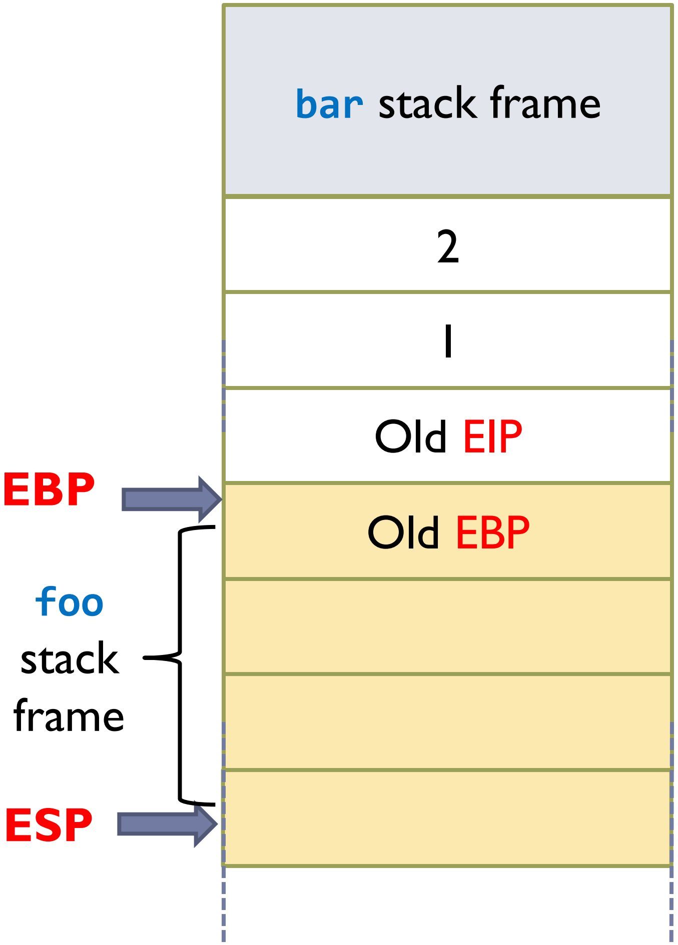
Function Call Convention

Step 5: Adjust ESP for function foo stack frame.

- Move ESP to some location below to create a new stack frame for function foo
- The stack space for function foo is pre-calculated based on the source code. It is used for storing the local variables and intermediate results.

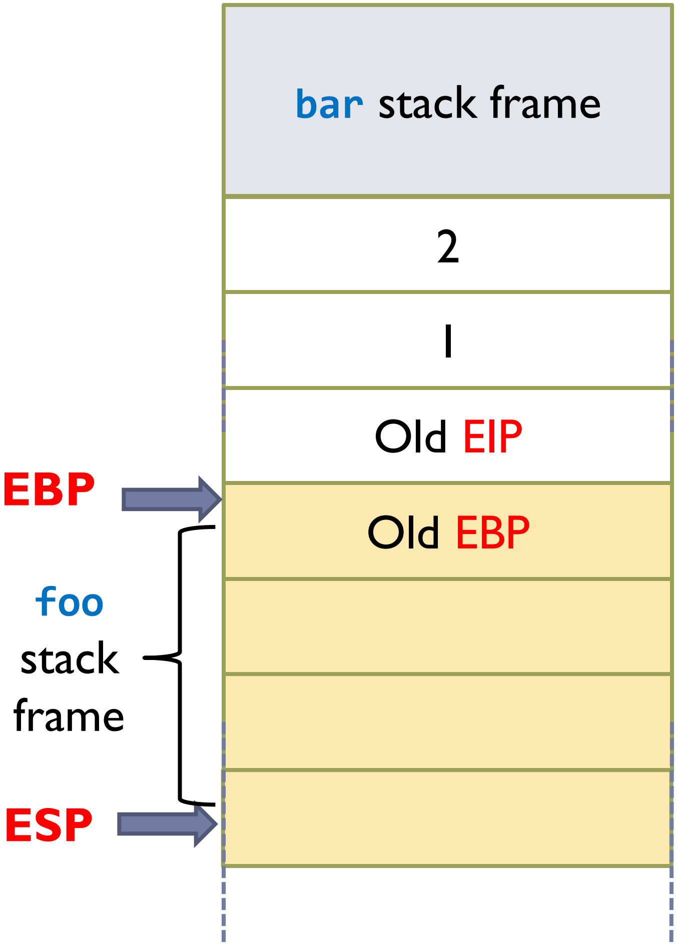
```
void bar( ) {
  foo(1, 2);
}
int foo(int x, int y){
  int z = x + y;
  return z;
}
```



