

Arrays in Assembly

Adalbert **Gerald** Soosai Raj

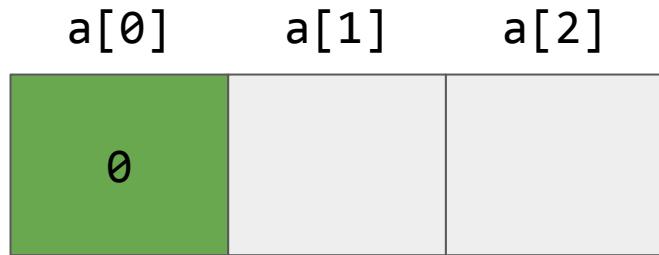
```
#define N 3
```

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

a[0]	a[1]	a[2]

```
#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;
    }
}
```

a[0]	a[1]	a[2]
0	1	

```
#define N 3
```

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

a[0]	a[1]	a[2]
0	1	2

func:

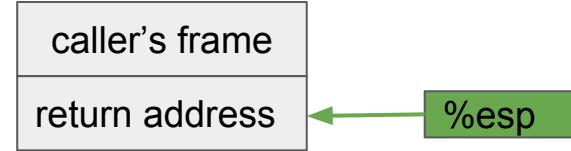
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

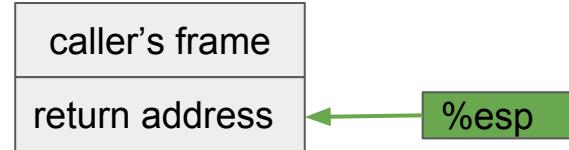
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

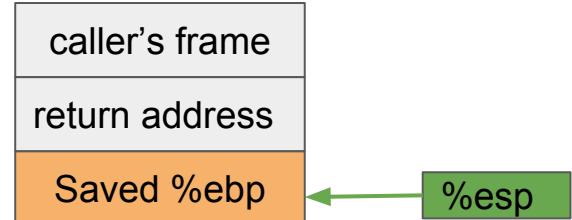
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

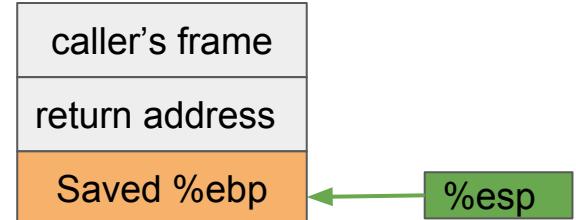
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

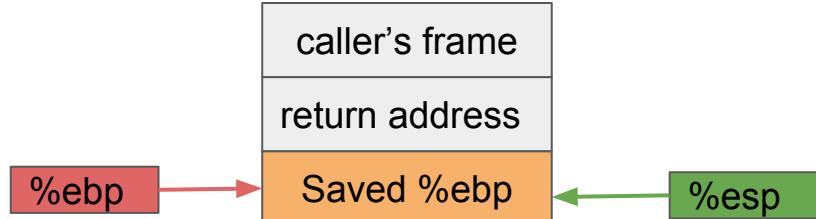
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

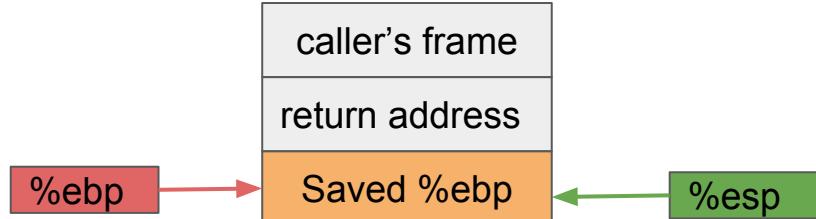
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

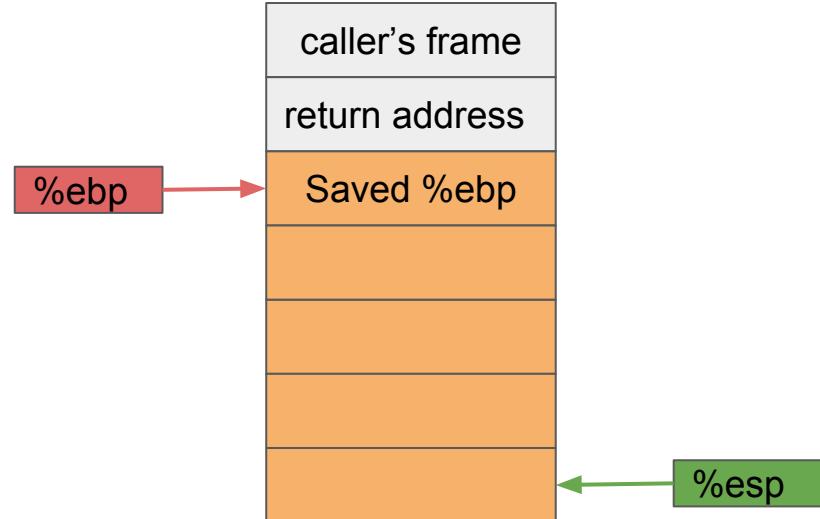
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

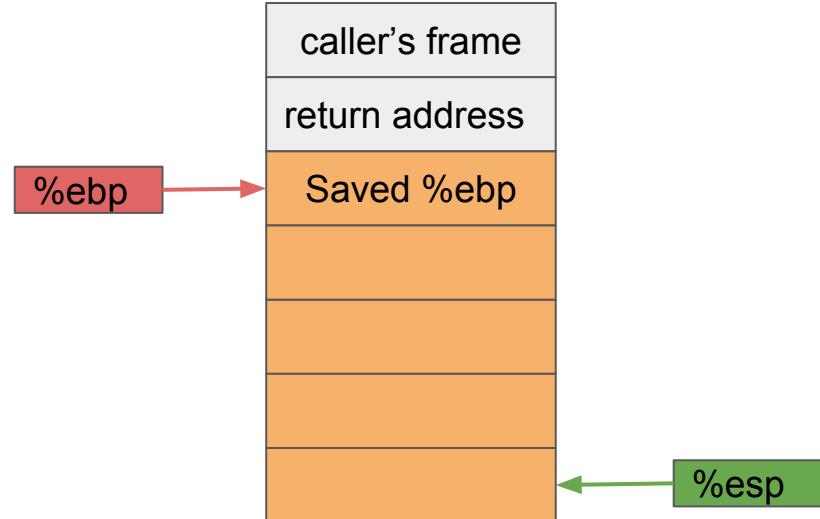
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

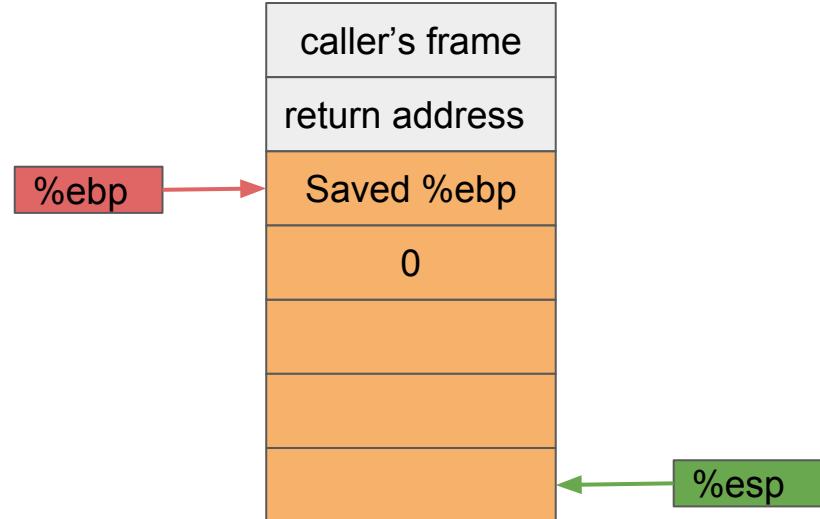
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

