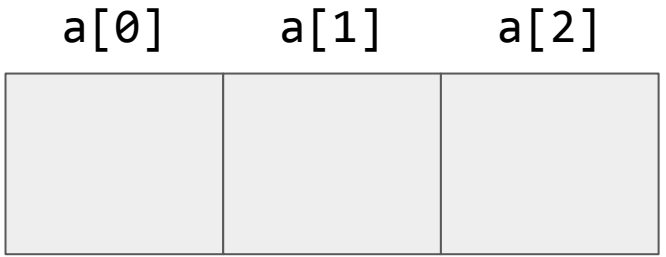
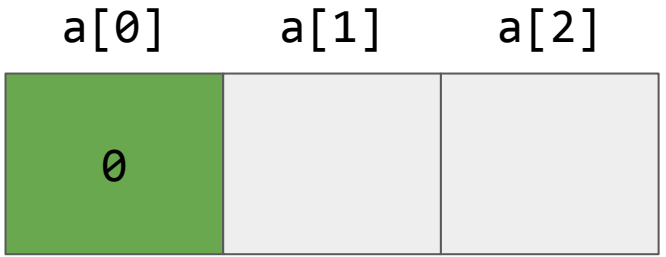
Arrays in Assembly

Adalbert **Gerald** Soosai Raj

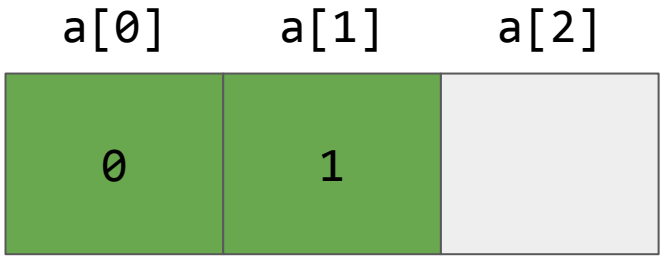
```
#define N 3

int func()
{
    int a[N];
    int i;
    for (i = 0; i < N; ++i) {
        a[i] = i;
    }
}
```



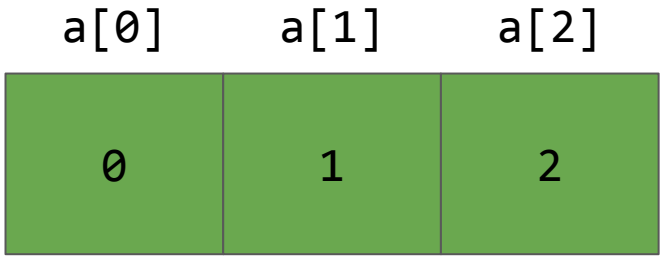
```
#define N 3

int func()
{
    int a[N];
    int i;
    for (i = 0; i < N; ++i) {
        a[i] = i;
    }
}
```



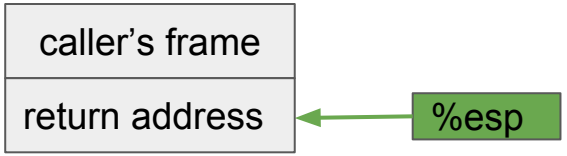
```
#define N 3

int func()
{
    int a[N];
    int i;
    for (i = 0; i < N; ++i) {
        a[i] = i;
    }
}
```



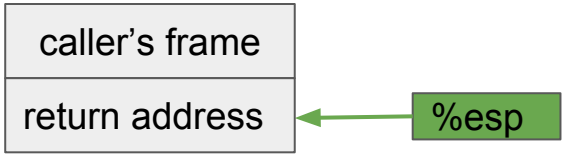
```
#define N 3

int func()
{
    int a[N];
    int i;
    for (i = 0; i < N; ++i) {
        a[i] = i;
    }
}
```



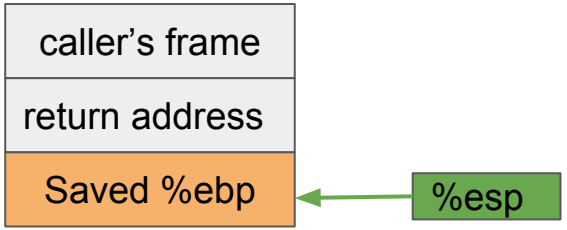
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



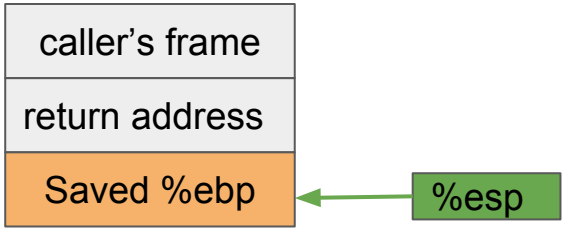
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



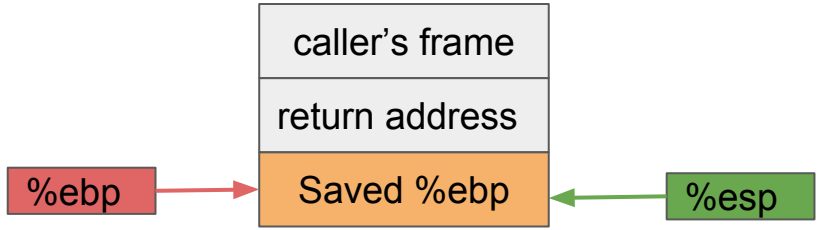
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



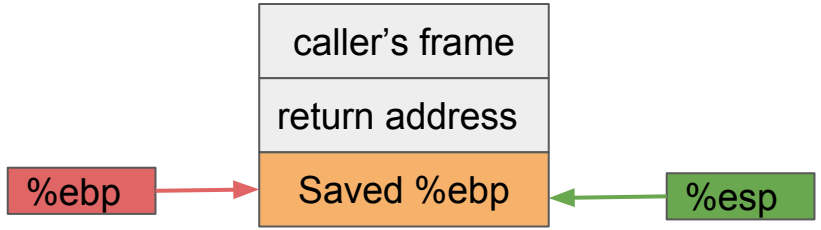
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



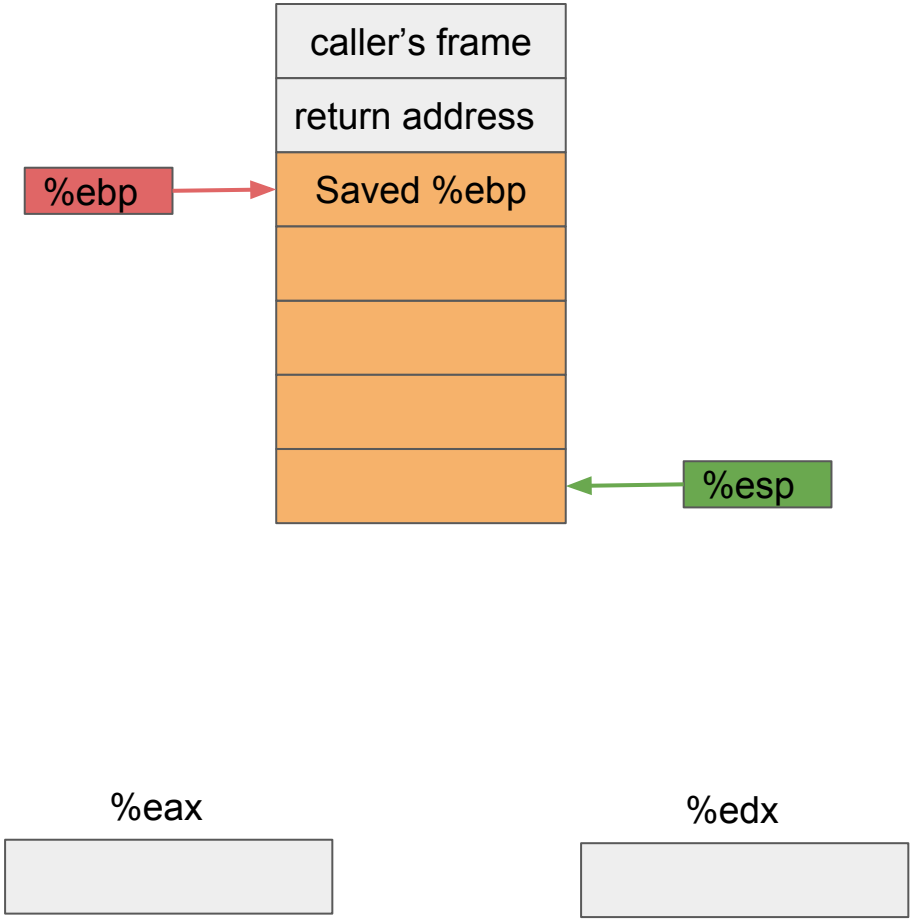
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



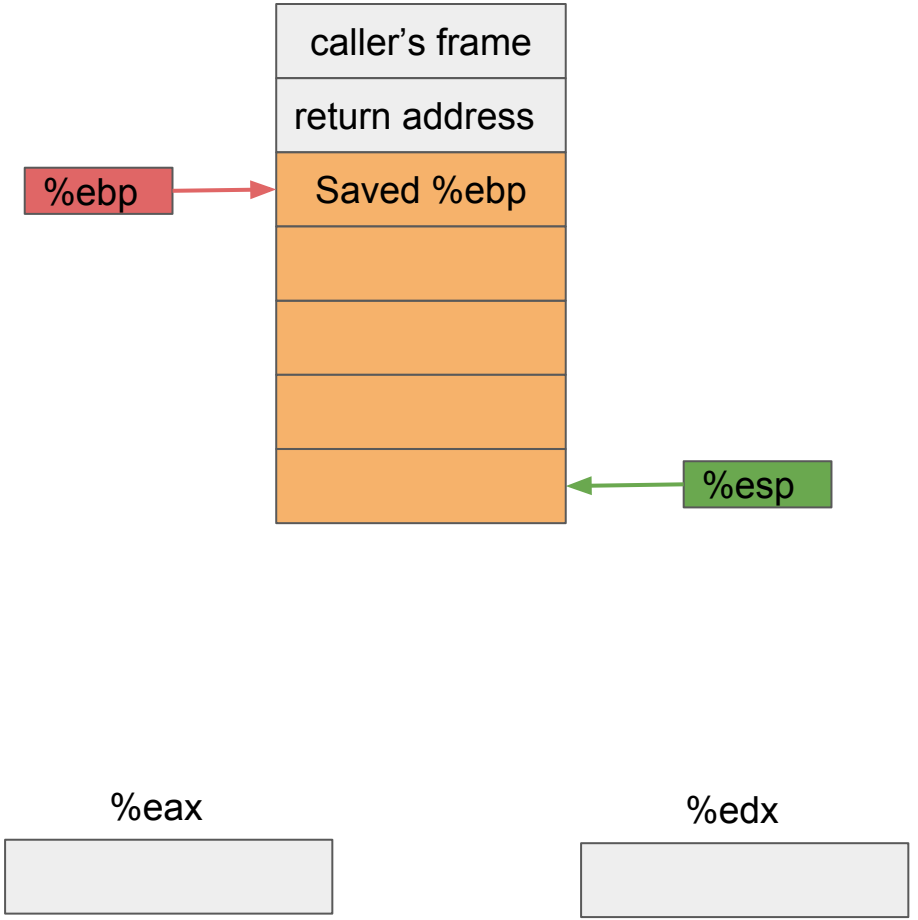
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



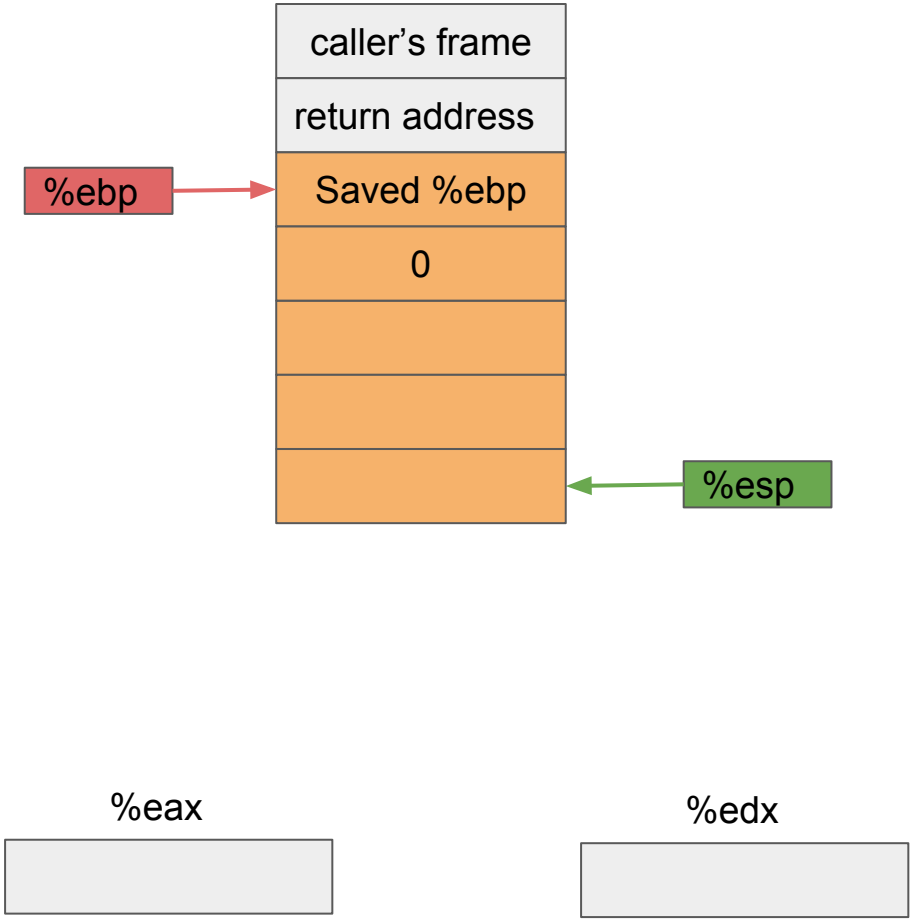
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



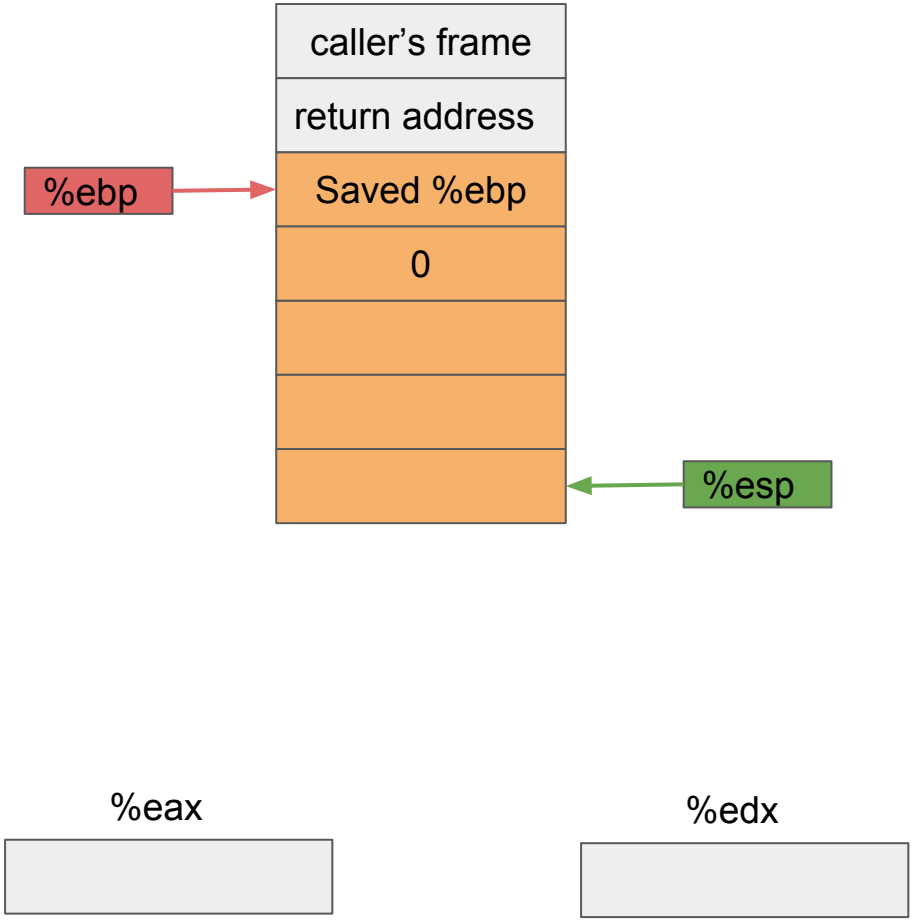
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



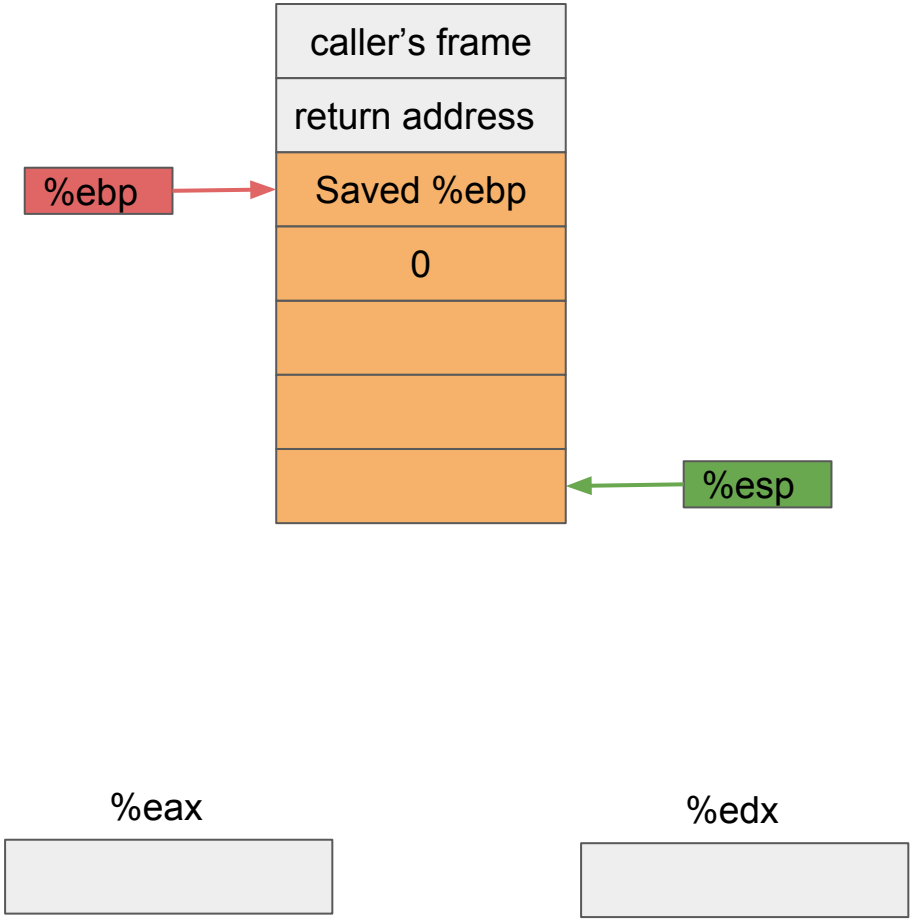
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