Function Call Convention

Step 6: Execute function foo within its stack frame.

- The returned result will be stored in the register EAX.

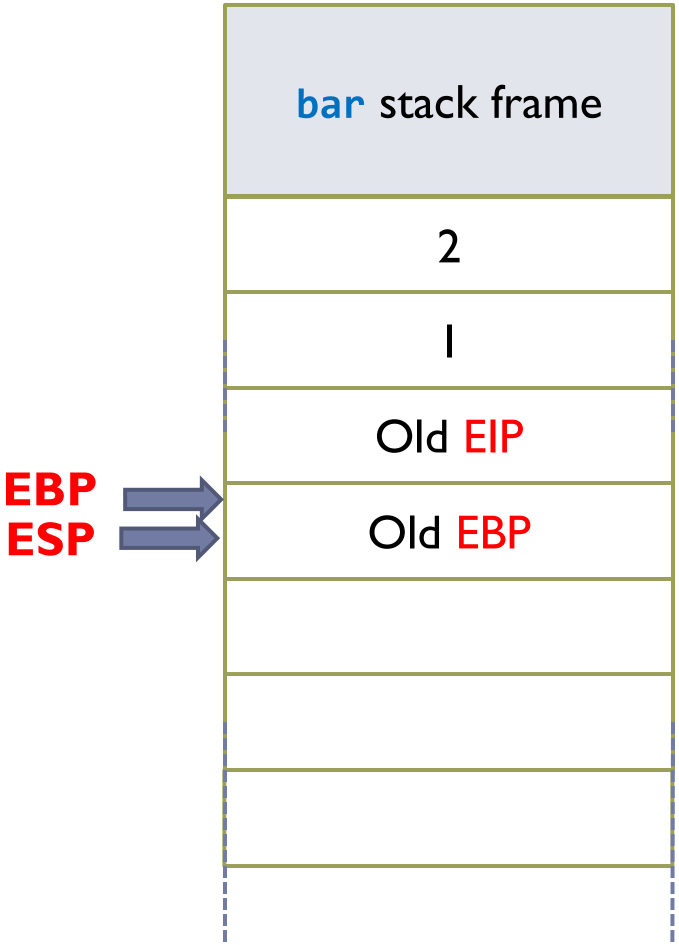


```
void bar( ) {
  foo(1, 2);
}
int foo(int x, int y){
  int z = x + y;
  return z;
}
```

Function Call Convention

Step 7: Adjust ESP.

- Move ESP to EBP
- This deletes the stack space allocated for function foo.