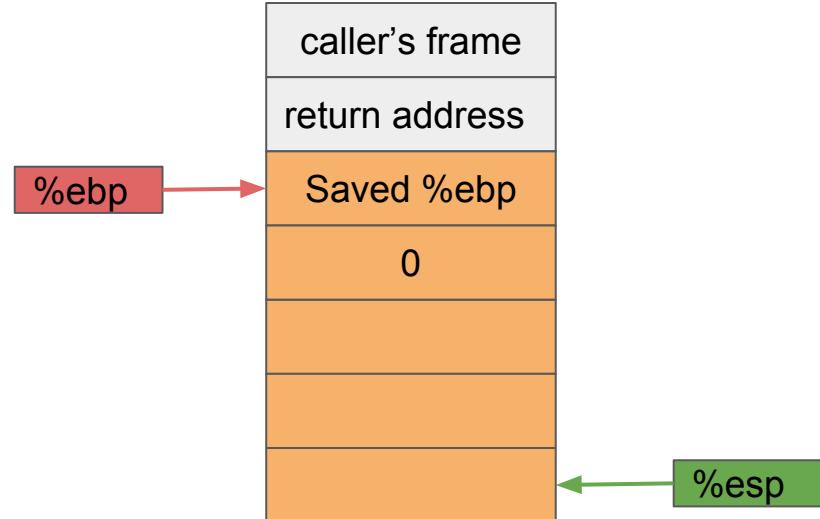
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



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)
```

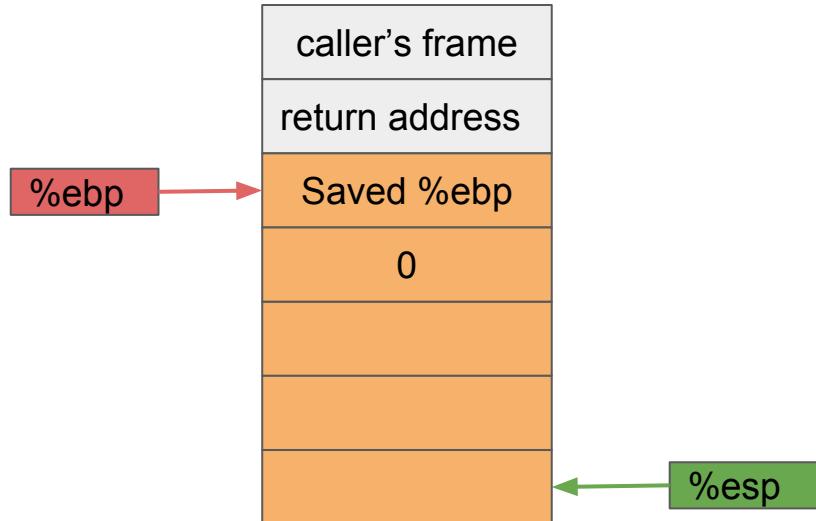
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax



%edx



func:

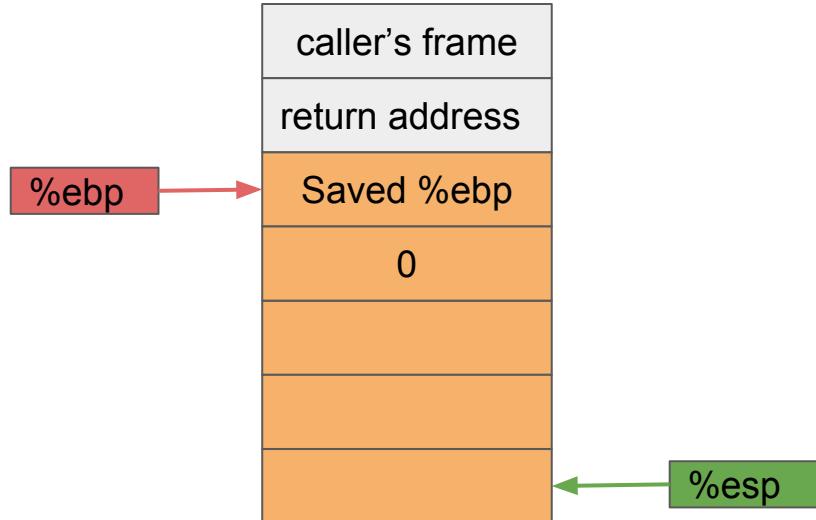
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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 0 and 2

%eax

%edx



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)
```

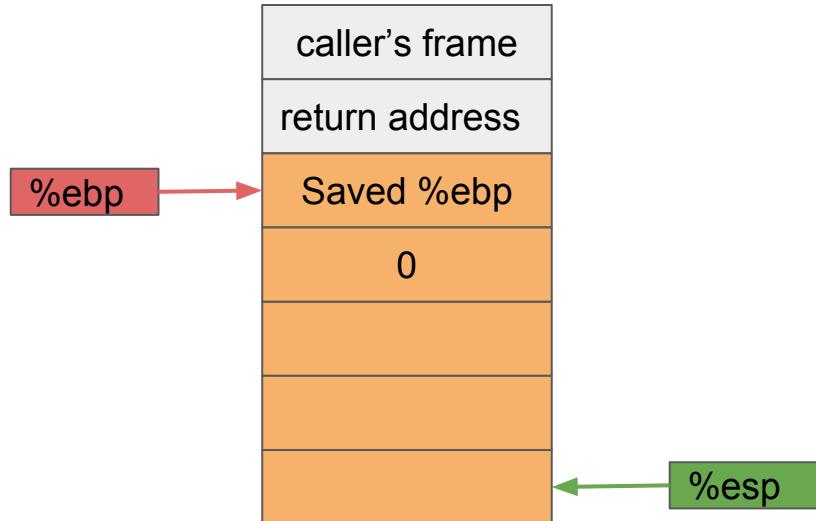
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax



%edx



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)
```

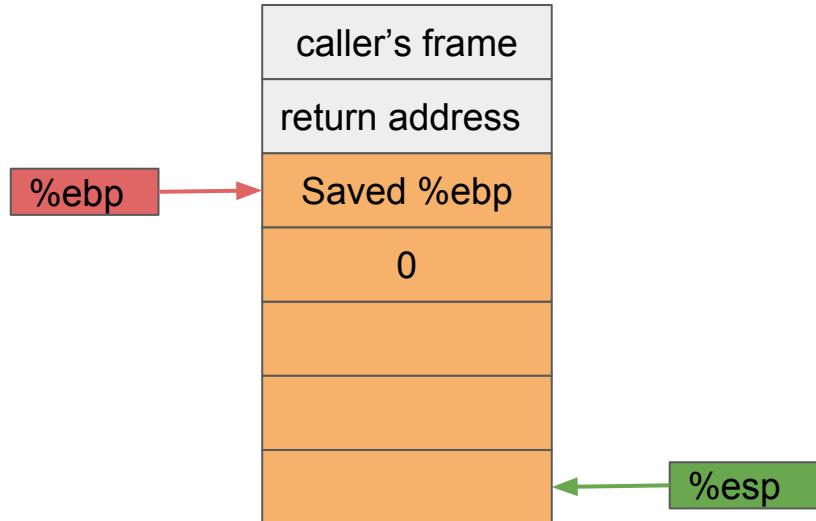
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

leave

ret



%eax

%edx



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)
```