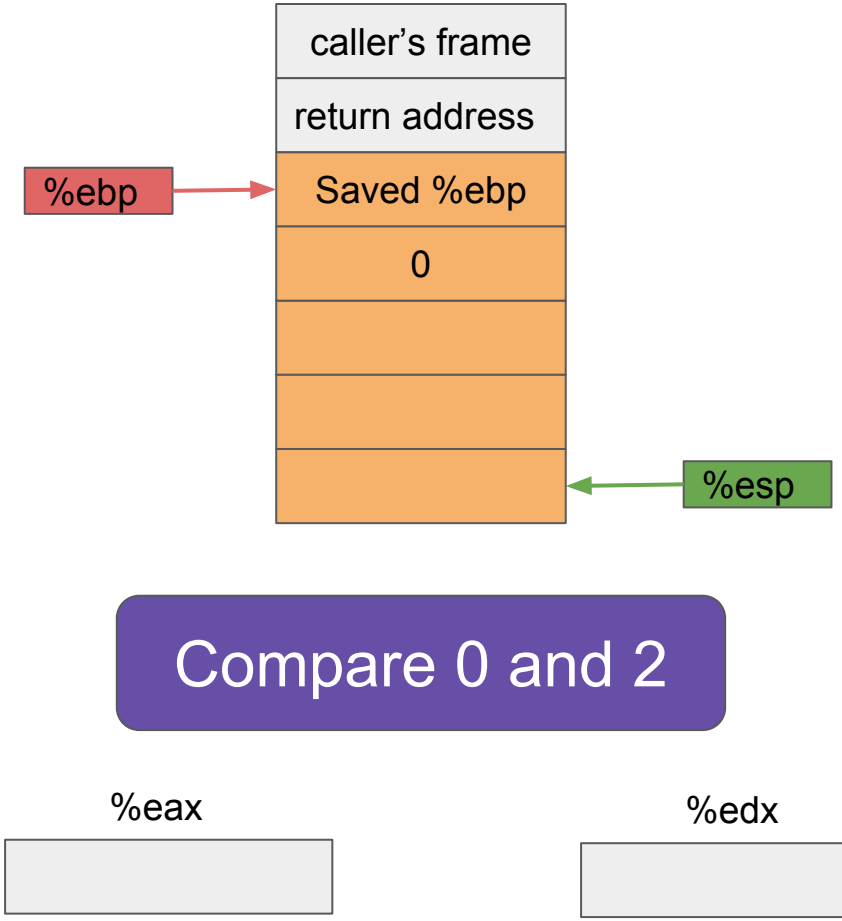
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



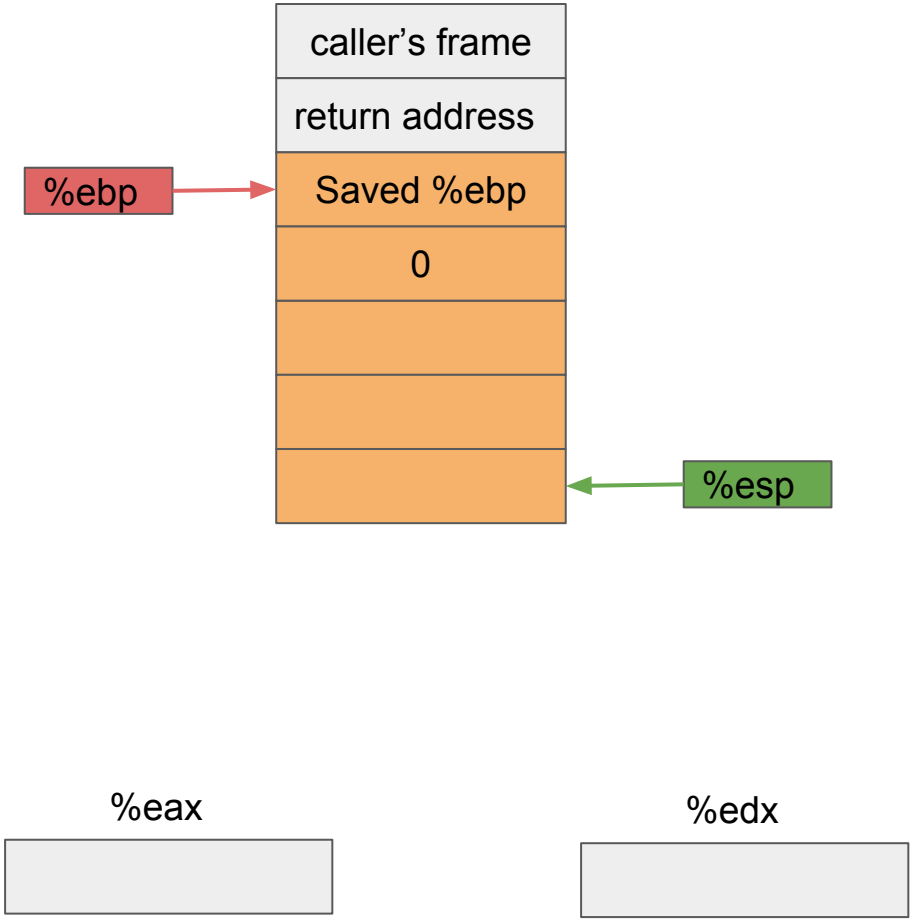
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



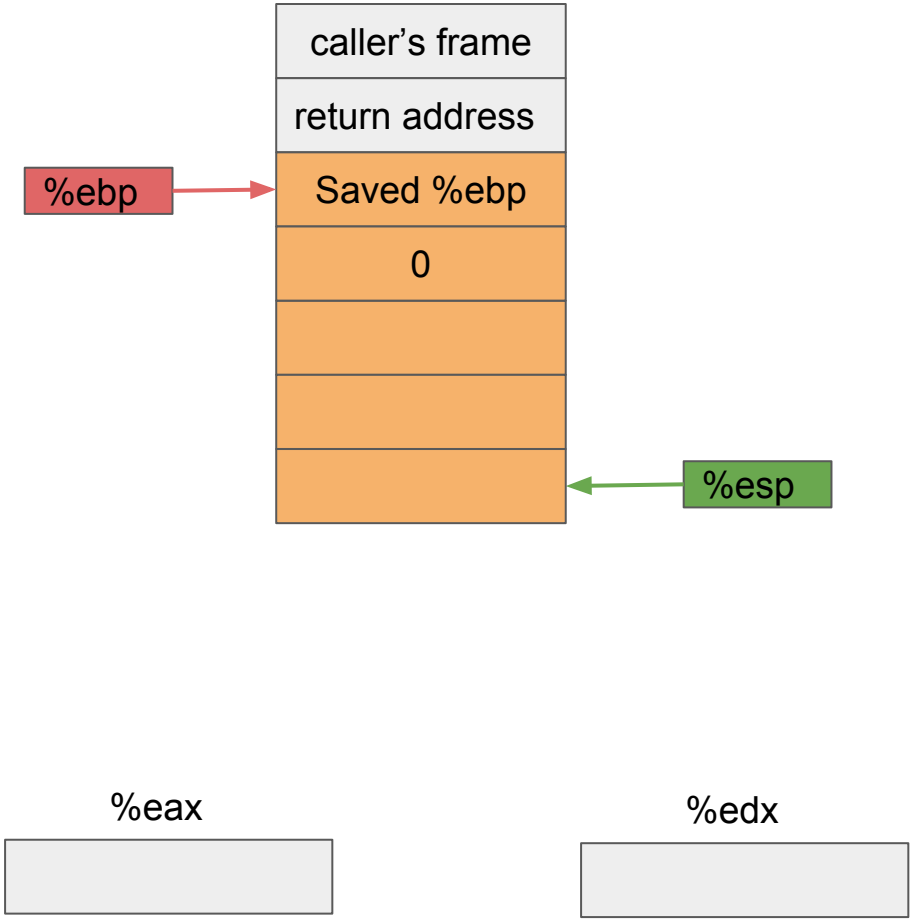
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



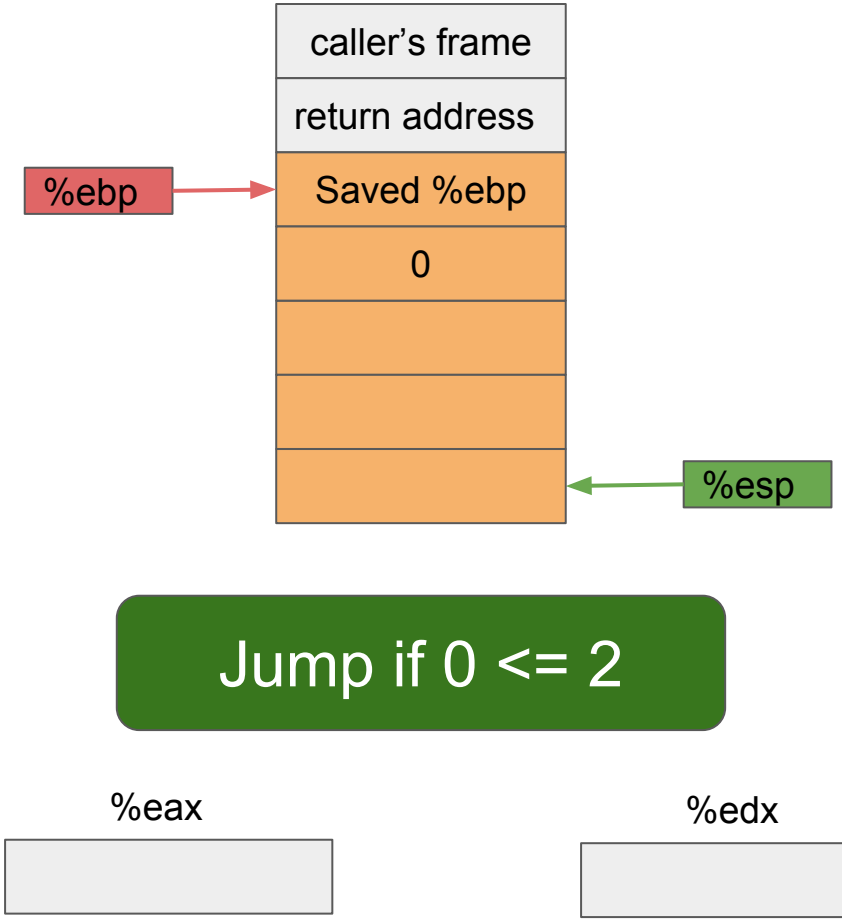
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



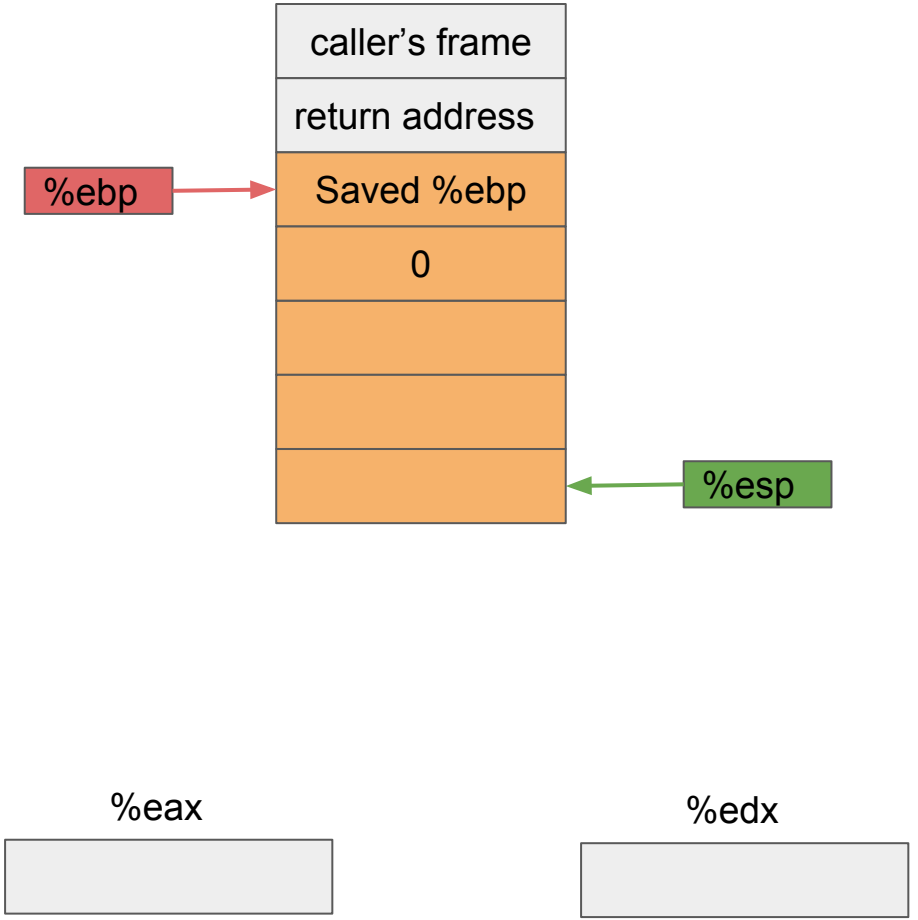
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



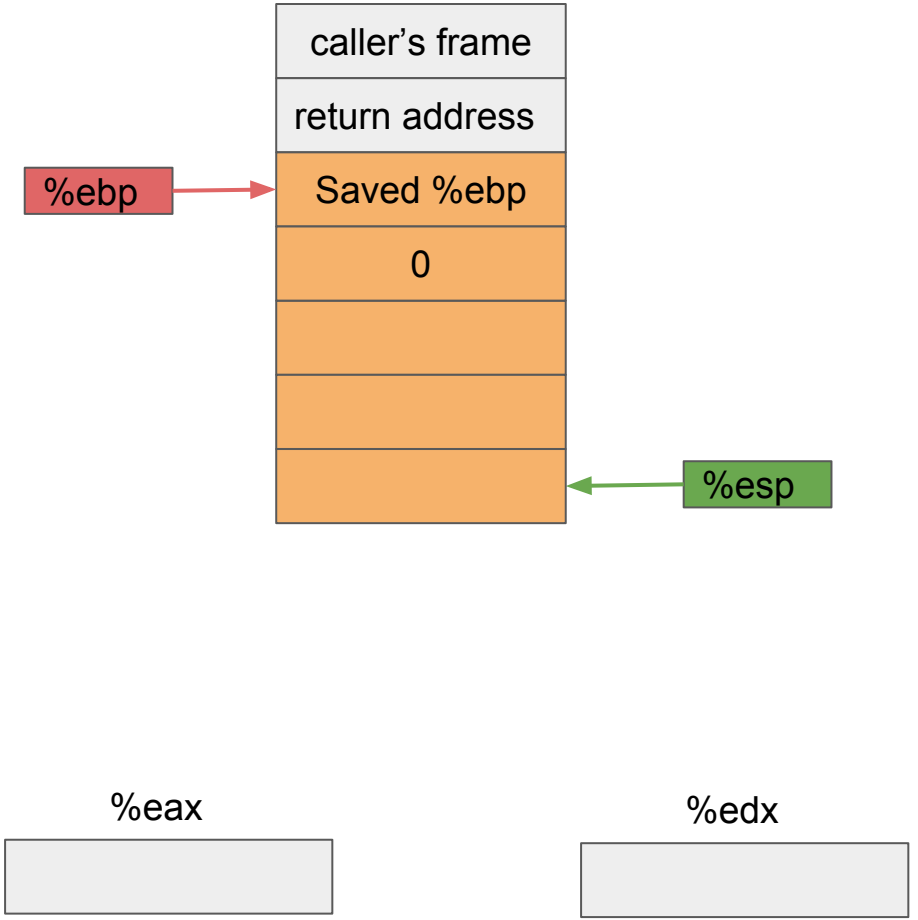
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



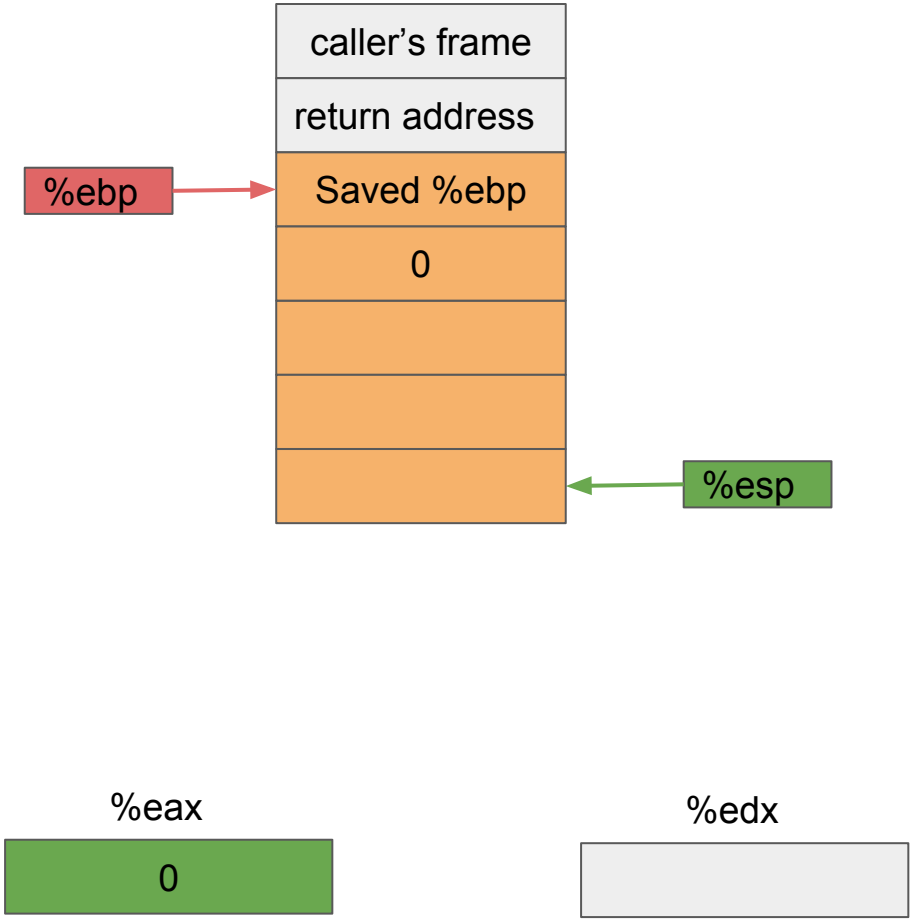
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



