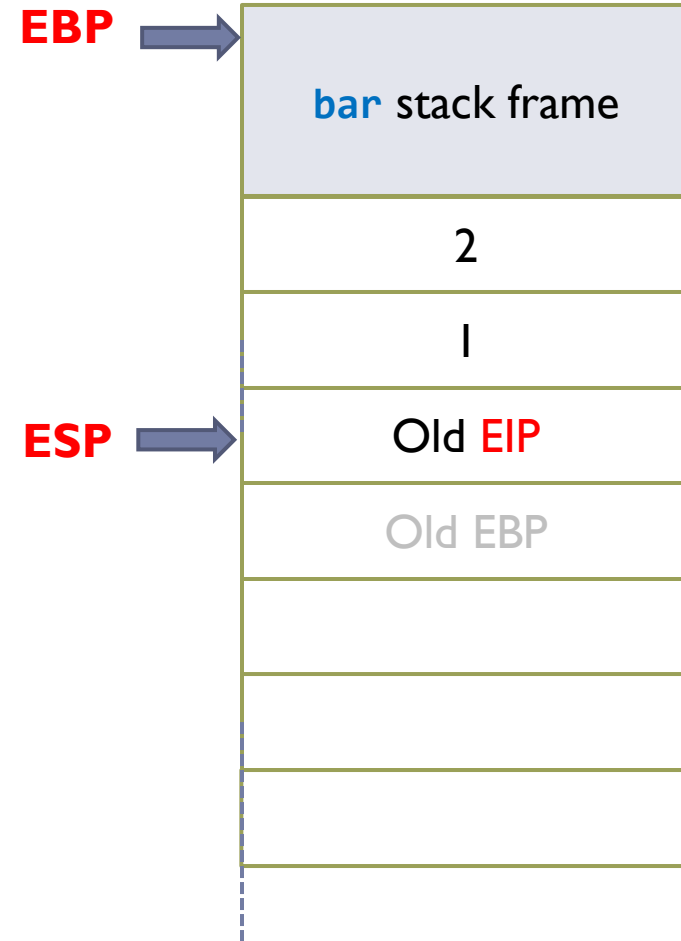


Function Call Convention

Step 8: Restore EBP.

- ▶ Pop a value from the stack (old EBP), and assign it to EBP.
- ▶ ESP is also updated (old EIP) due to the pop operation.
- ▶ (old EBP) is deleted from the stack.

```
void bar( ) {  
    foo(1, 2);  
}  
int foo(int x, int y){  
    int z = x + y;  
    return z;  
}
```

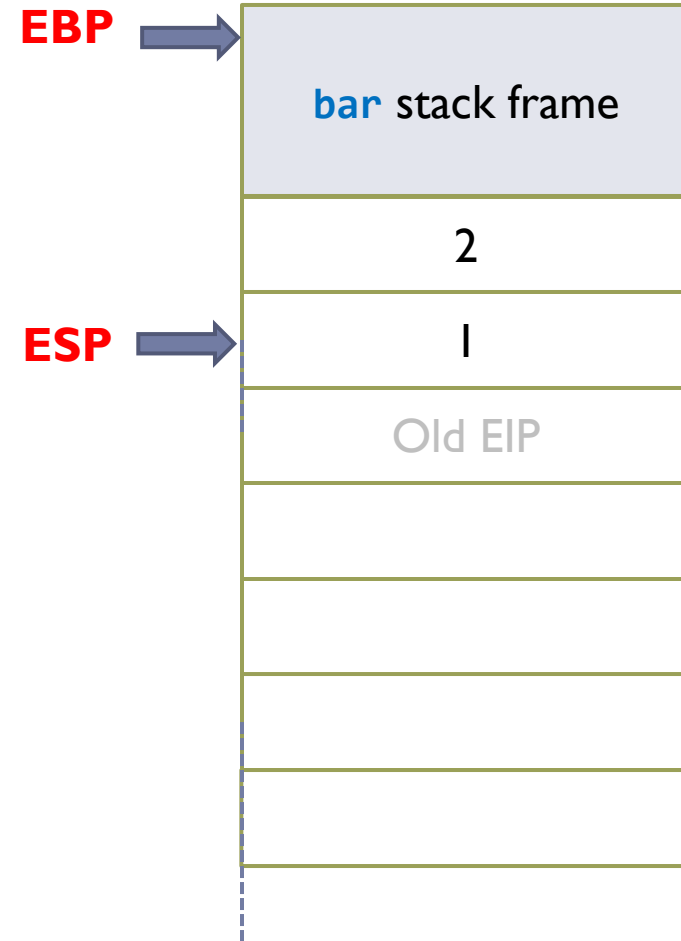


Function Call Convention

Step 9: Restore EIP.