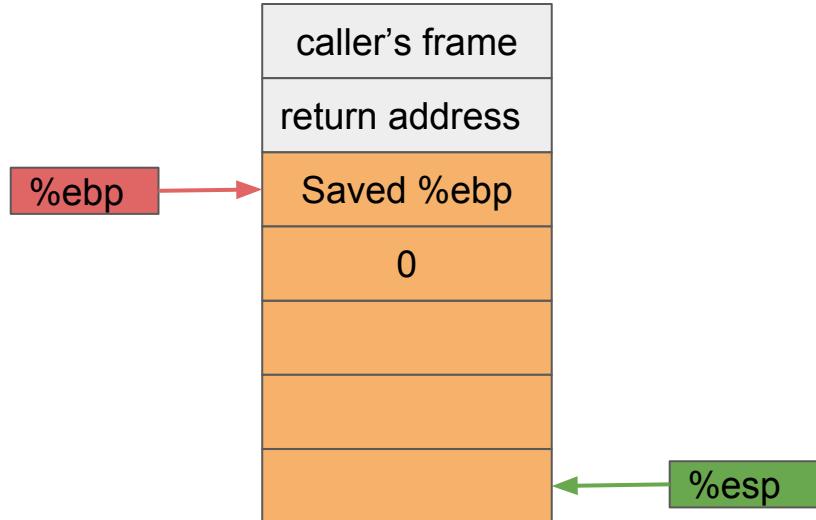
.L2:

```
cmpl $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



Jump if $0 \leq 2$

%eax

%edx

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)
```

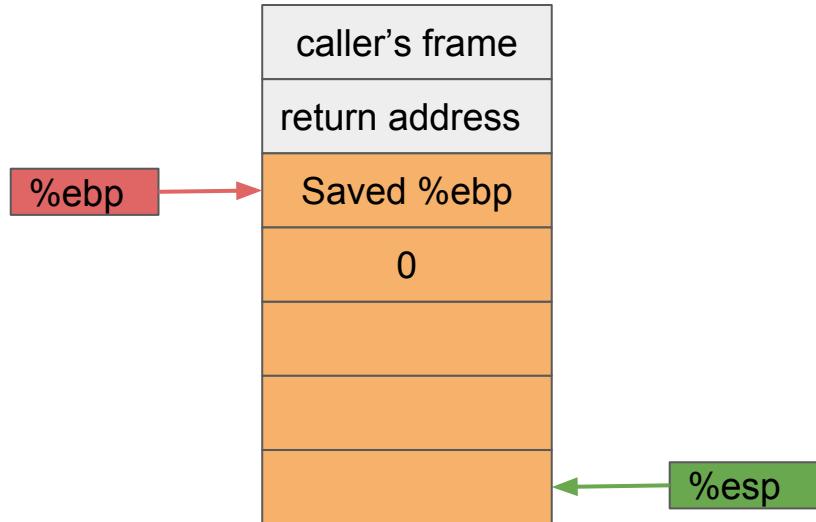
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

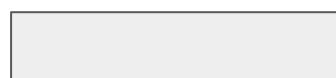
leave

ret



%eax

%edx



func:

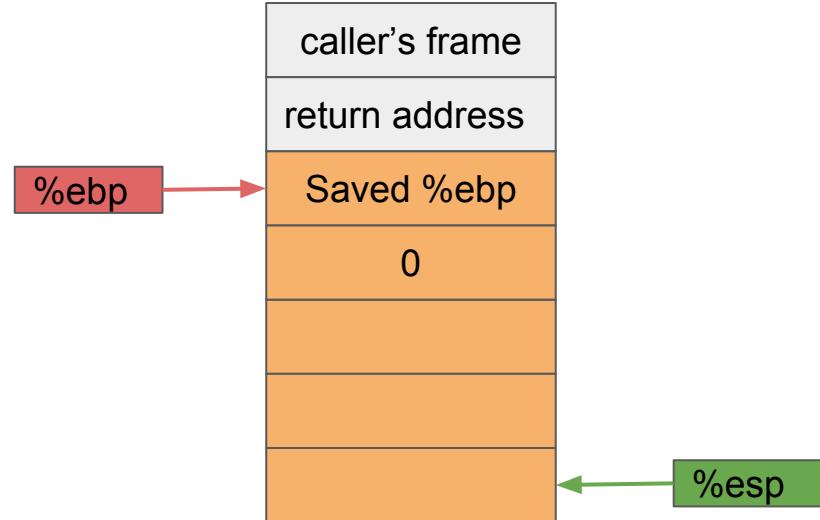
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx



func:

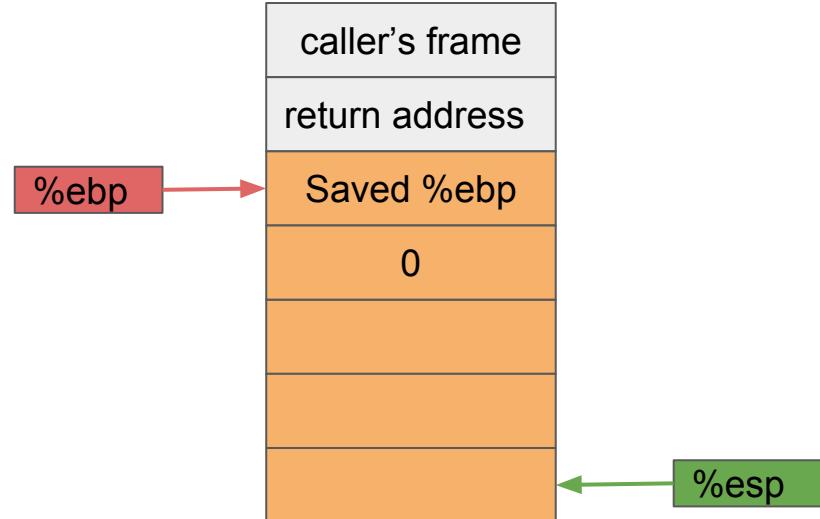
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