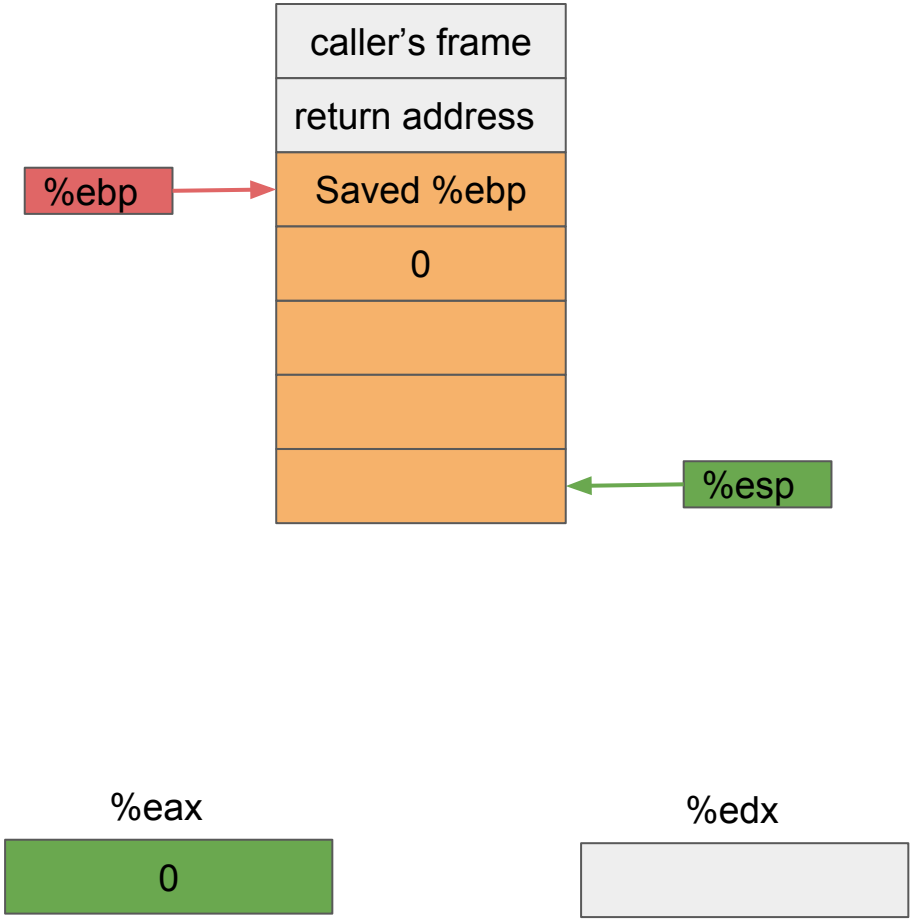
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



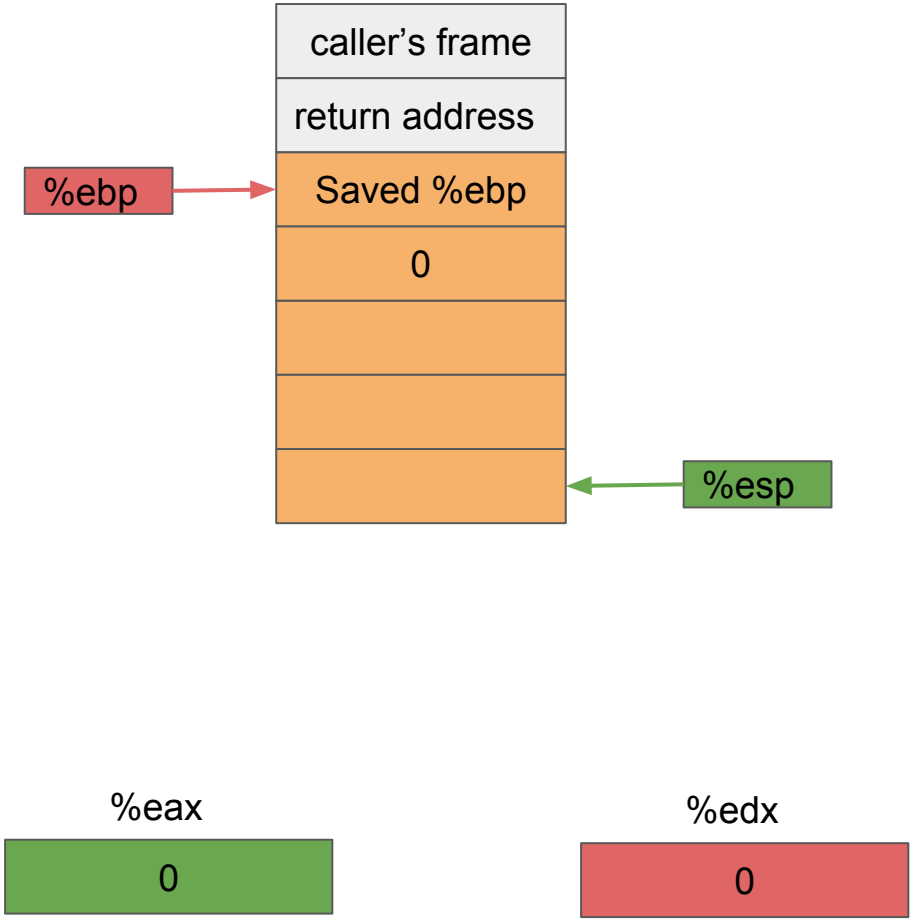
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



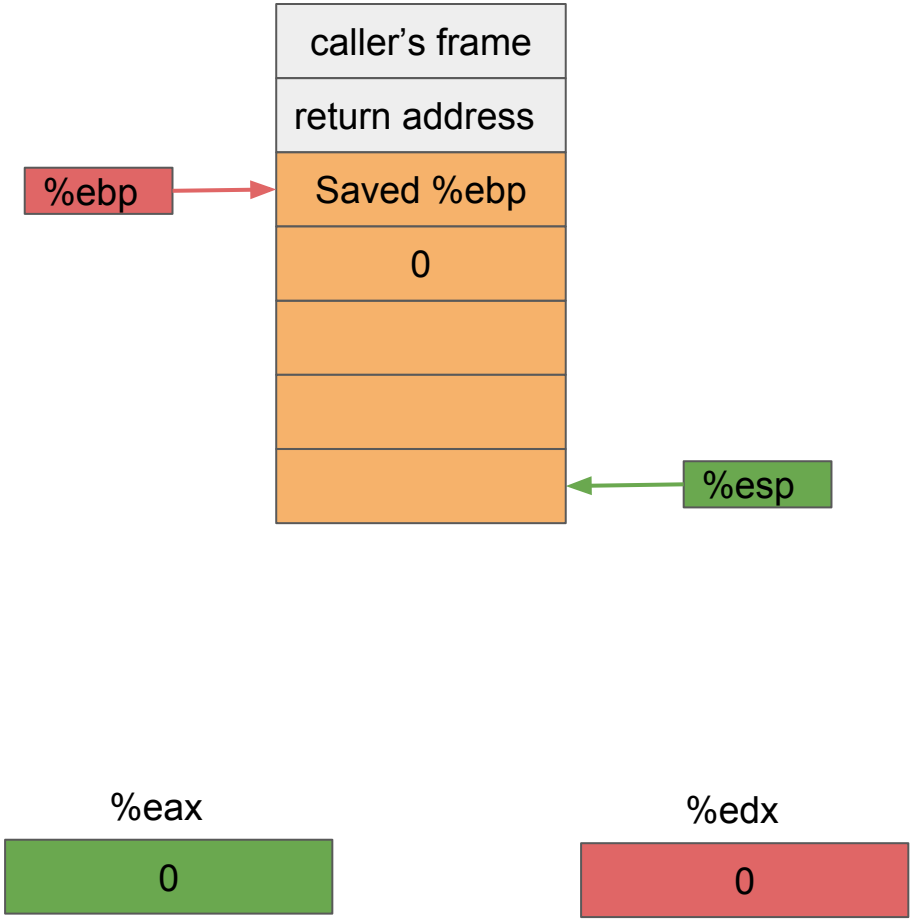
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



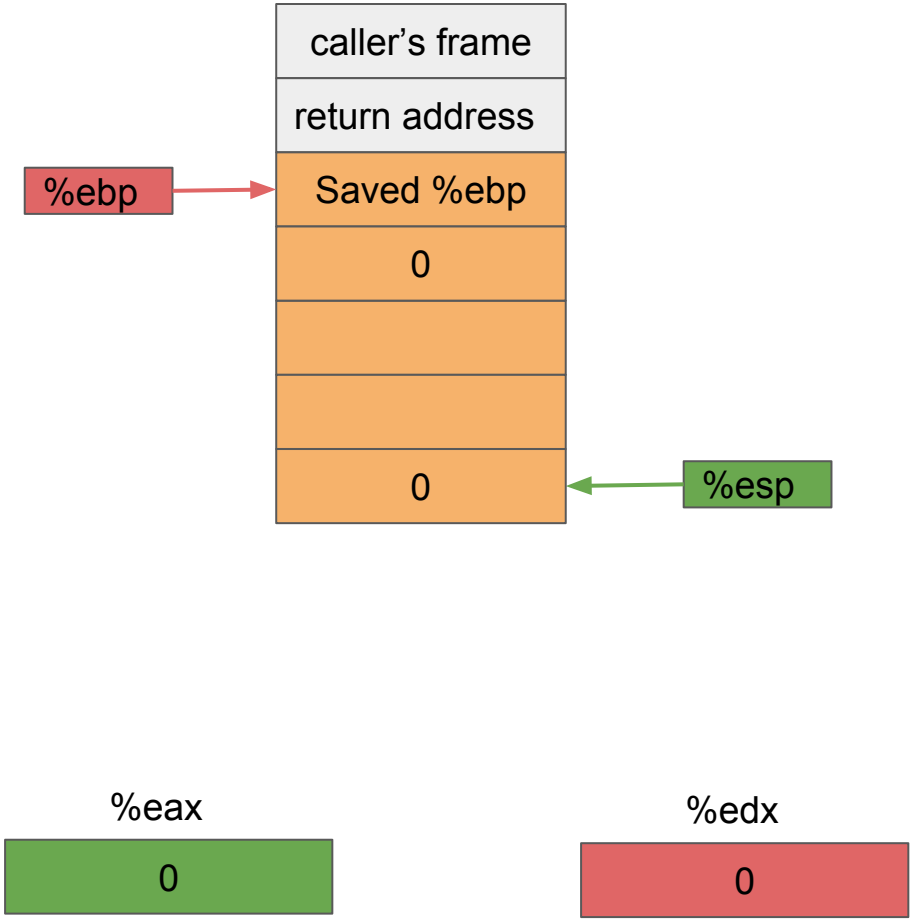
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



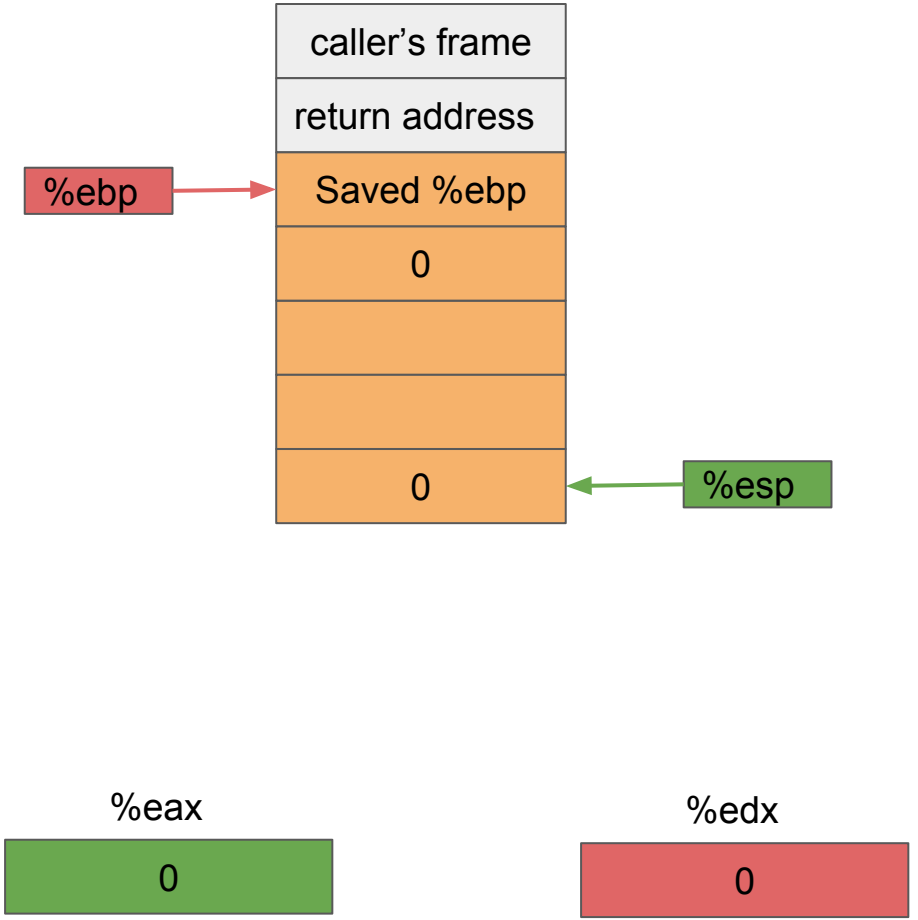
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



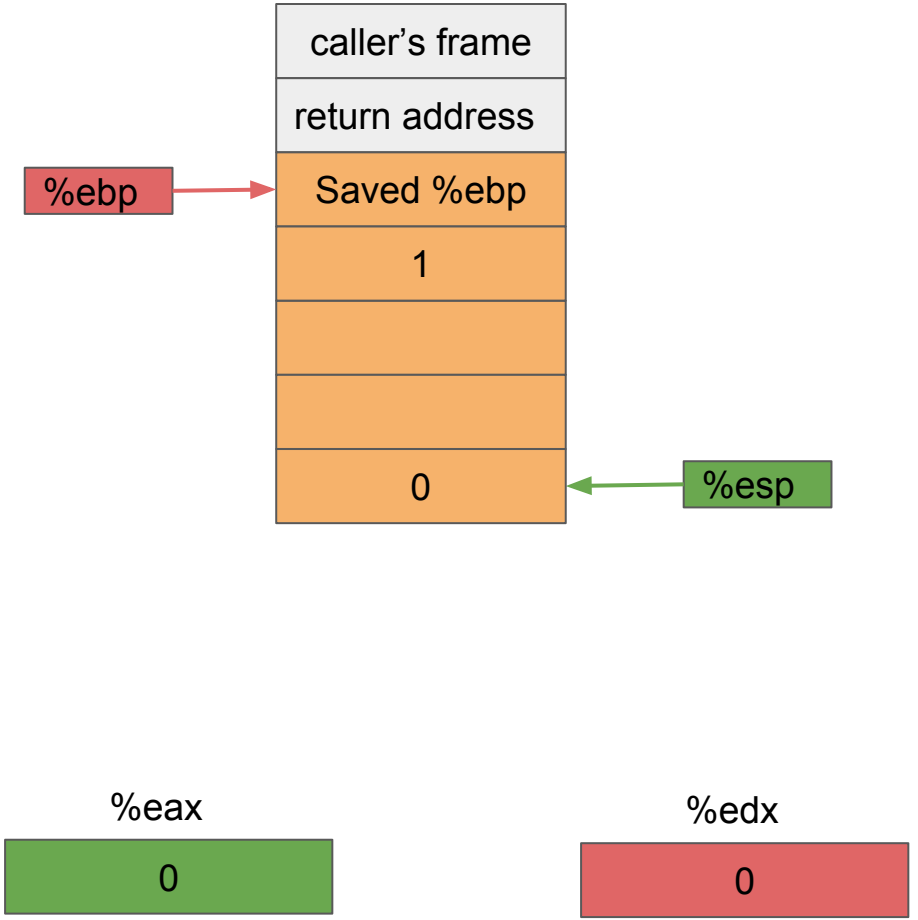
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



