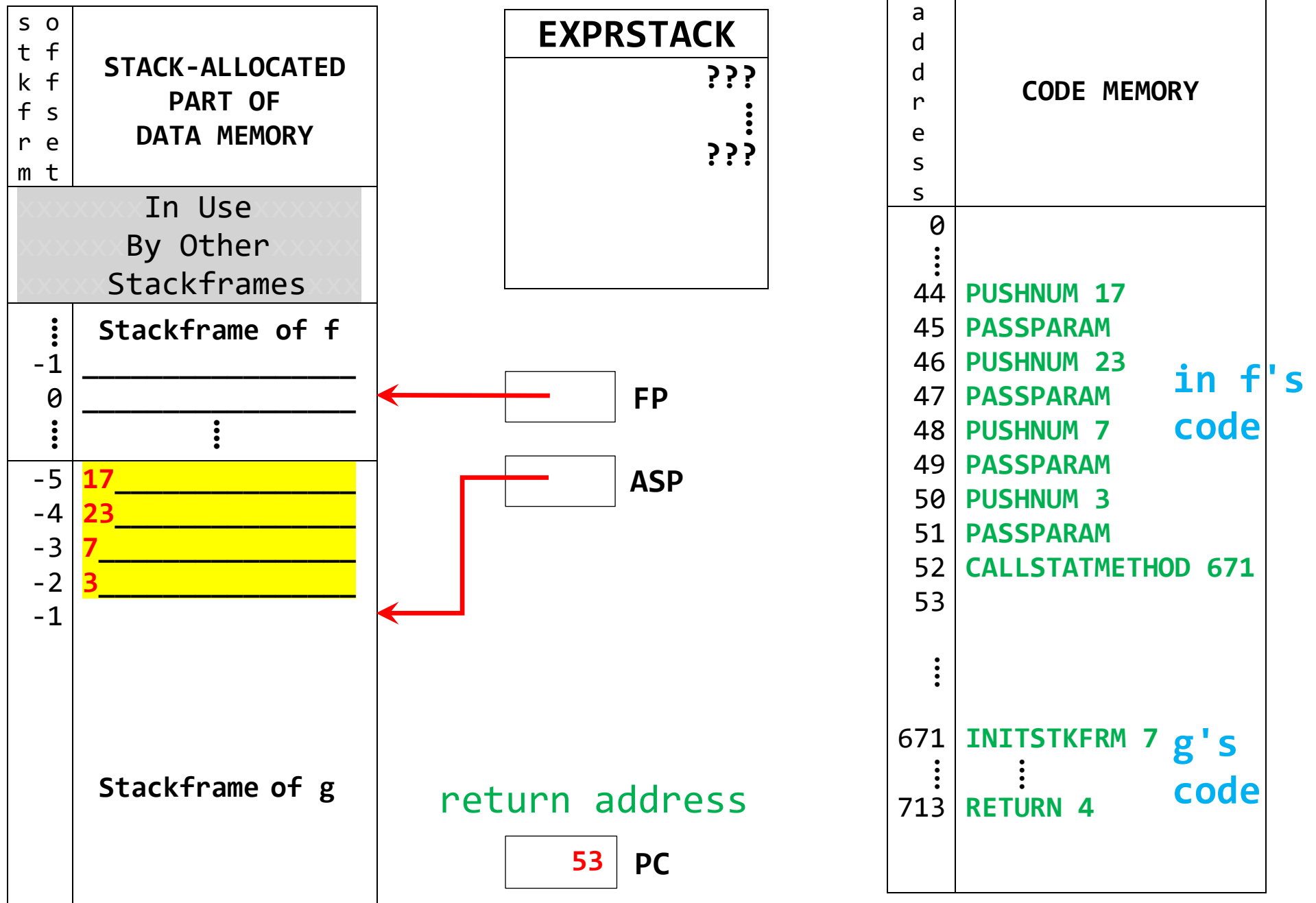
## After Step 1 of Execution of 713: RETURN 4



After S-POP FP *during Execution of* 713: RETURN 4



## After S-POP PC during of Execution of 713: RETURN 4



## After Final Step of Execution of 713: RETURN 4