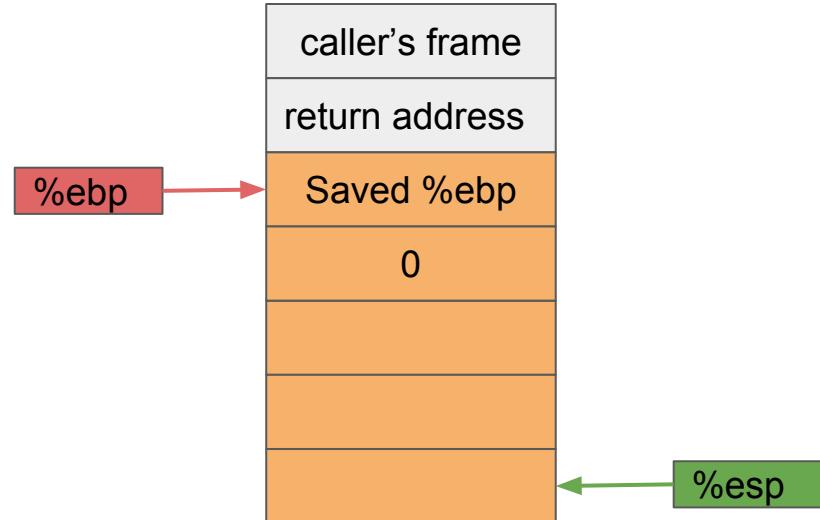
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

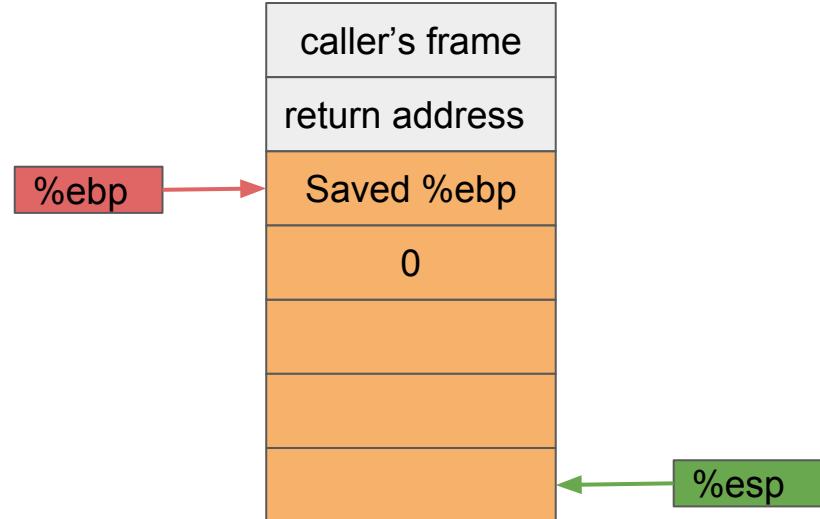
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

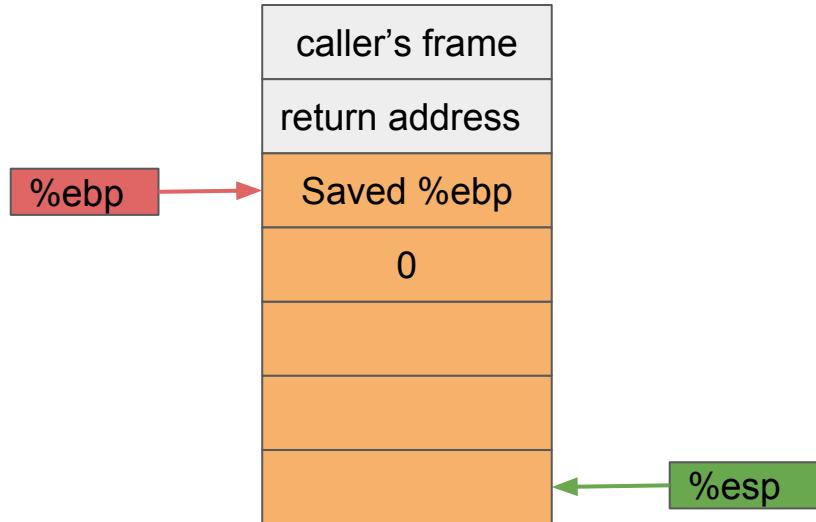
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

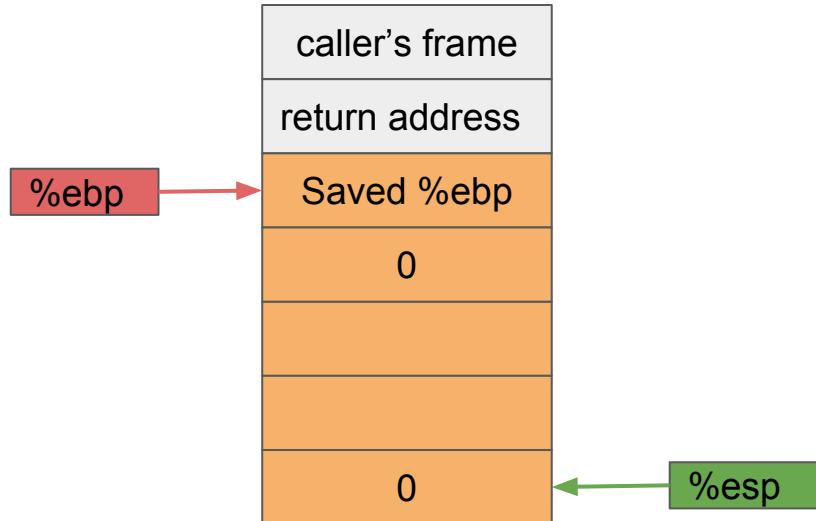
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx

0

0

func:

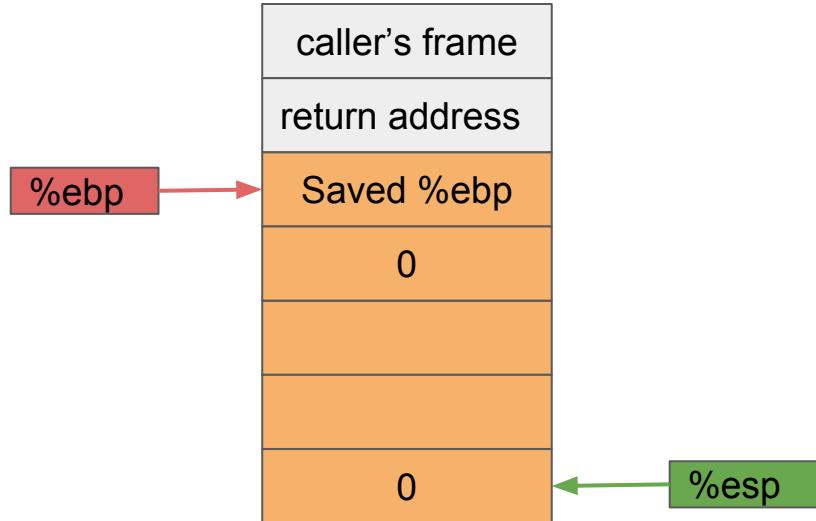
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx

0

0

func:

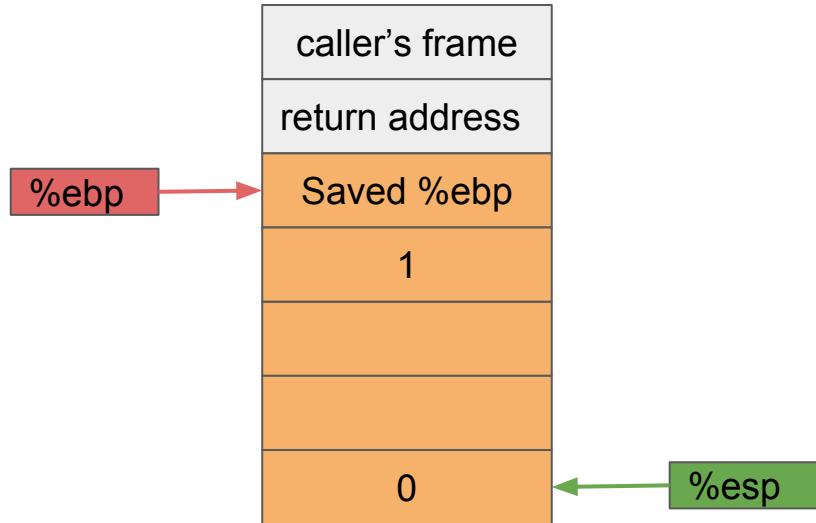
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx