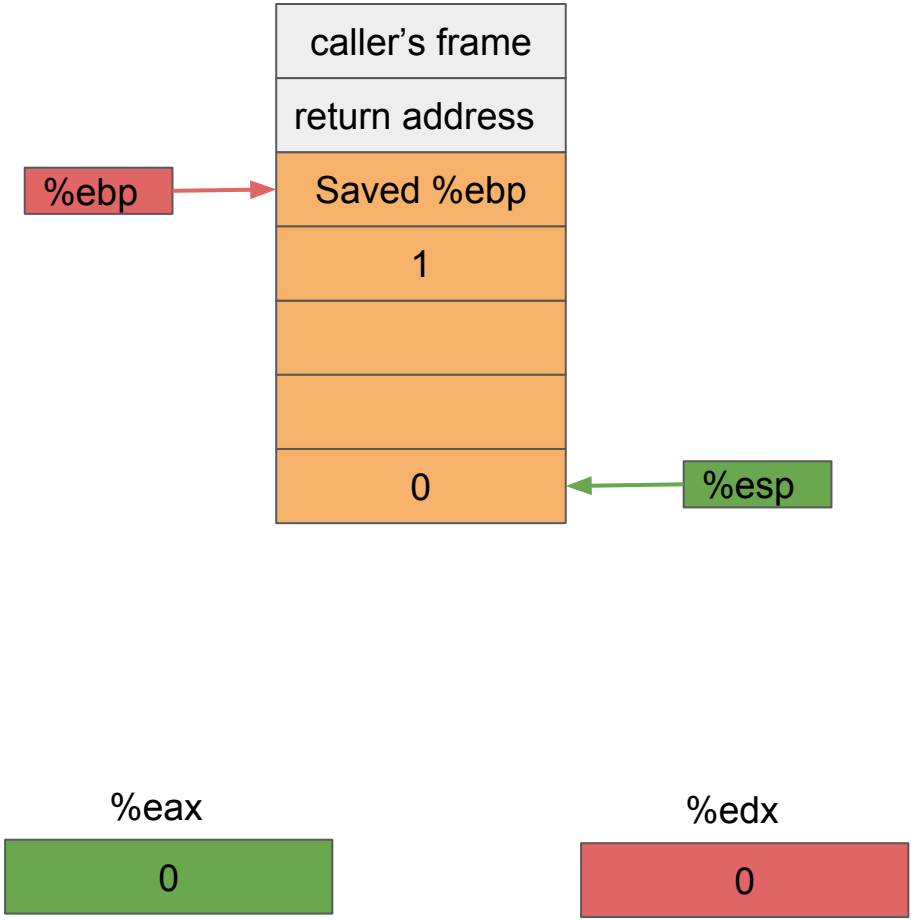
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



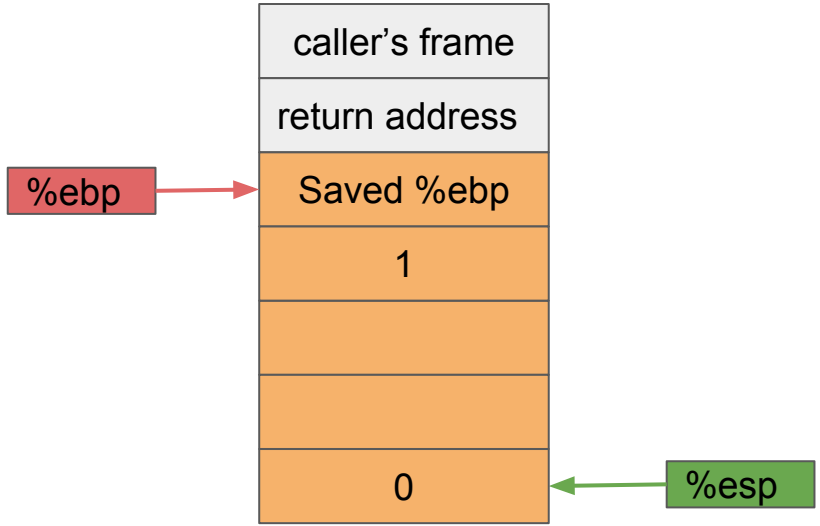
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

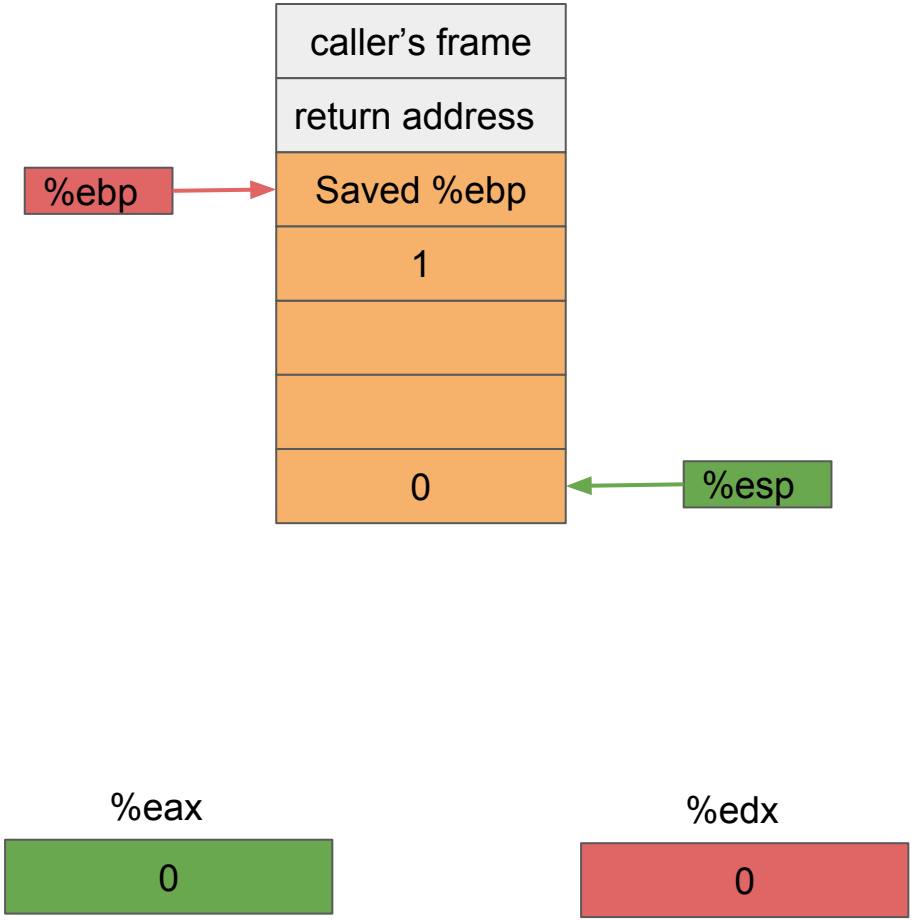


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

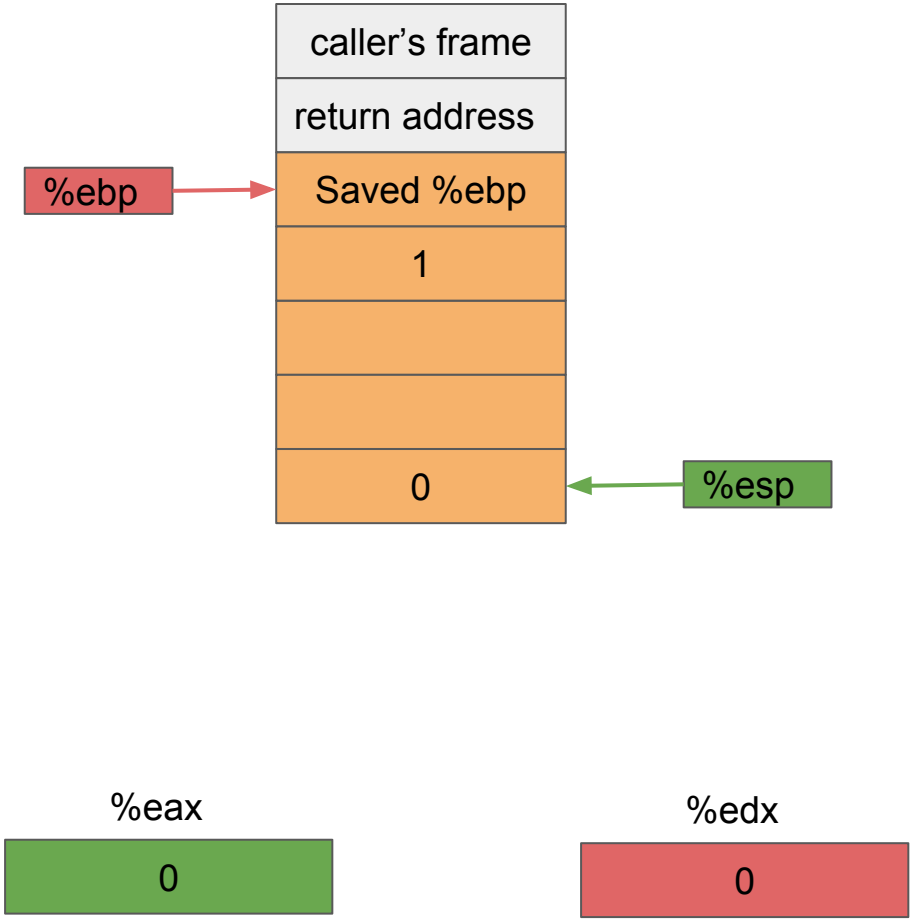


Compare 1 and 2

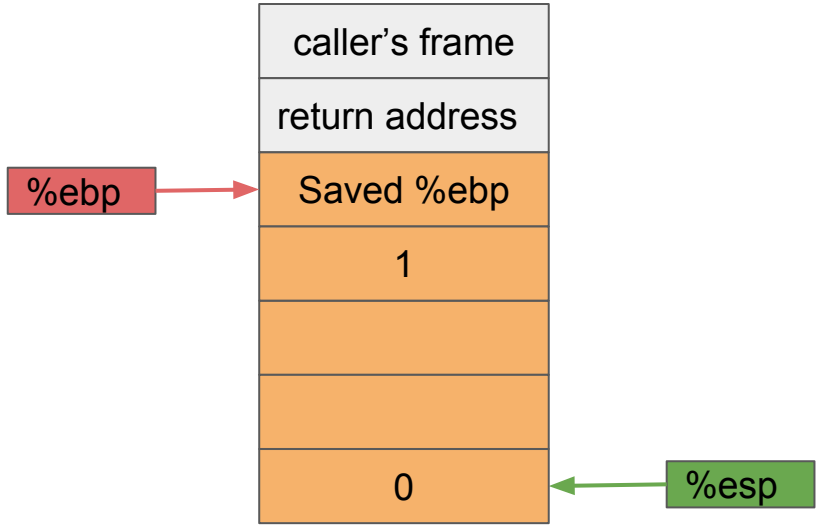
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

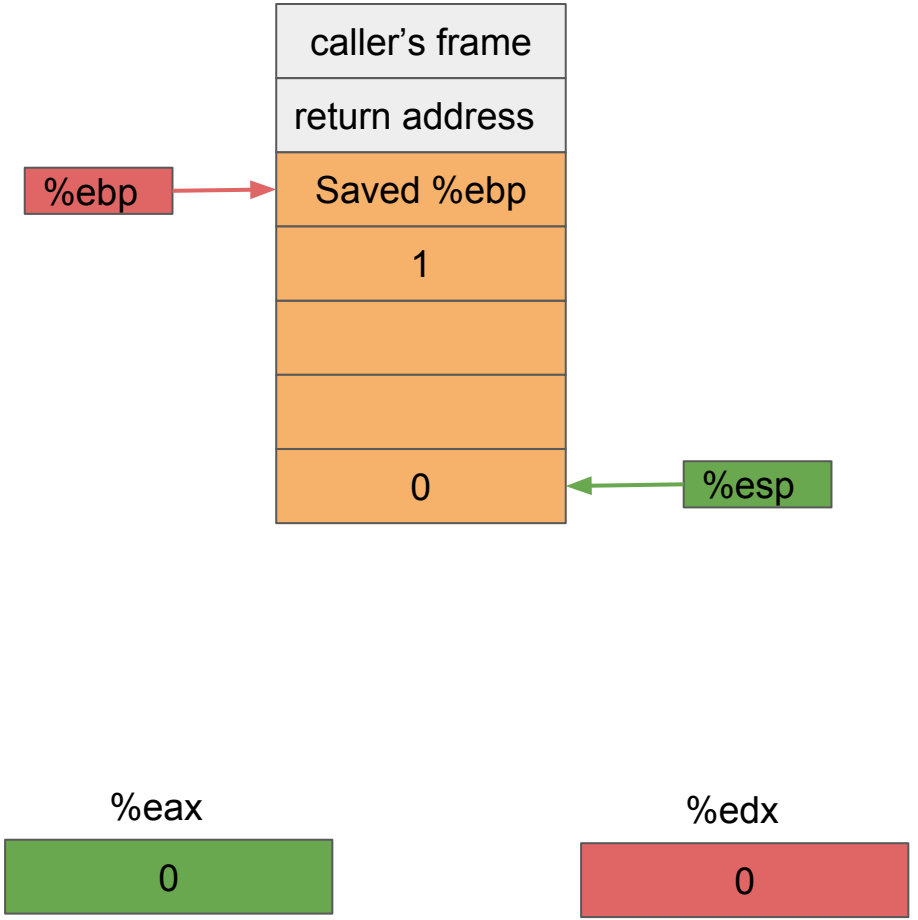


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

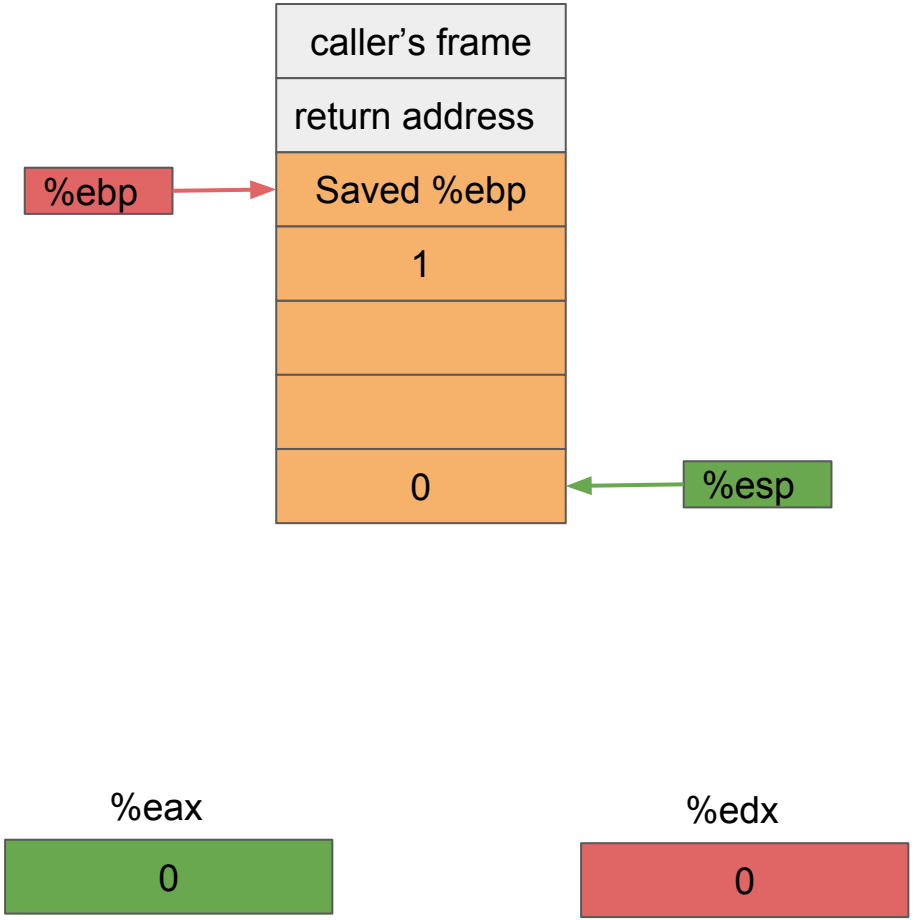


Jump if 1 <= 2

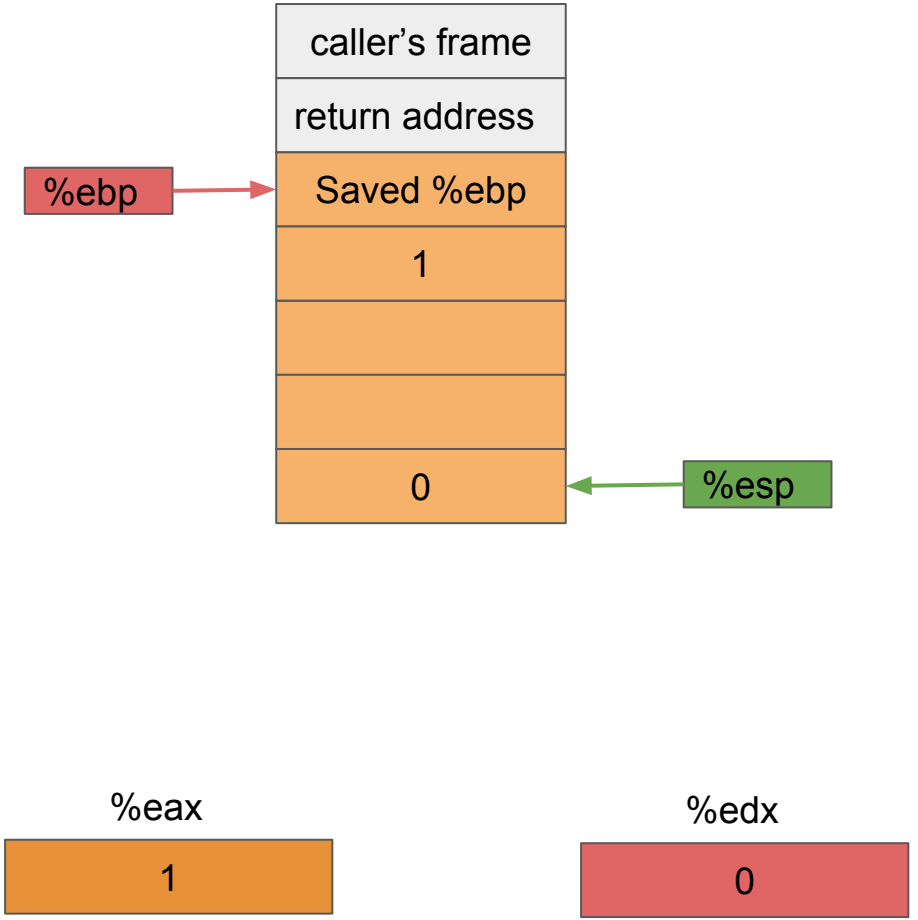
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