bar stack frame

2

1

Old EIP

Old EBP

EBP

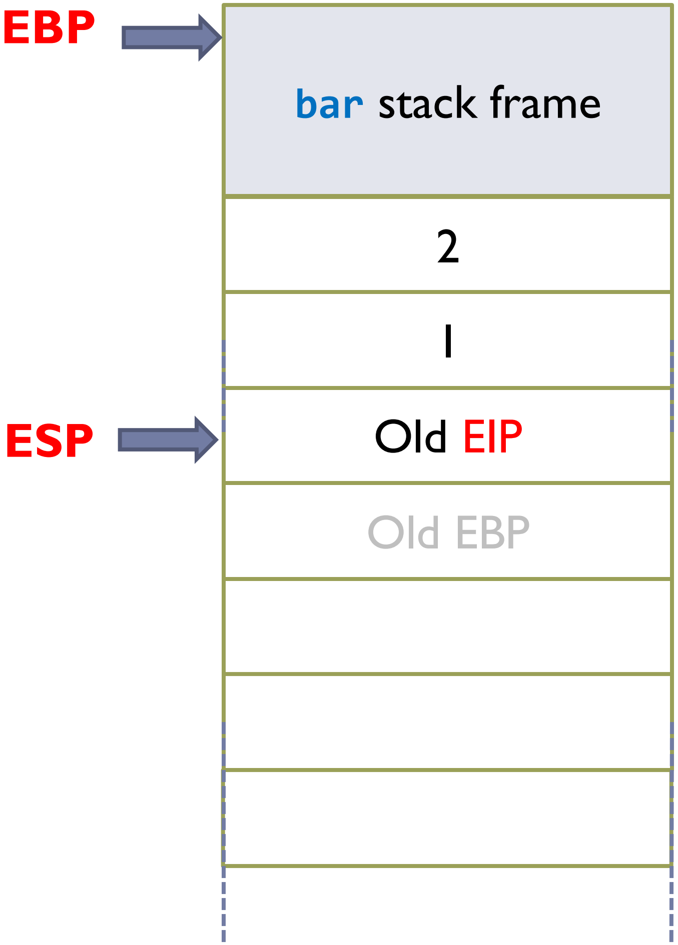
ESP

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

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.

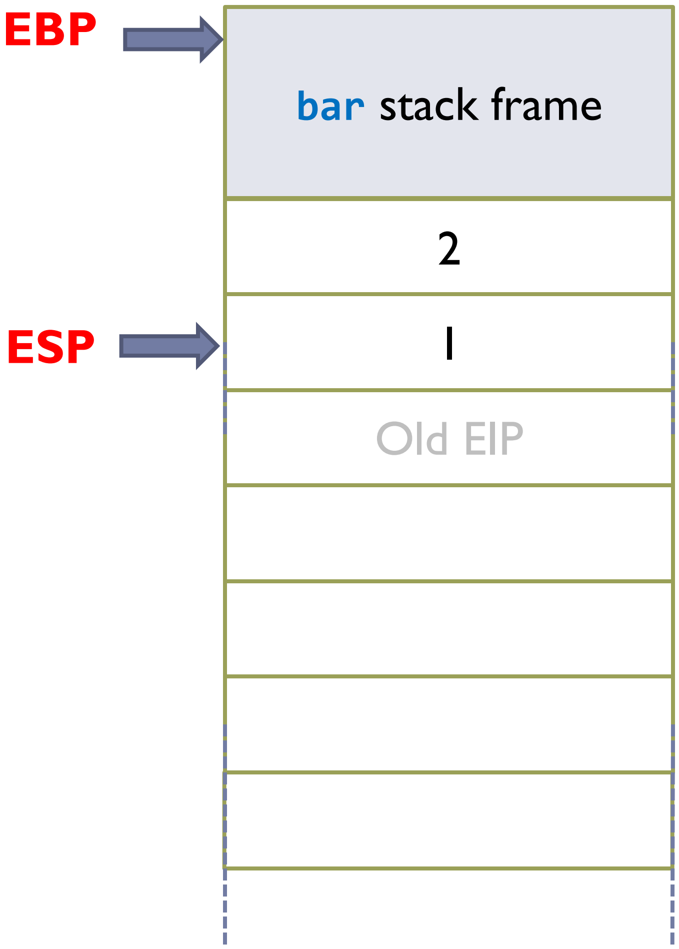


```
void bar( ) {
  foo(1, 2);
}
int foo(int x, int y){
  int z = x + y;
  return z;
}
```

Function Call Convention

Step 9: Restore EIP.

- Pop a value from the stack (old EIP), and assign it to EIP.
- ESP is also updated (1) 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;
}
```