0

0

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)
```

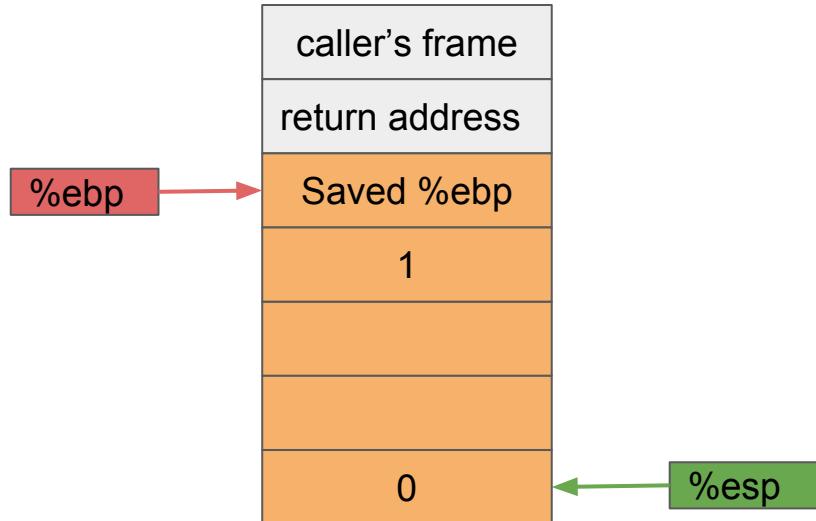
.L2:

```
cmpl $2, -4(%ebp)
```

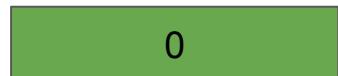
```
jle .L3
```

```
leave
```

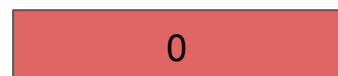
```
ret
```



%eax



%edx



func:

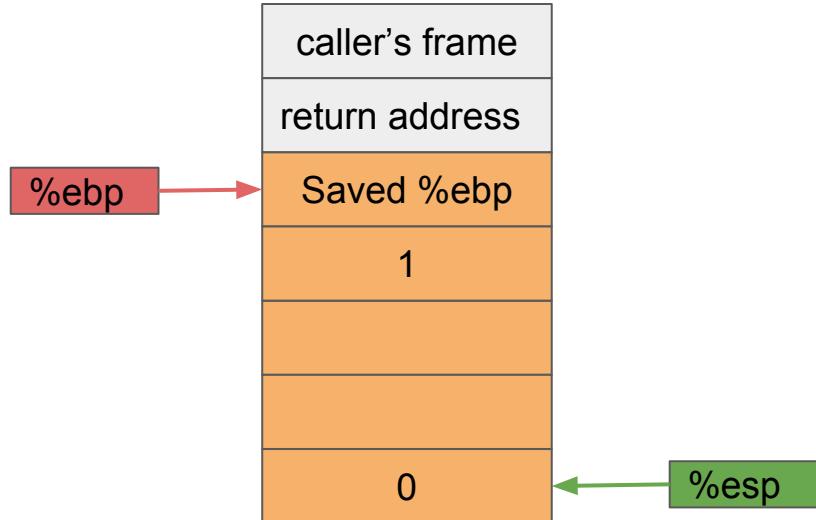
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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

%eax

%edx

0

0

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)
```

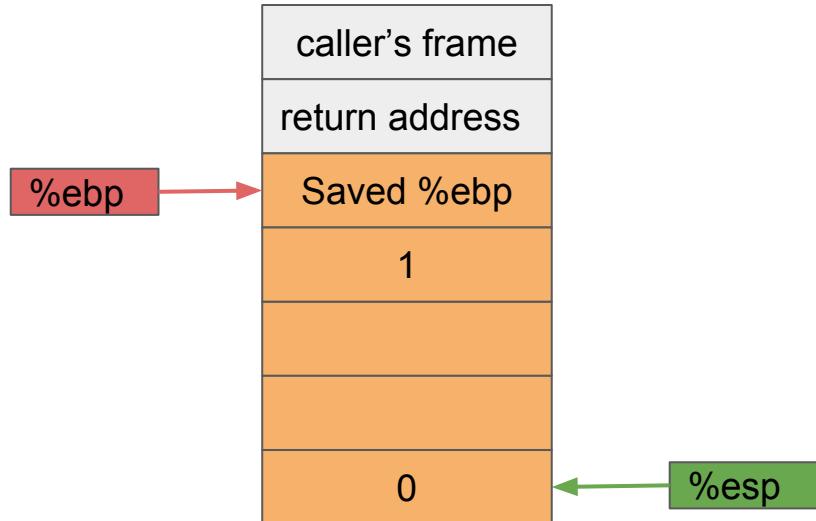
.L2:

```
cmpl $2, -4(%ebp)
```

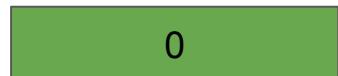
```
jle .L3
```

```
leave
```

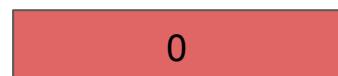
```
ret
```



%eax



%edx



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)
```

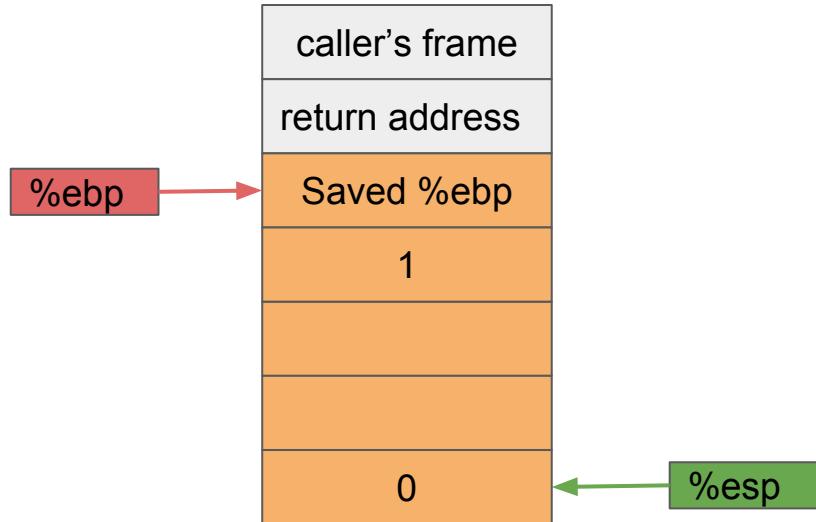
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