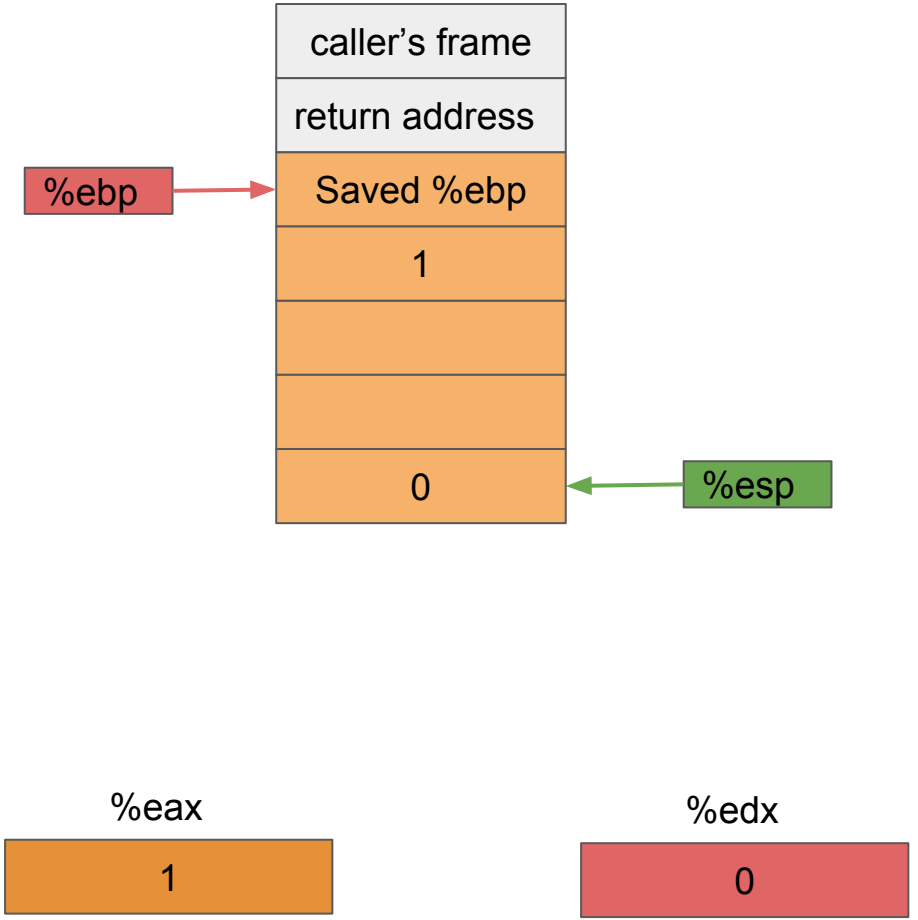
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



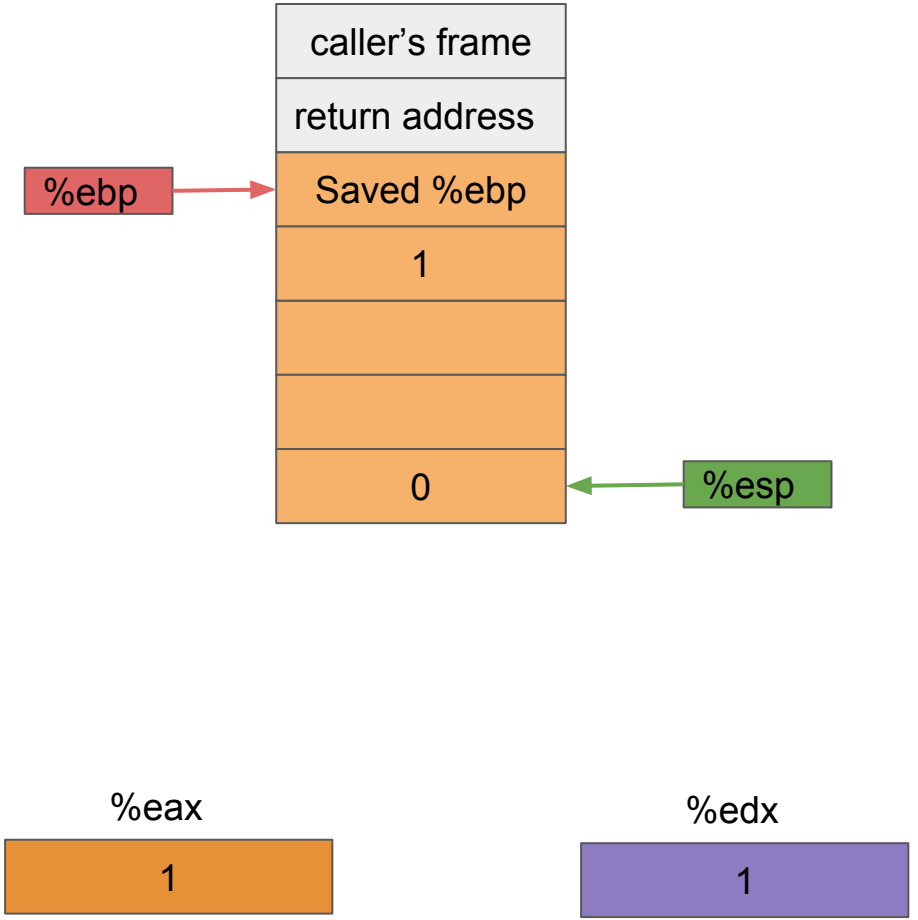
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



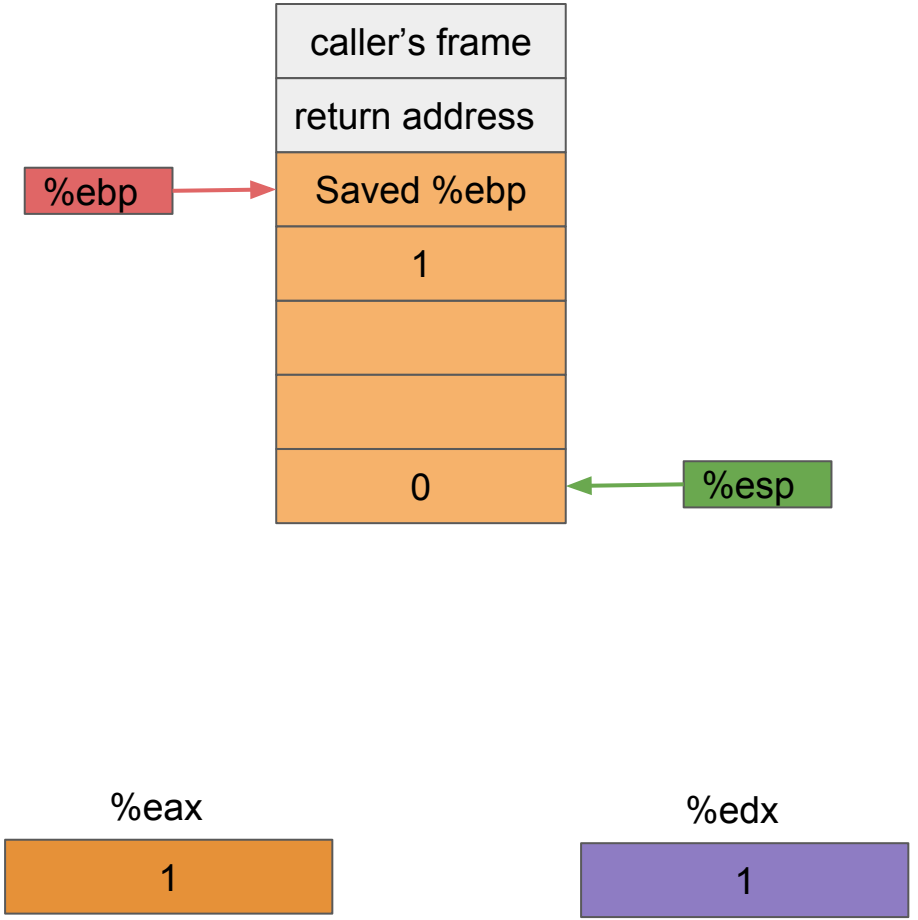
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



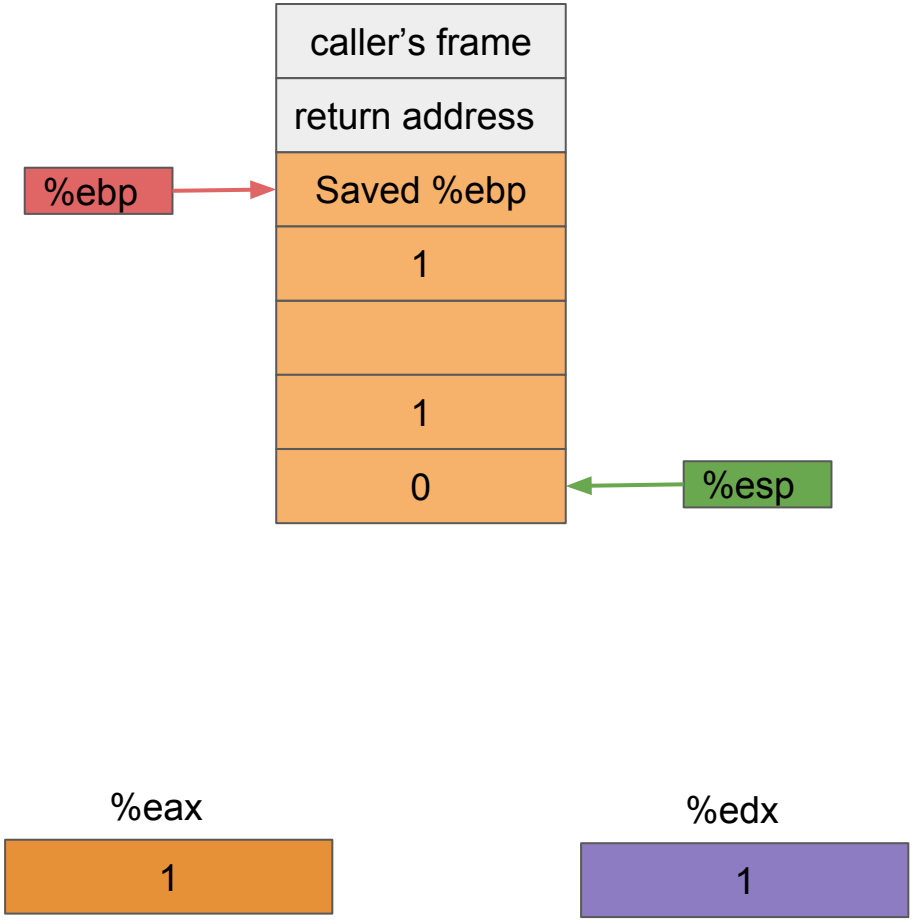
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



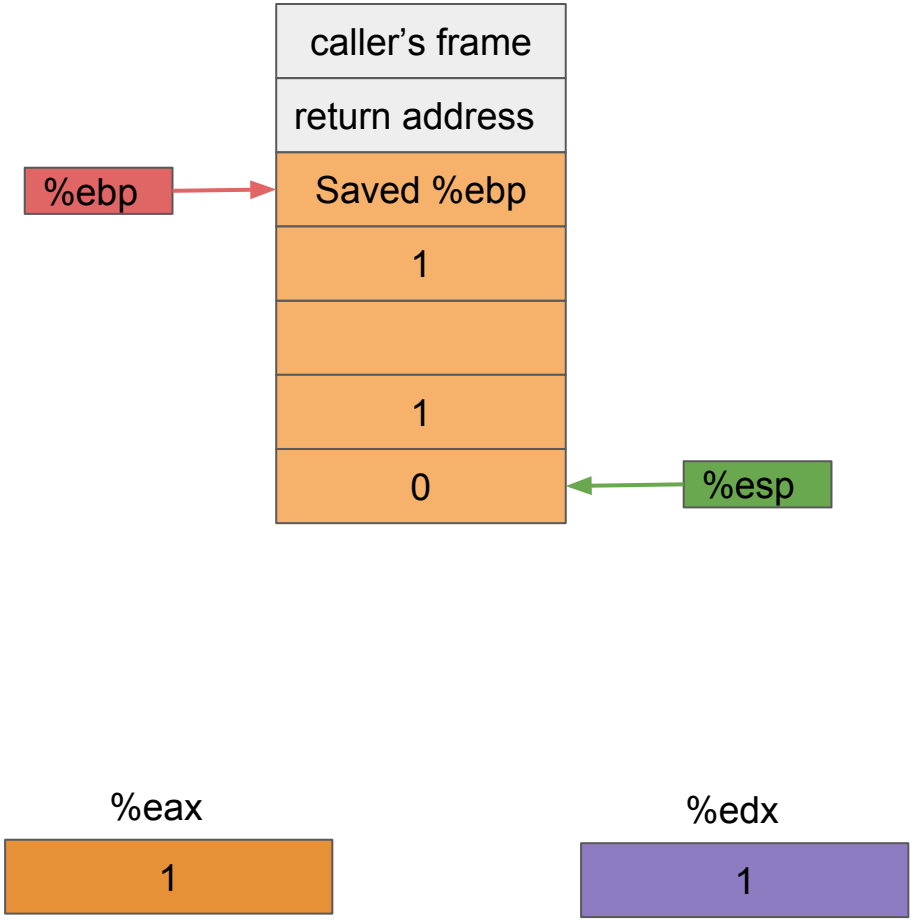
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



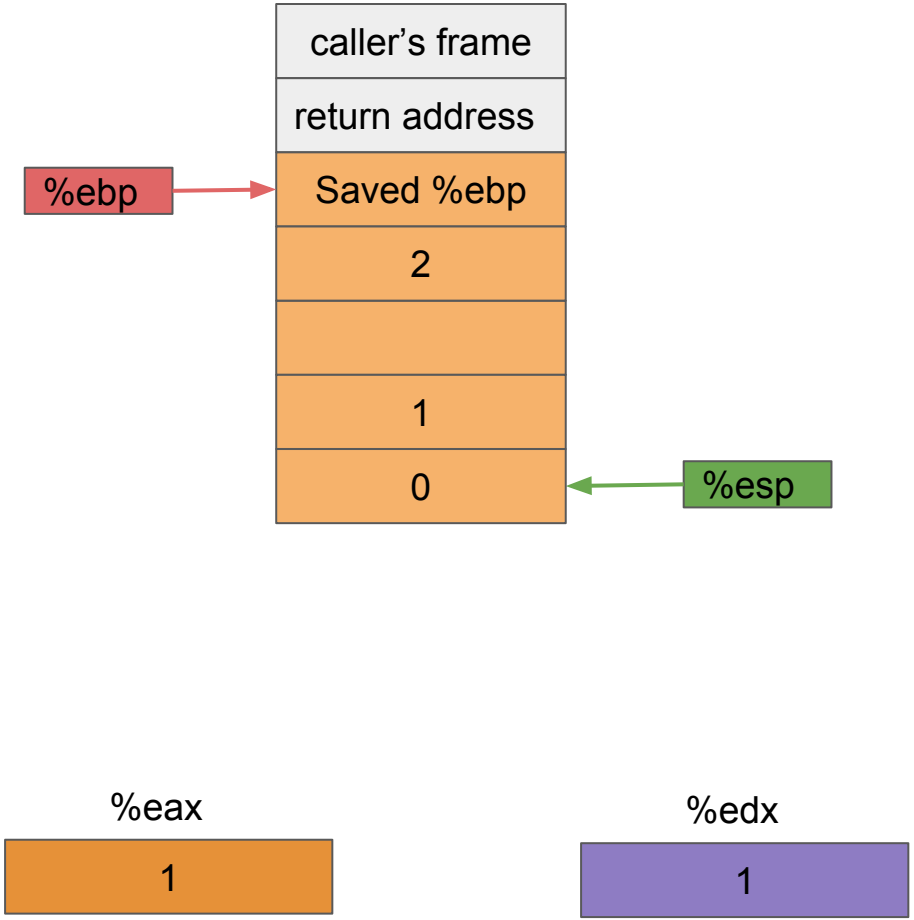
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



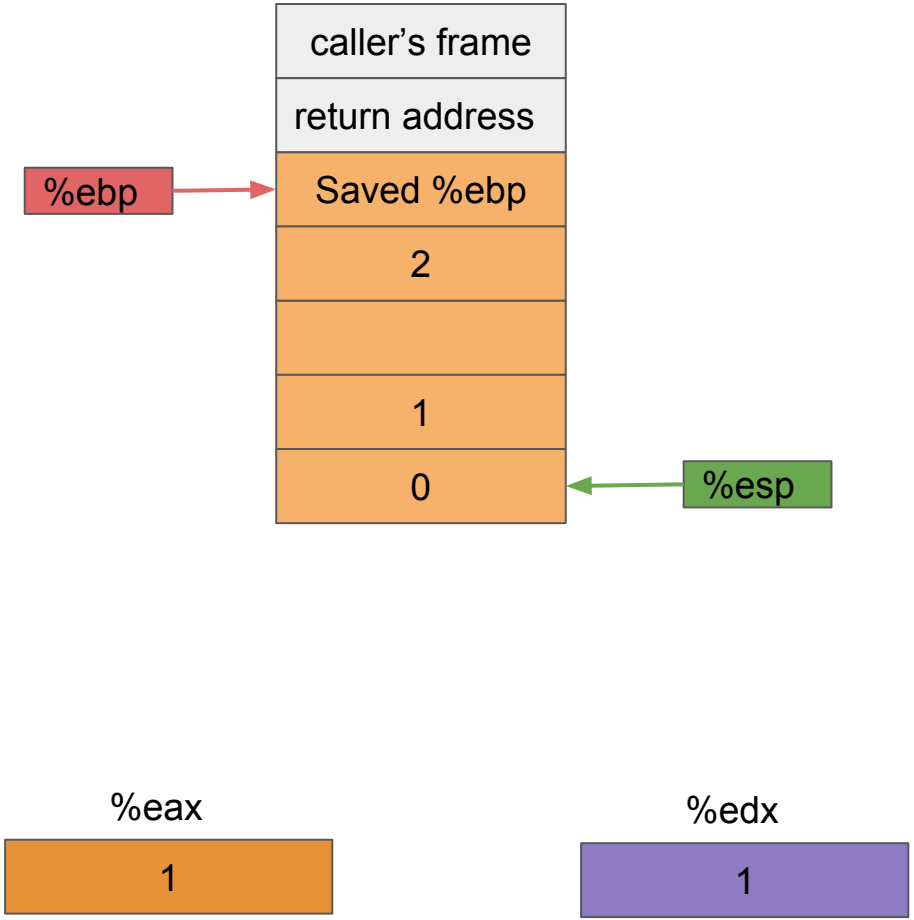
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