- ▶ Pop a value from the stack (old EIP), and assign it to EIP.
- ▶ ESP is also updated (↑) due to the pop operation.
- ▶ (old EIP) is deleted from the stack.

```
void bar( ) {  
    foo(1, 2);  
}  
int foo(int x, int y){  
    int z = x + y;  
    return z;  
}
```

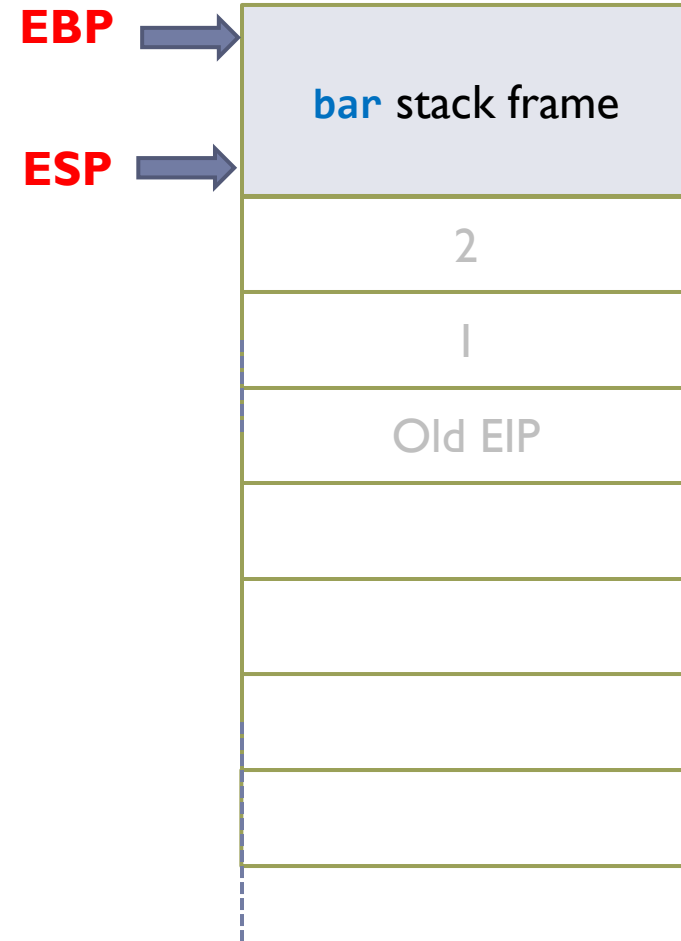


Function Call Convention

Step 10: Delete function parameters.

- ▶ Pop values from the stack (1, 2).
- ▶ **ESP** is also updated (old **ESP**) due to the pop operation.
- ▶ Function parameters (1, 2) are deleted from the stack.
- ▶ Continue the execution in function **bar**.

```
void bar( ) {  
    foo(1, 2);  
}  
int foo(int x, int y){  
    int z = x + y;  
    return z;  
}
```



Outline

- ▶ Review: Memory Layout and Function Call Convention
- ▶ **Buffer Overflow Vulnerability**

A Common Vulnerability in C Language

String

- ▶ An array of characters (1 Byte).
- ▶ Must end with NULL (or `'\0'`). A string of length n can hold only $n-1$ characters, while the last character is reserved for NULL.

`char* strcpy (char* dest, char* src)`

- ▶ Copy string `src` to `dest`
- ▶ No checks on the length of the destination string.

What if the source string is larger than destination string?

```
char str[6] = "Hello";
```

	X
	X
[5]	\0
[4]	o
[3]	l
[2]	l
[1]	e
[0]	H

```
char* strcpy (char* dest, const char* src) {  
    unsigned i;  
    for (i=0; src[i] != '\0'; ++i)  
        dest[i] = src[i];  
    dest[i] = '\0';  
    return dest;  
}
```