```
leave
```

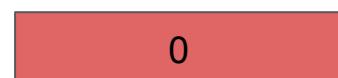
```
ret
```



%eax



%edx



func:

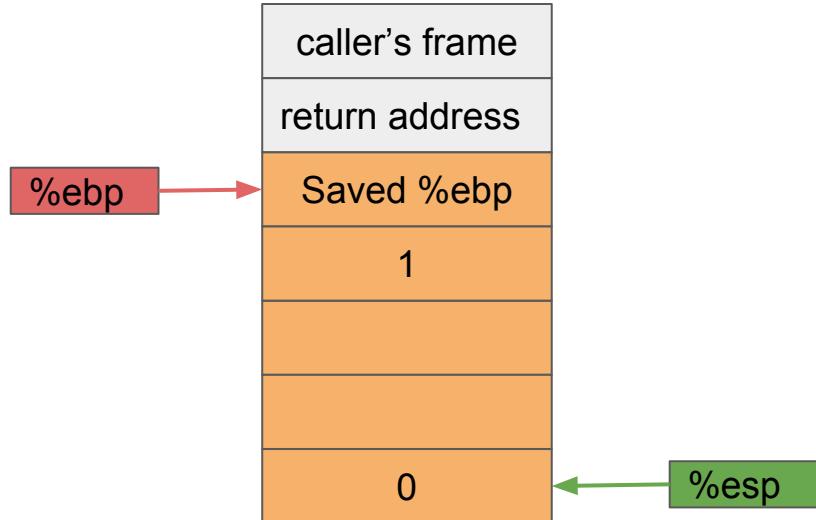
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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 $1 \leq 2$

%eax

%edx

0

0

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)
```

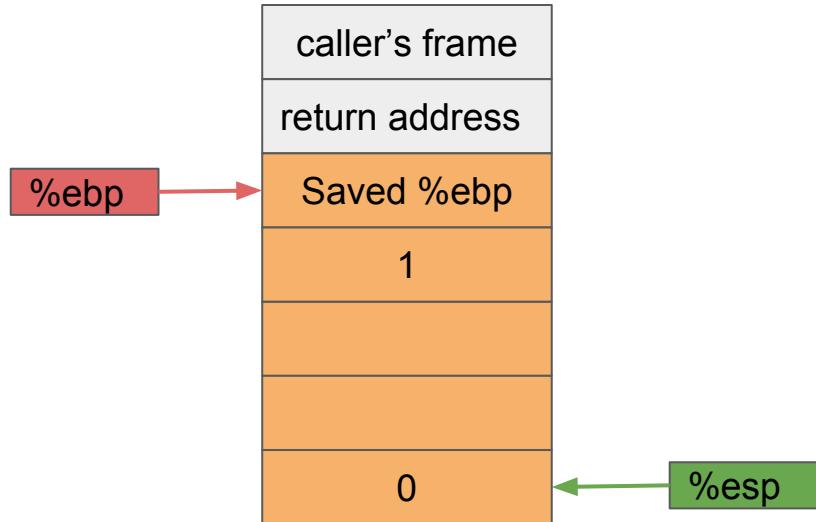
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

```
leave
```

```
ret
```



%eax

%edx

0

0

func:

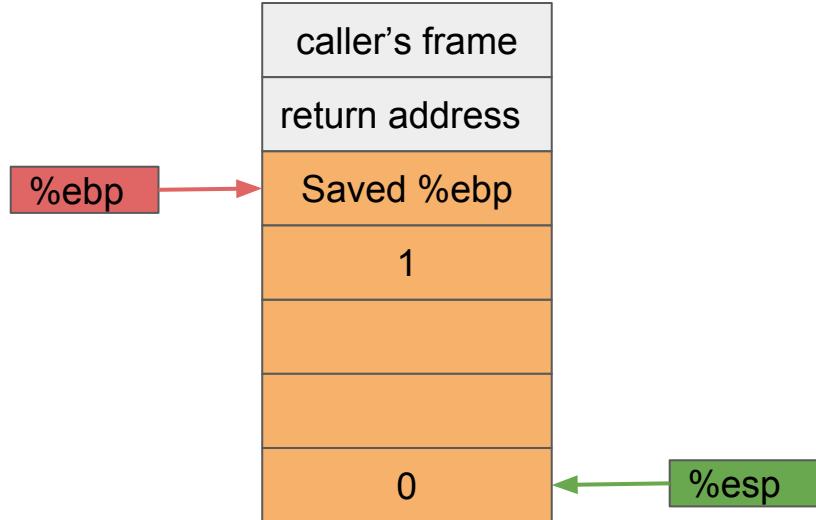
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx

0

0

func:

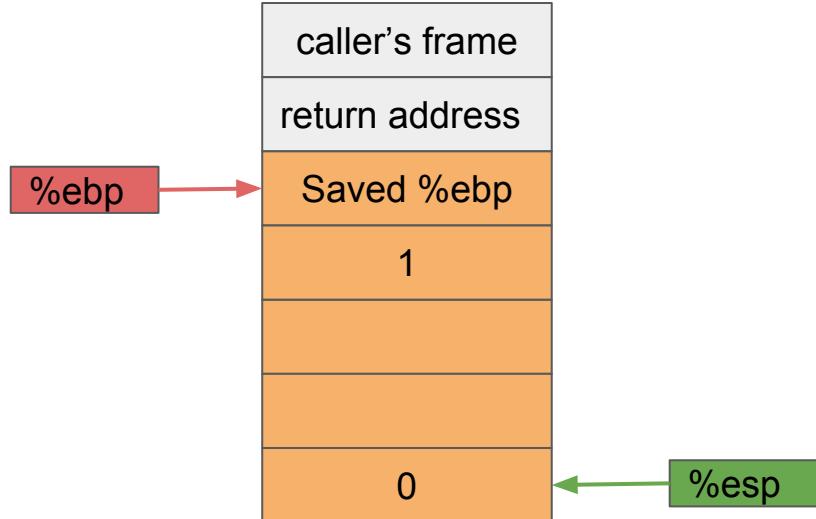
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

1

%edx

0

func:

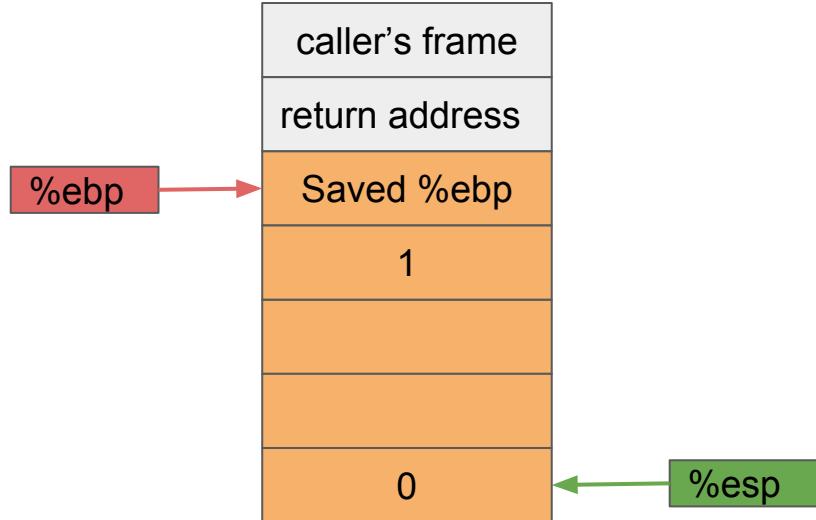
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