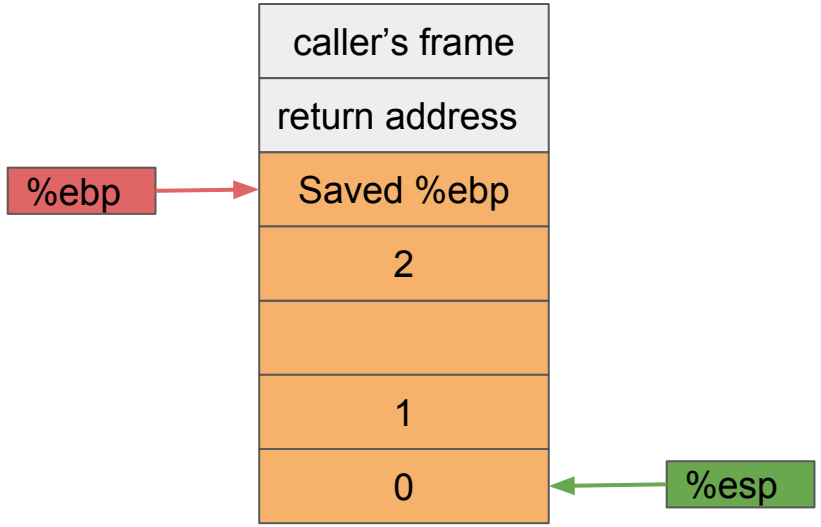
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

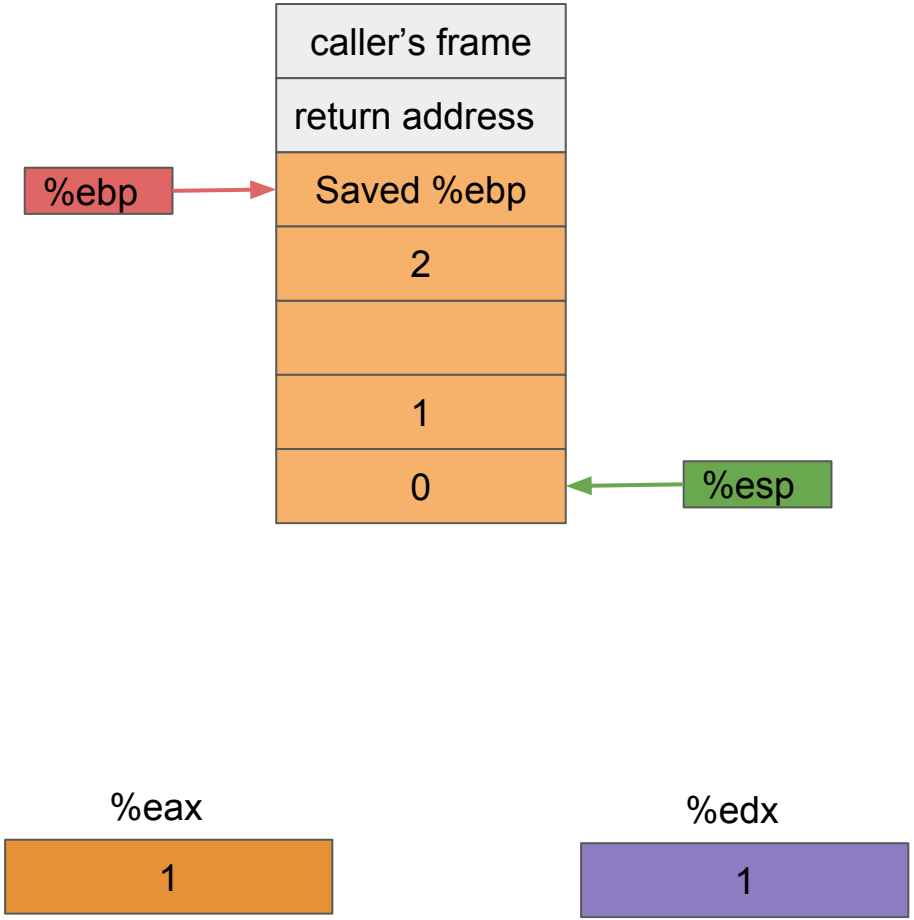


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

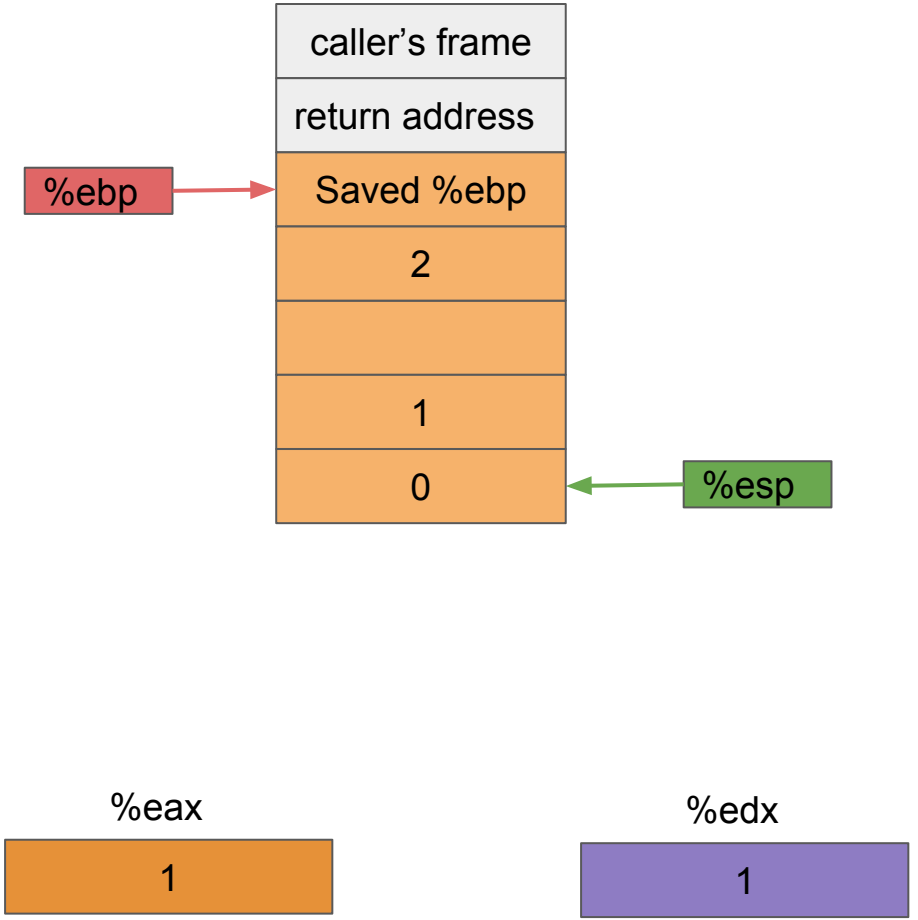


Compare 2 and 2

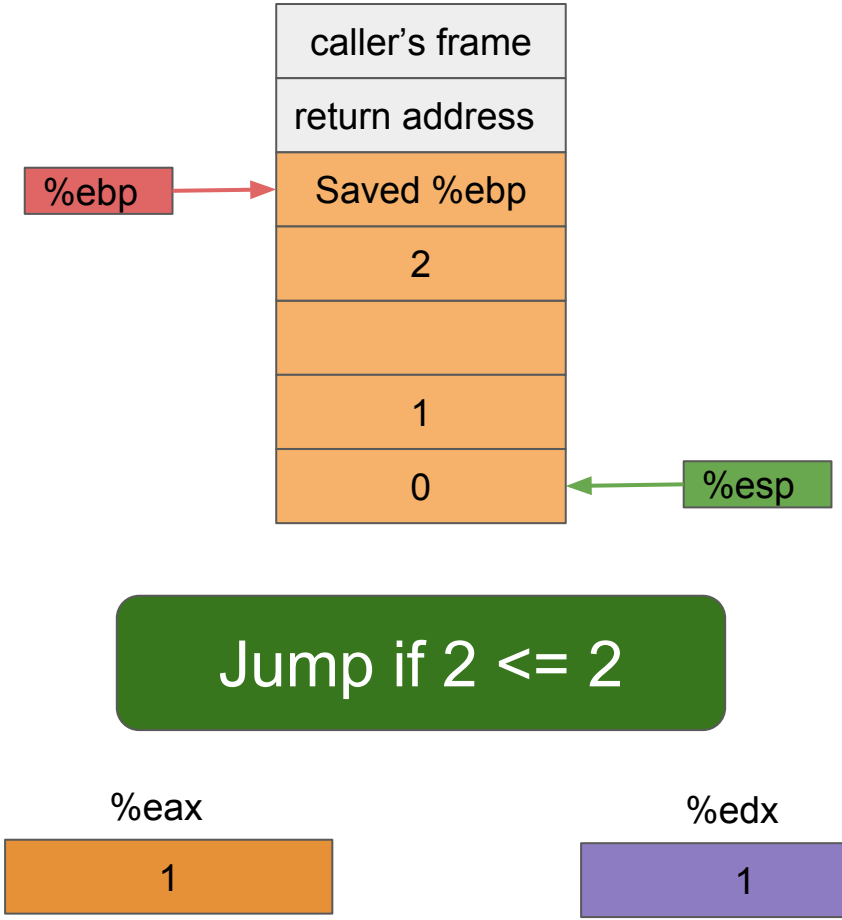
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



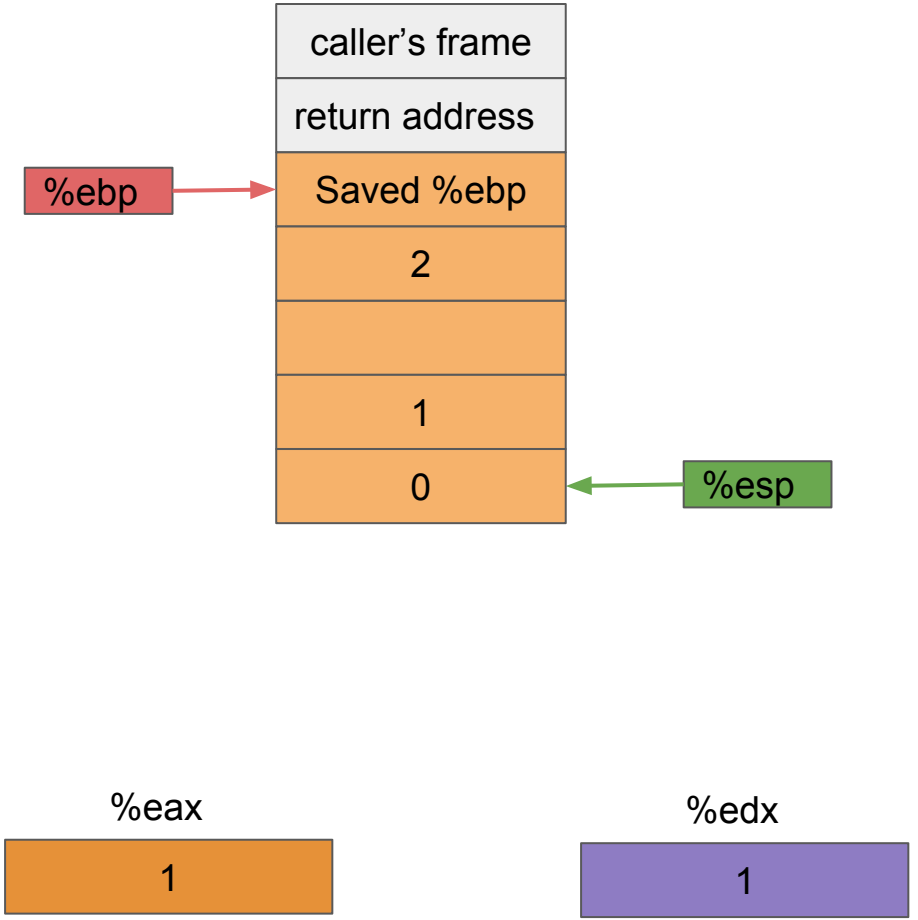
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



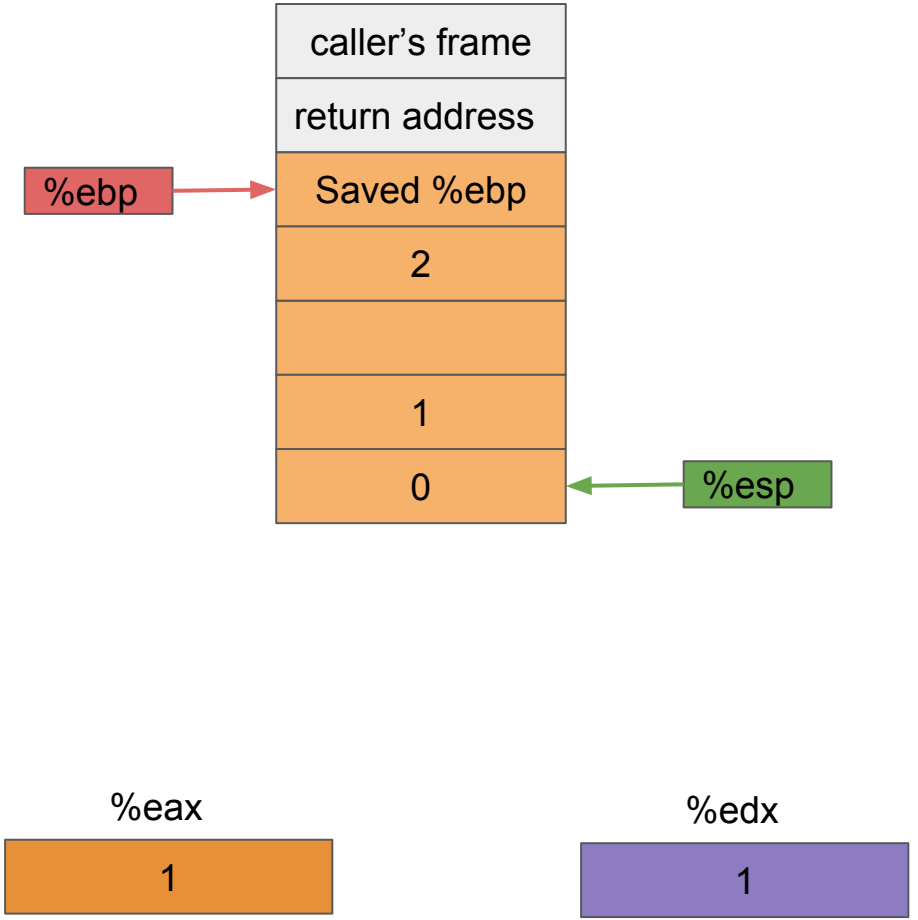
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



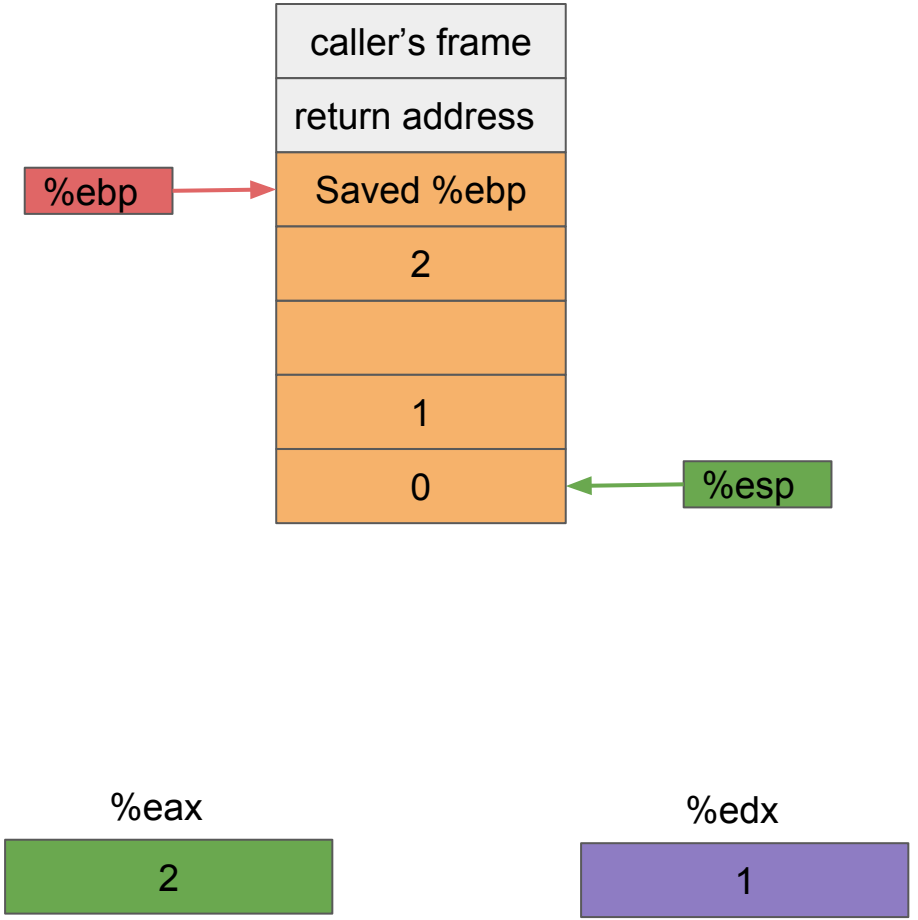
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