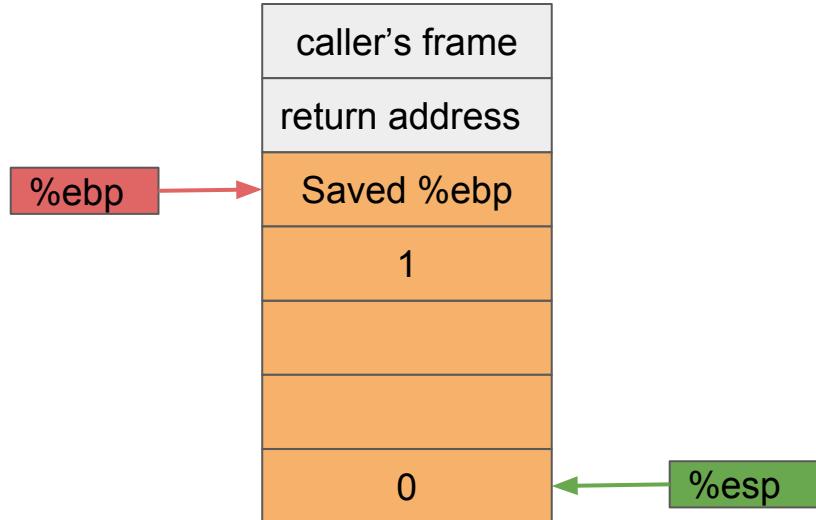
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

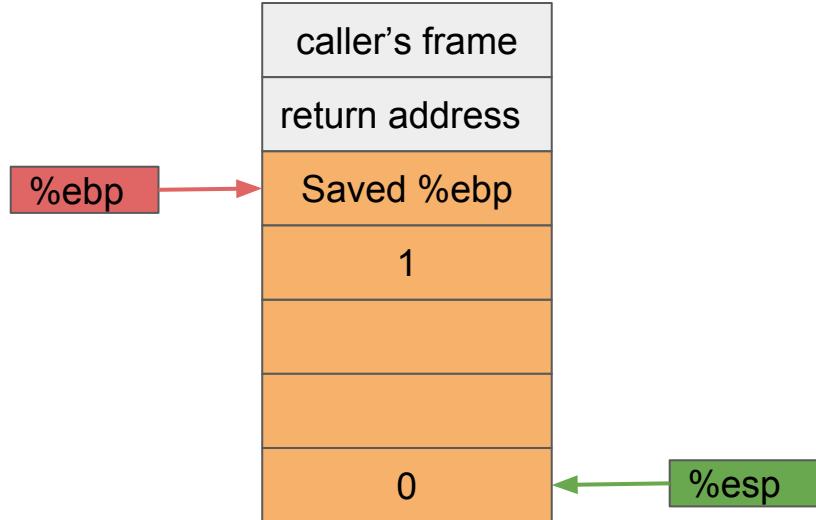
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax

%edx

1

1

func:

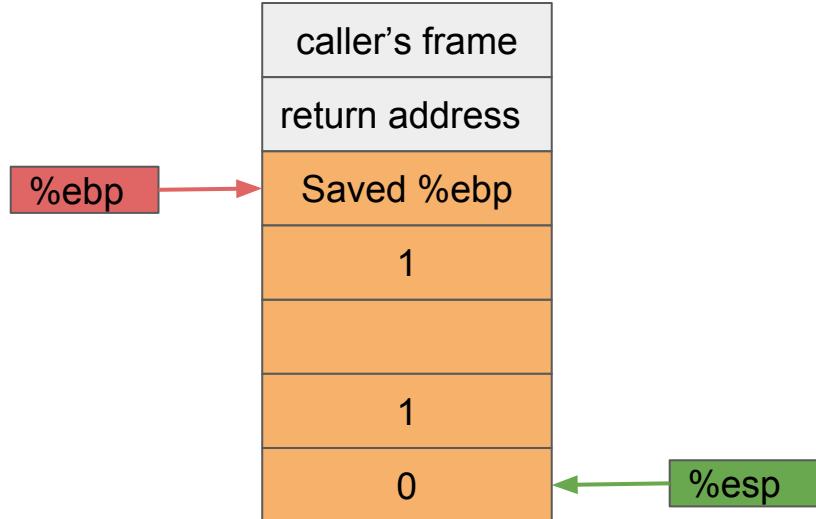
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

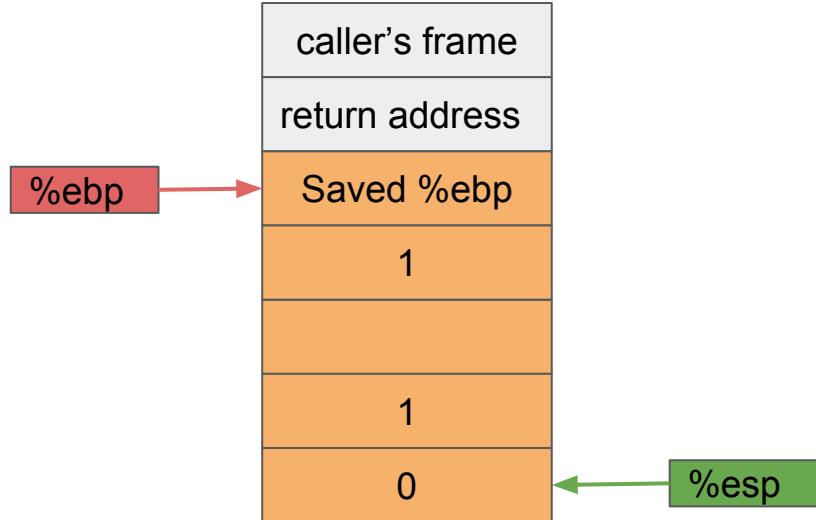
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

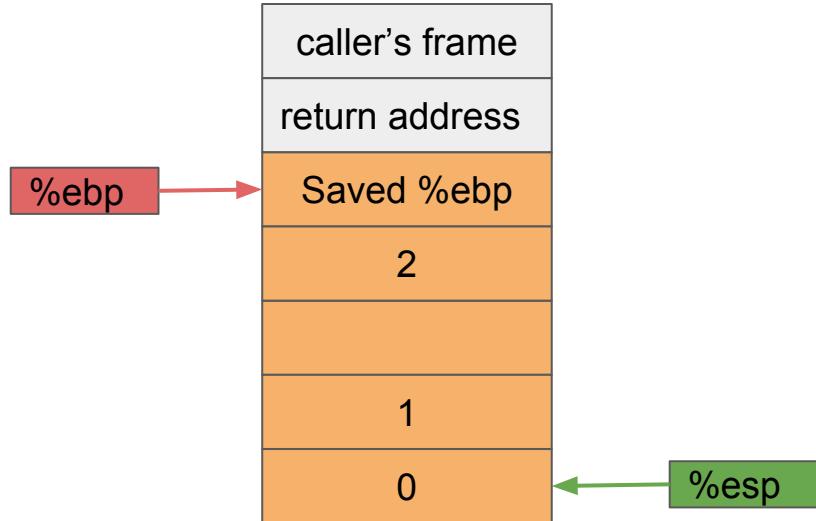
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