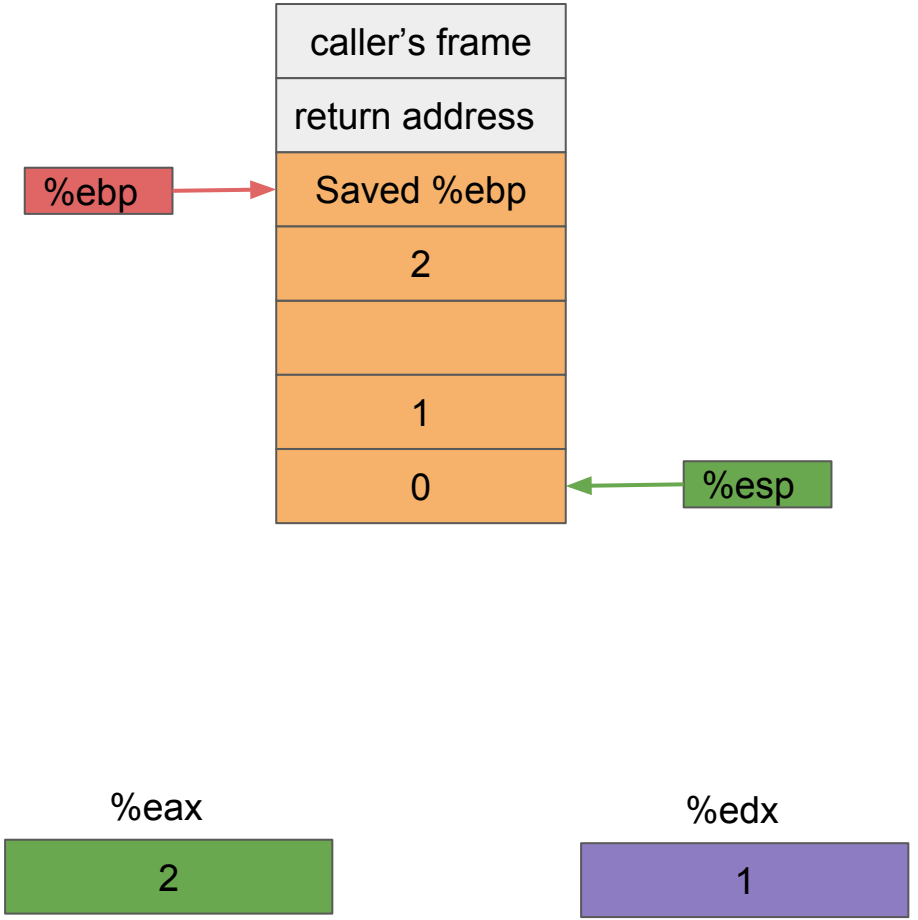
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



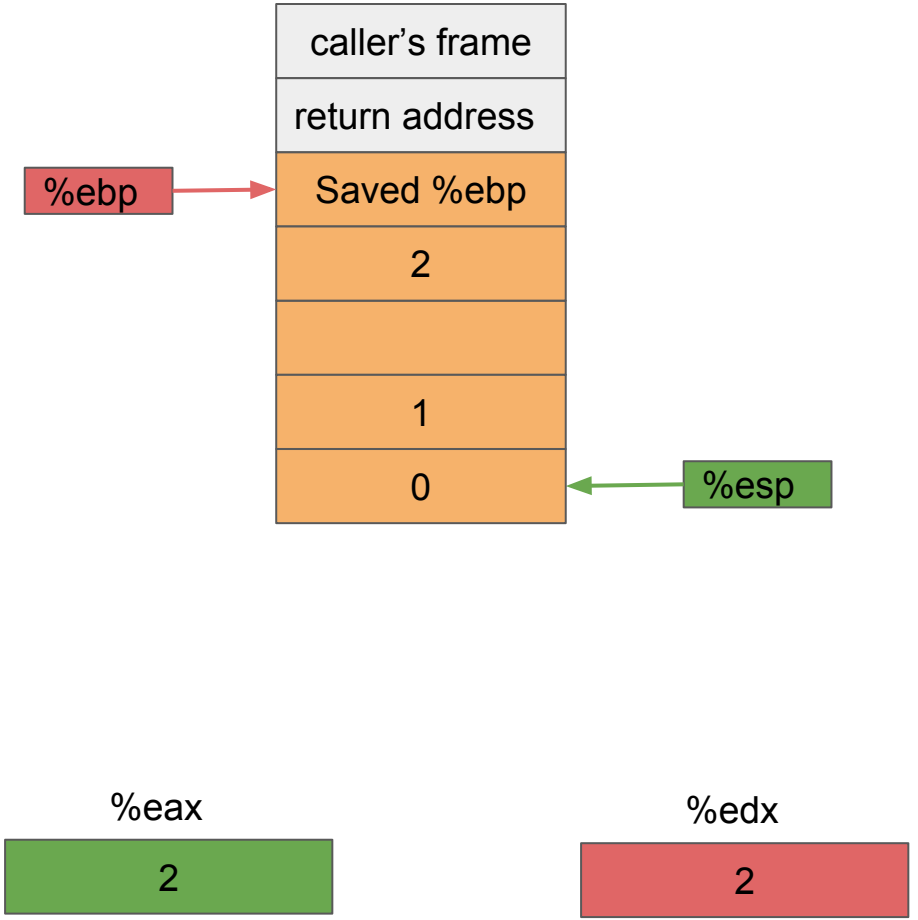
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



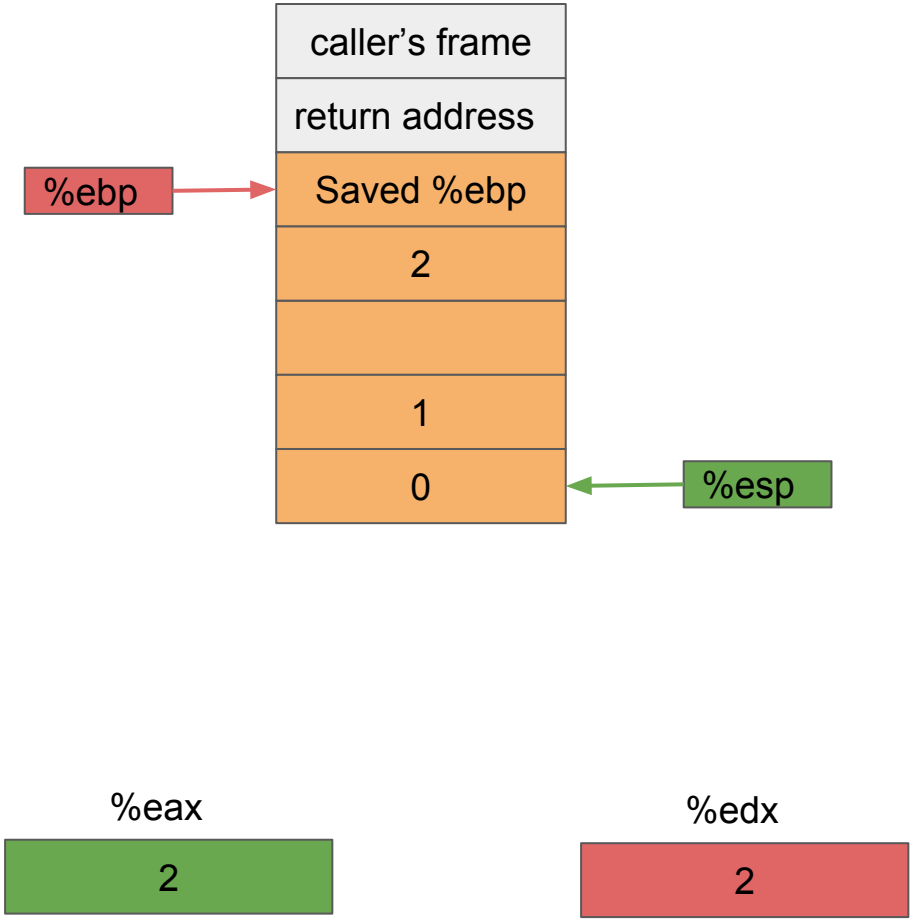
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



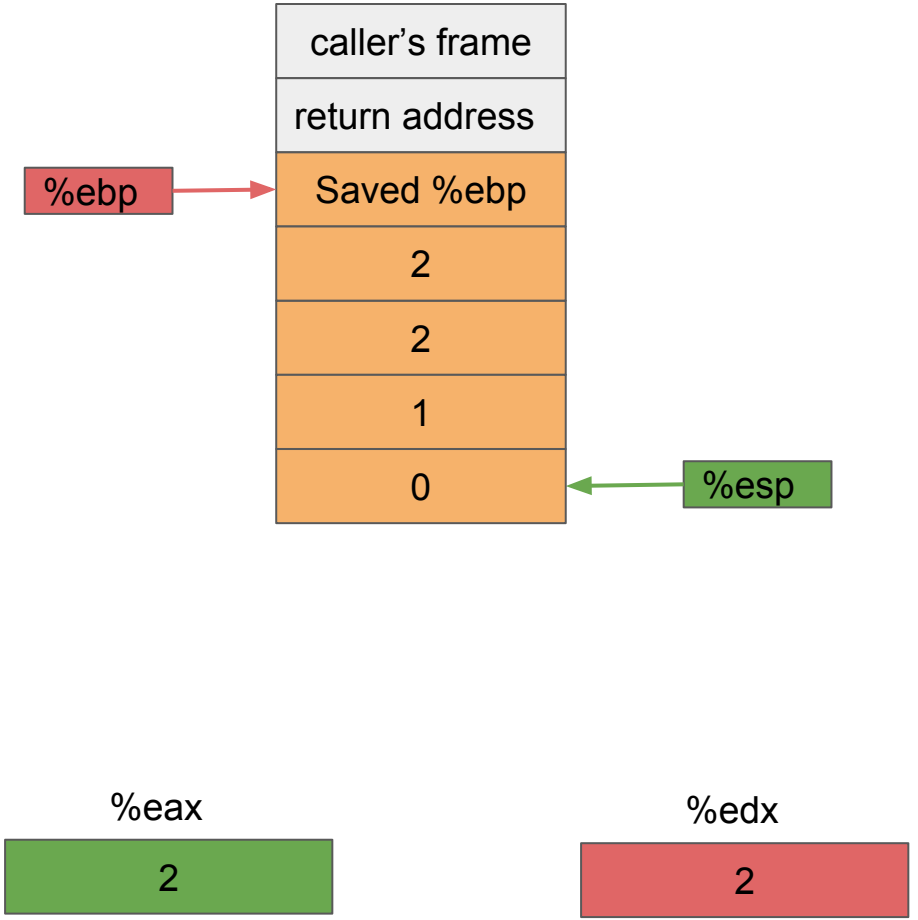
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



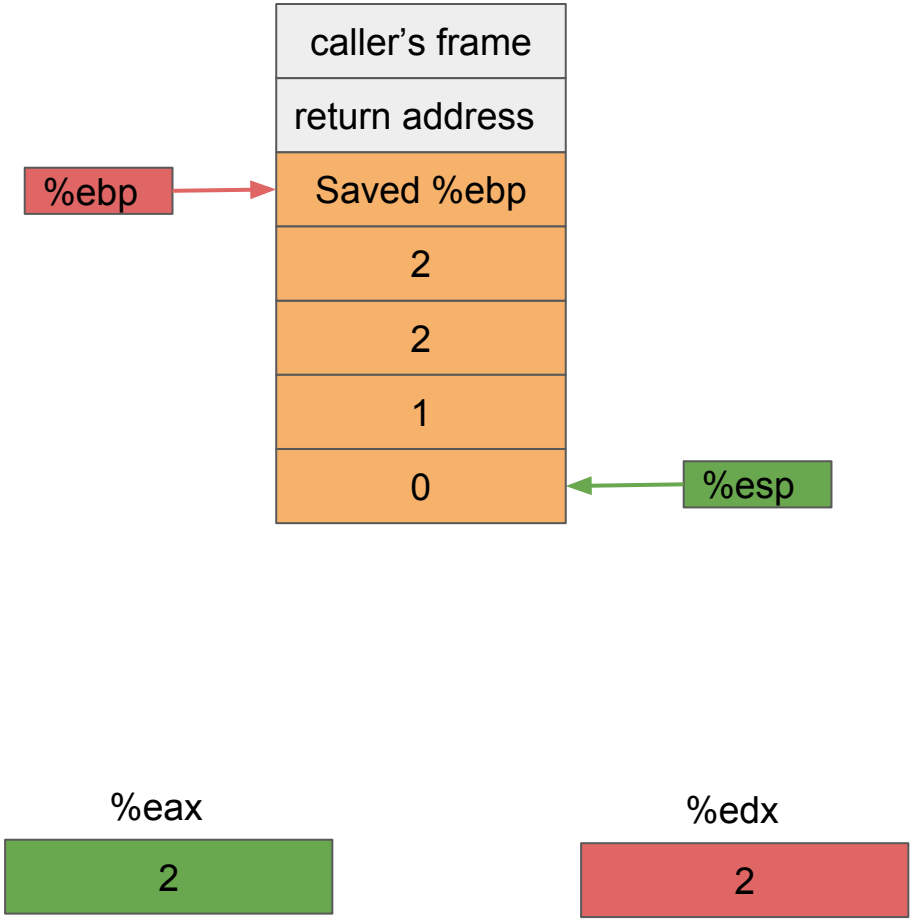
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



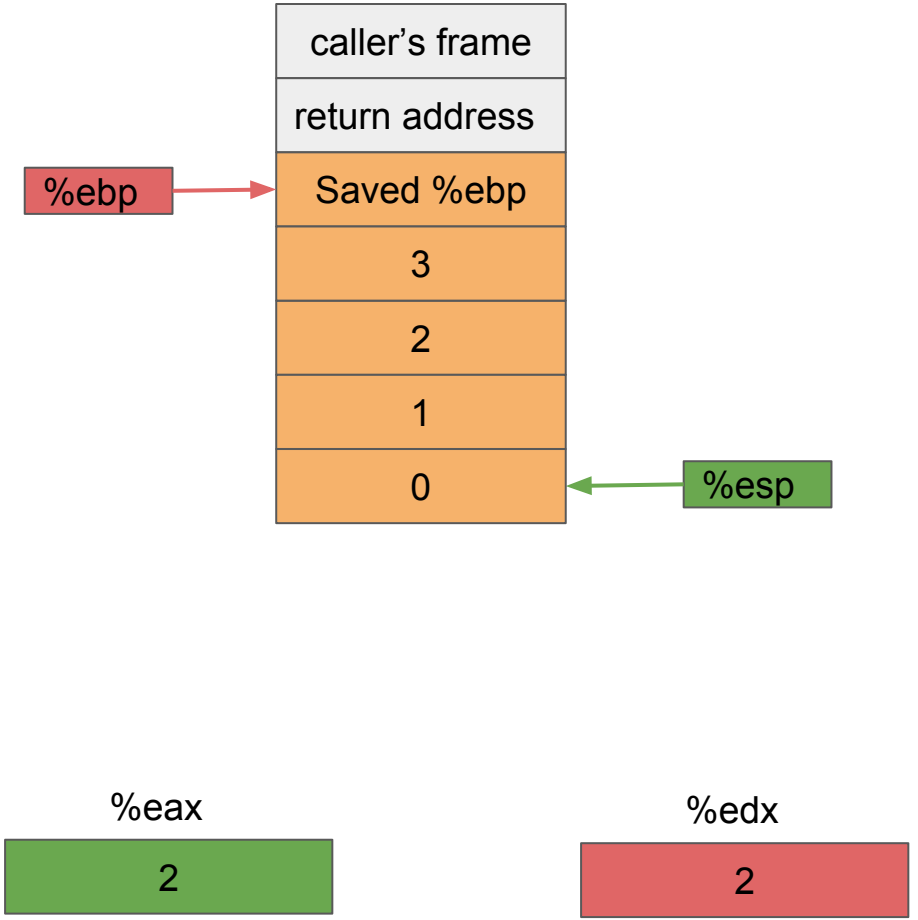
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



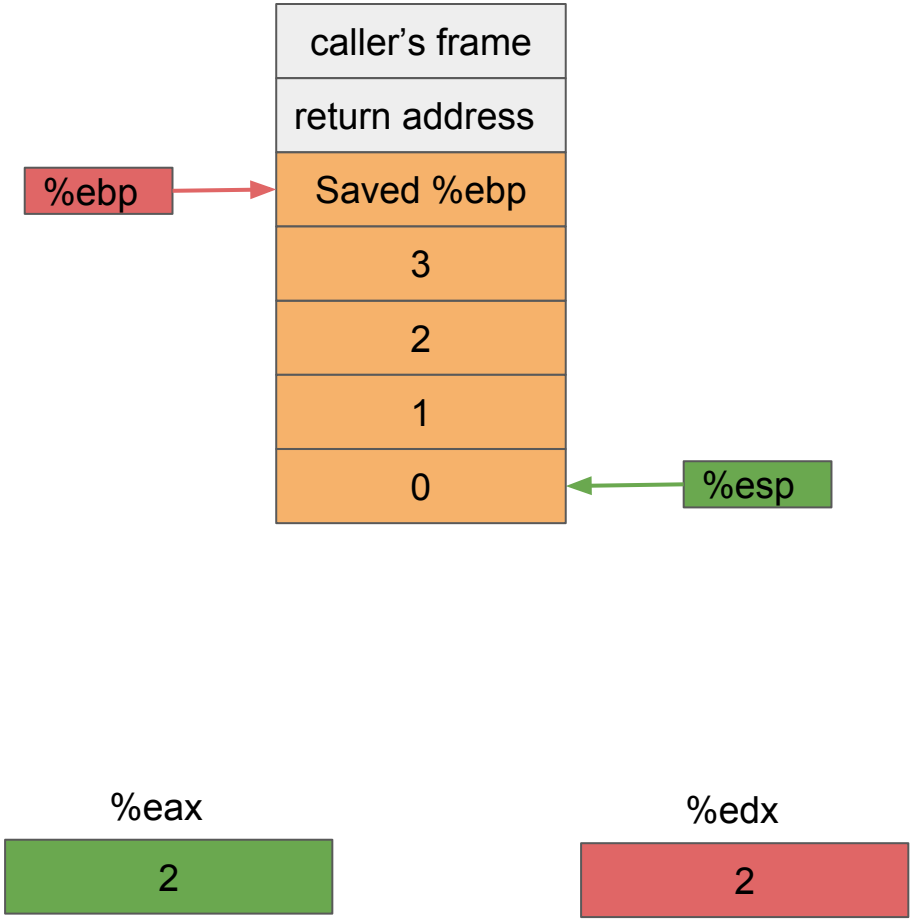
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



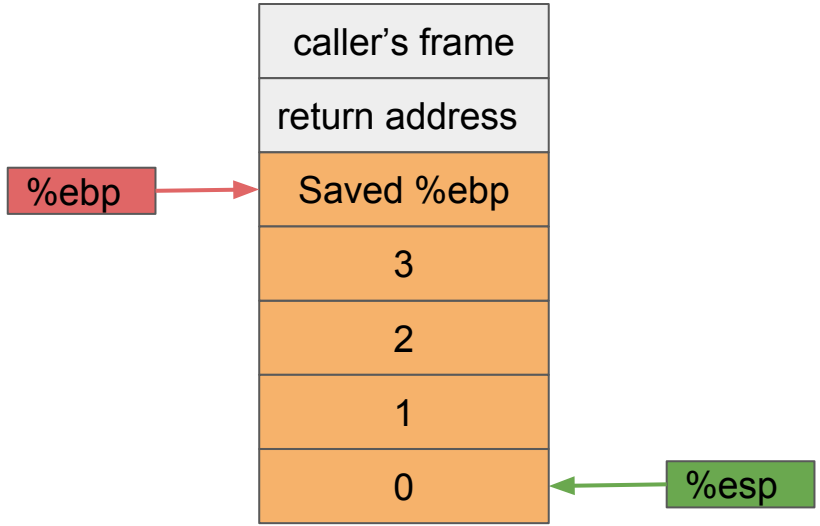
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

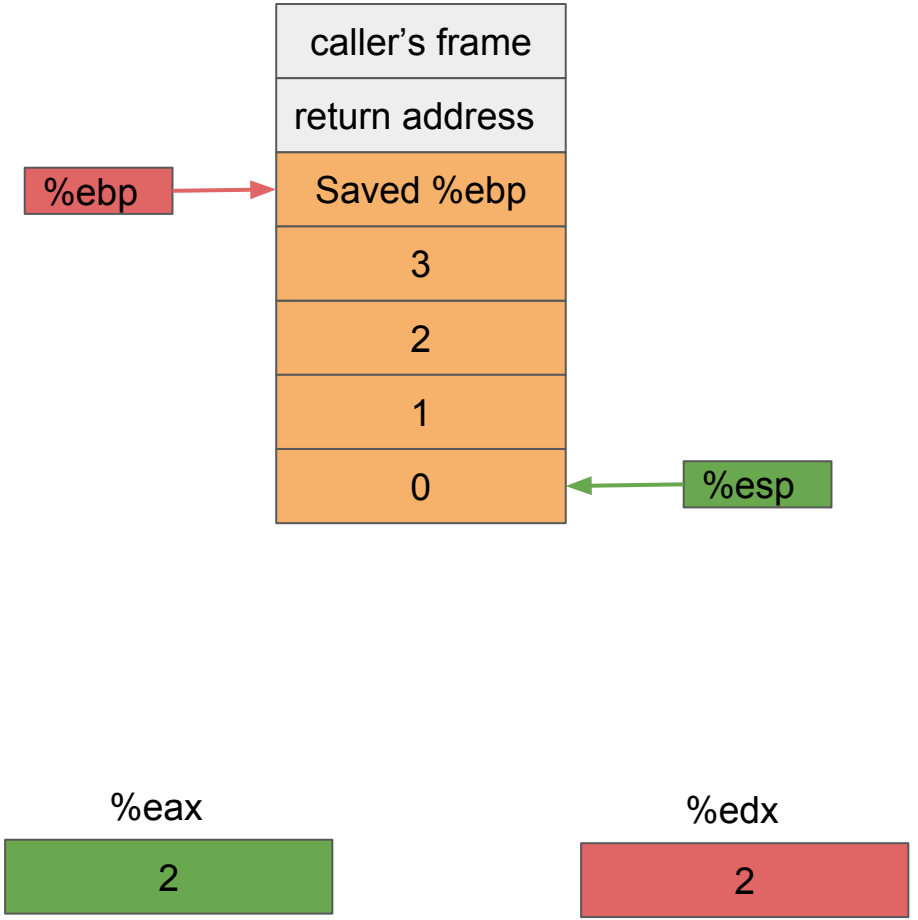


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```

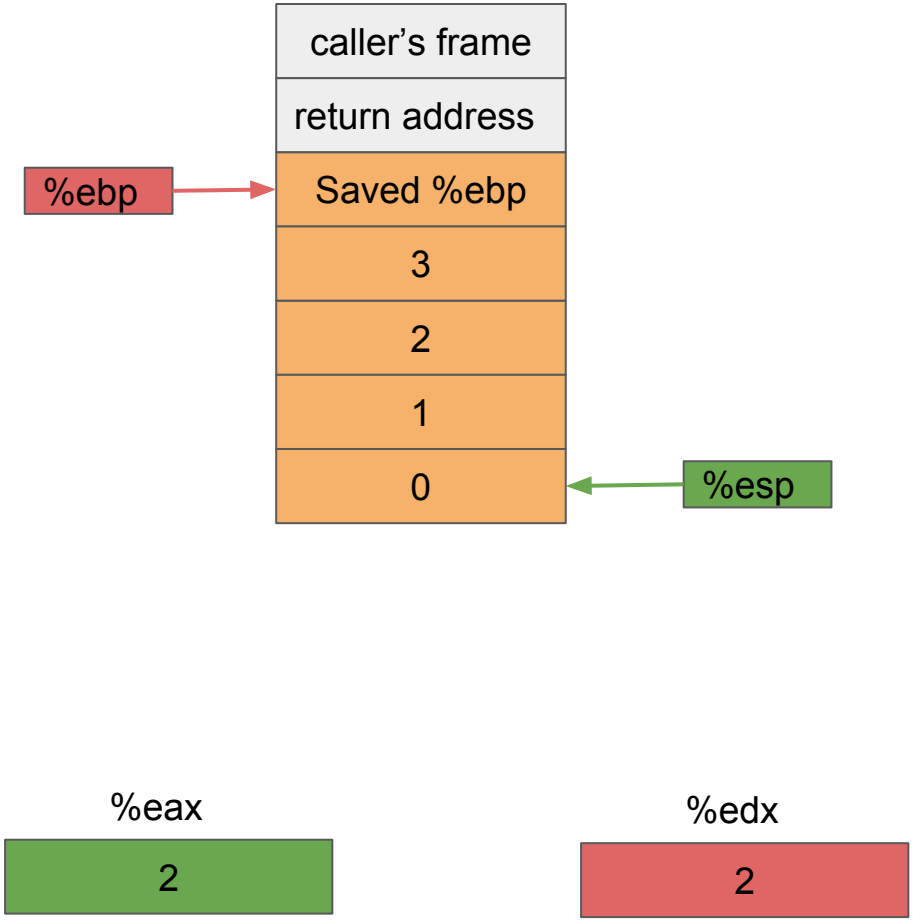


Compare 3 and 2

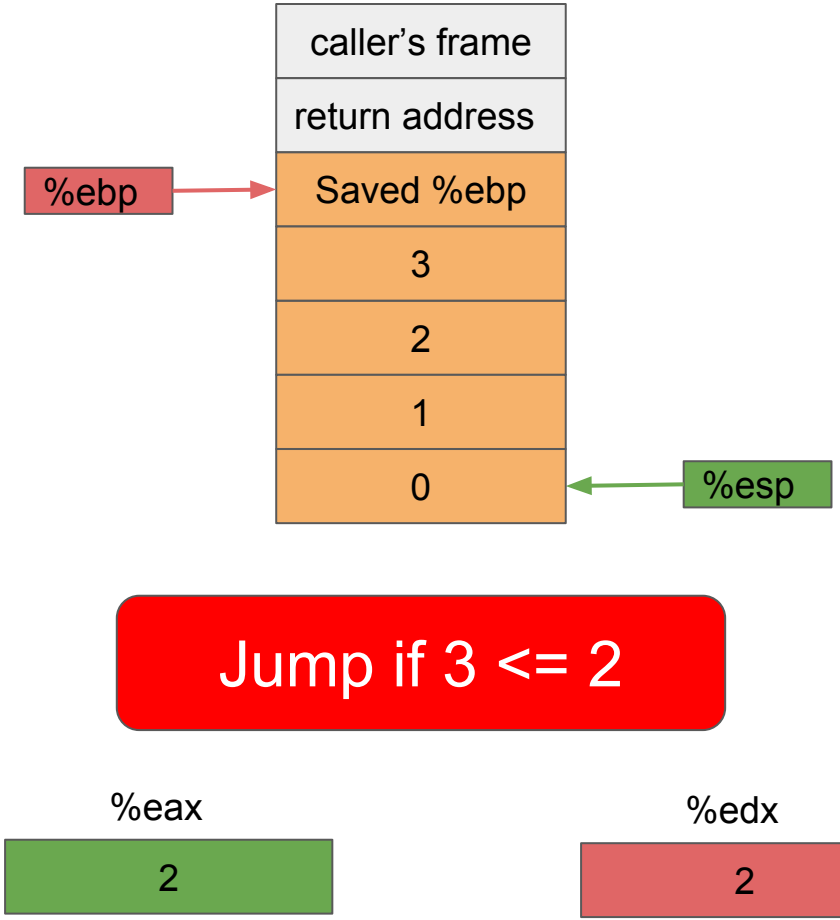
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



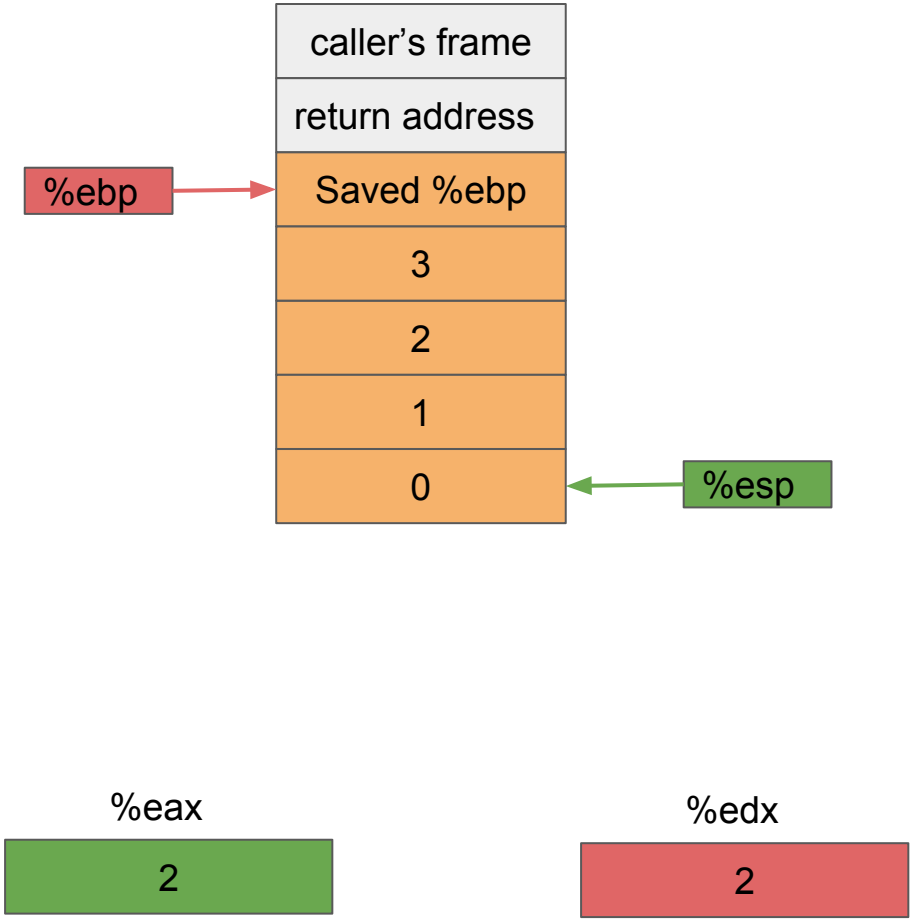
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



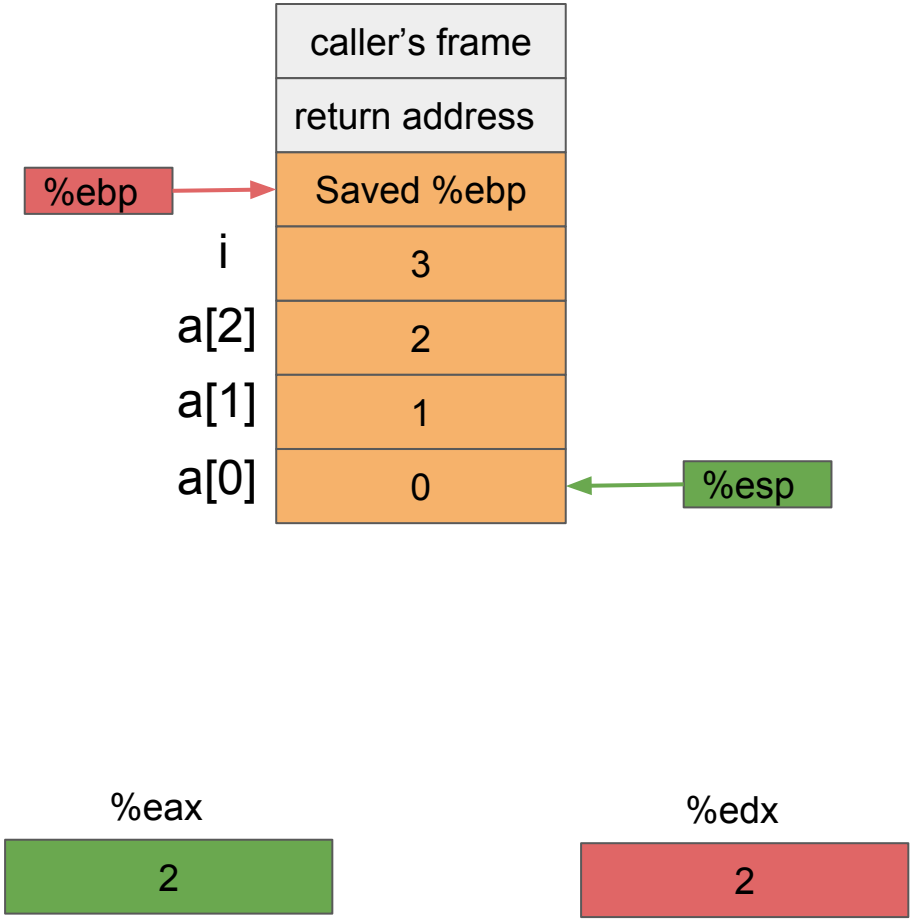
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



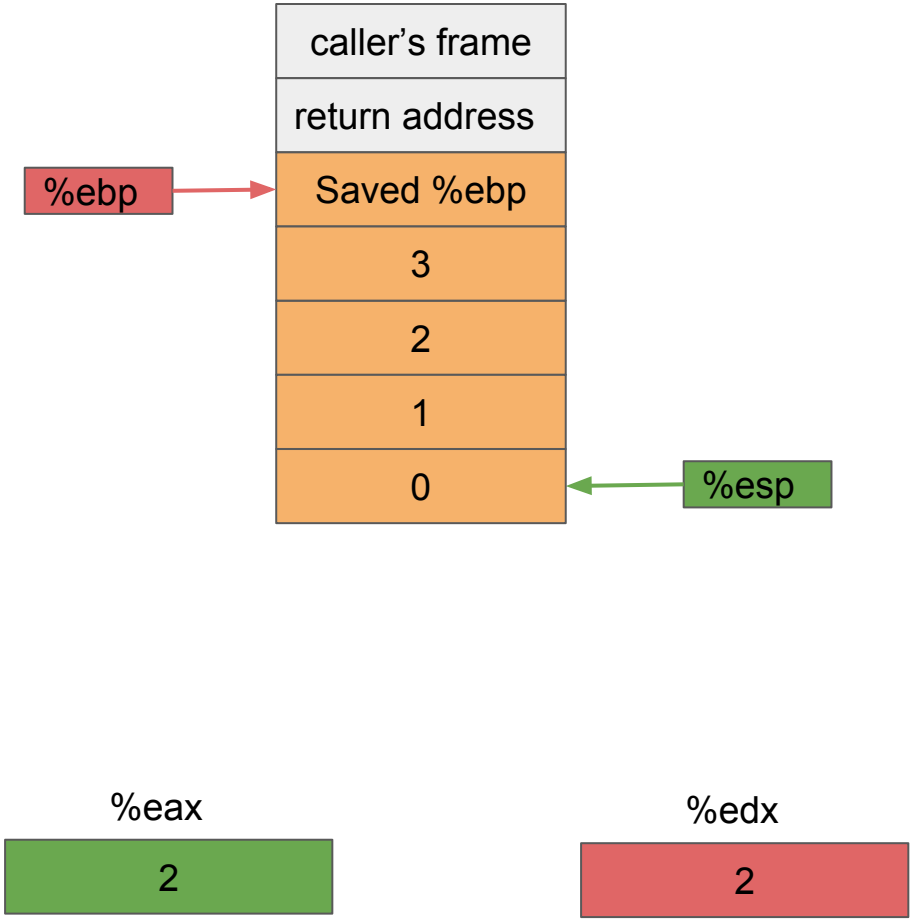
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



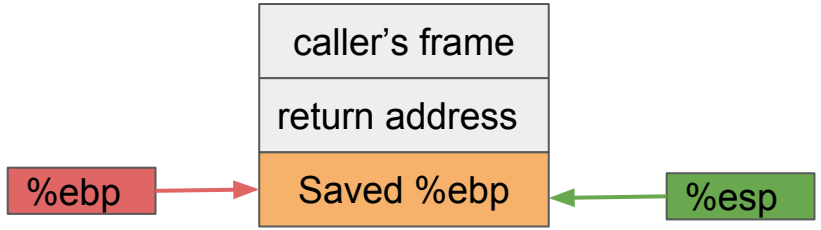
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



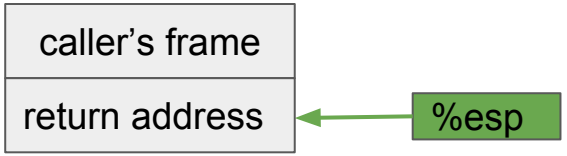
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



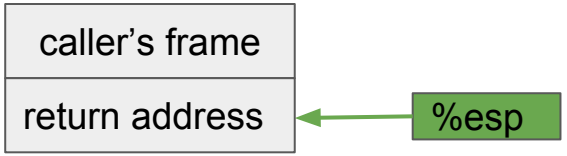
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    -4(%ebp), %edx
    movl    %edx, -16(%ebp,%eax,4)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $2, -4(%ebp)
    jle .L3
    leave
    ret
```