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)
```

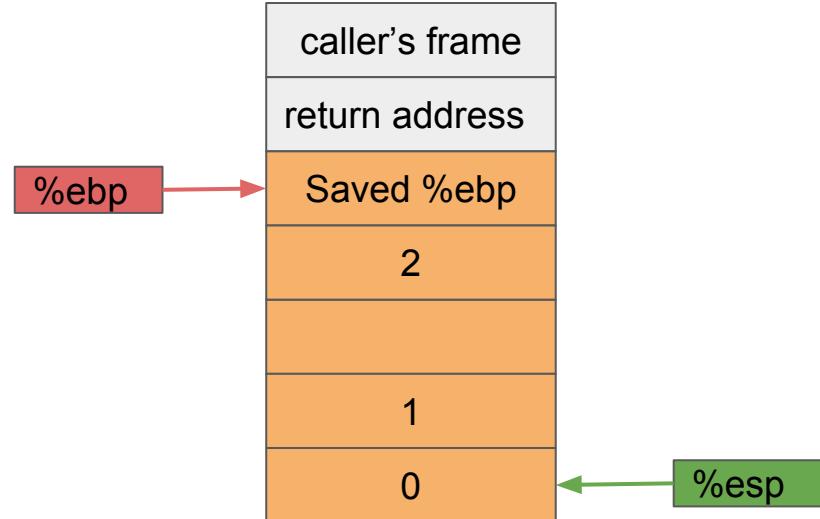
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax



%edx



func:

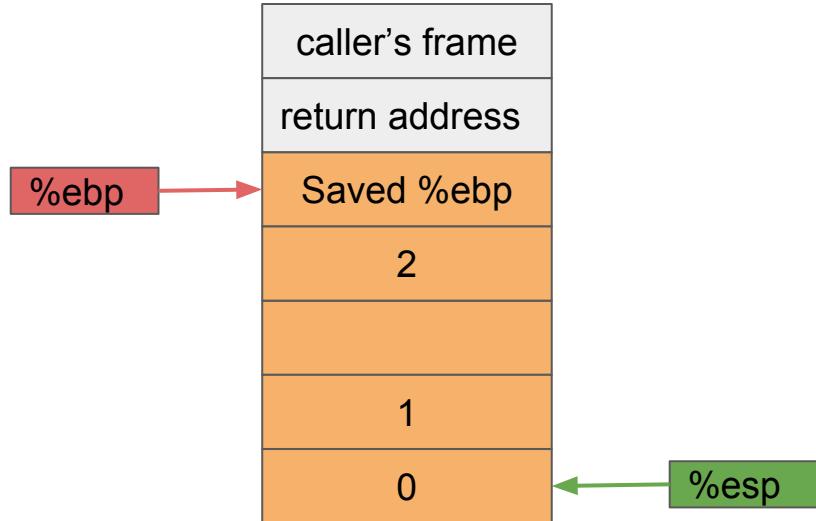
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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

%eax

%edx

1

1

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)
```

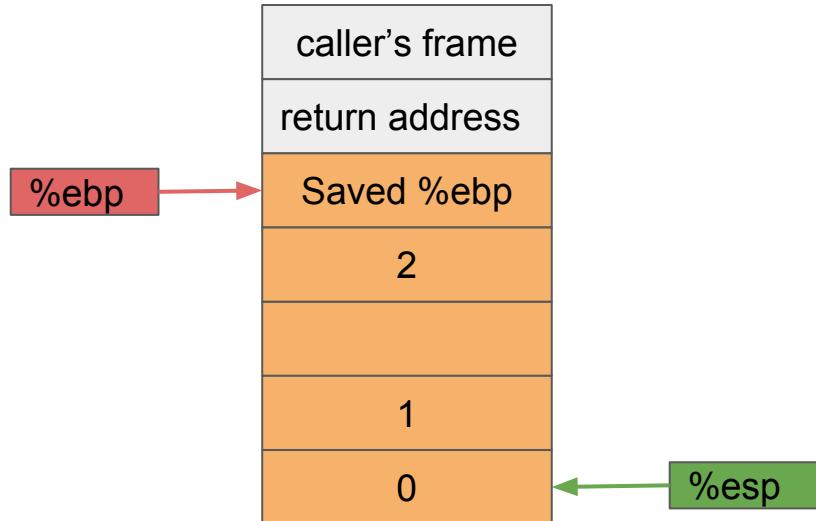
.L2:

```
cmpl $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax



%edx



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)
```

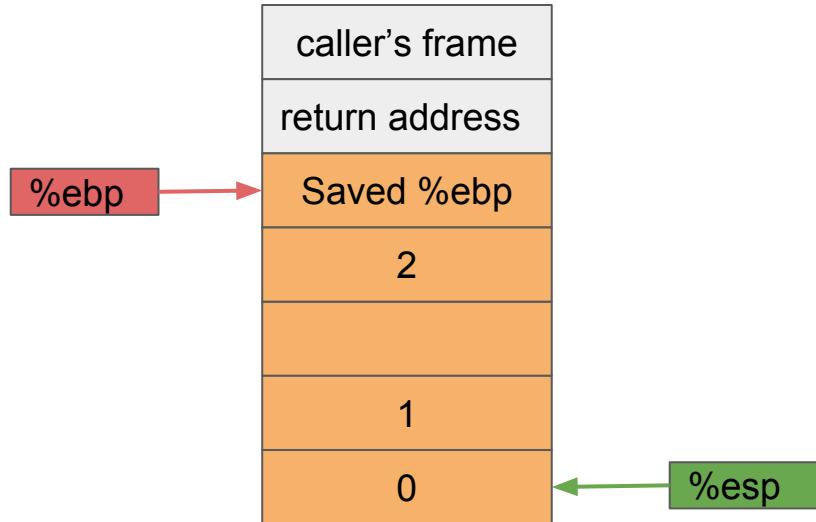
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

leave

ret



%eax



%edx



func:

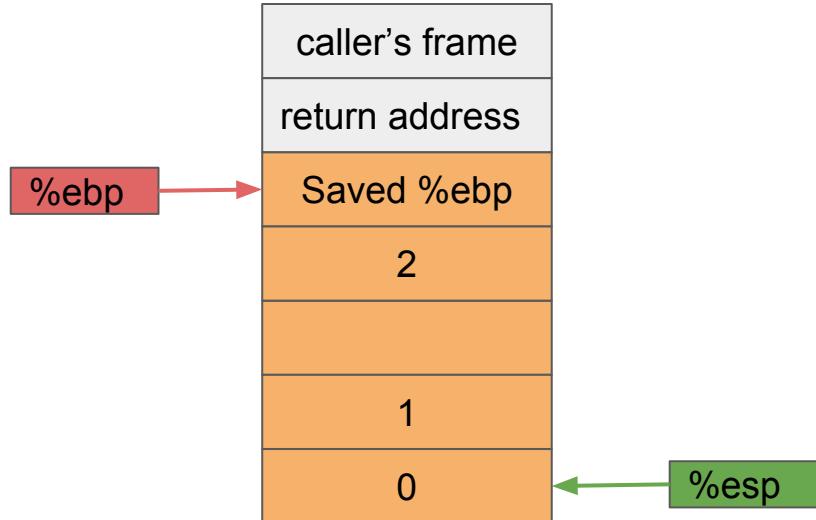
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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 $2 \leq 2$

%eax

1

%edx

1

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)
```

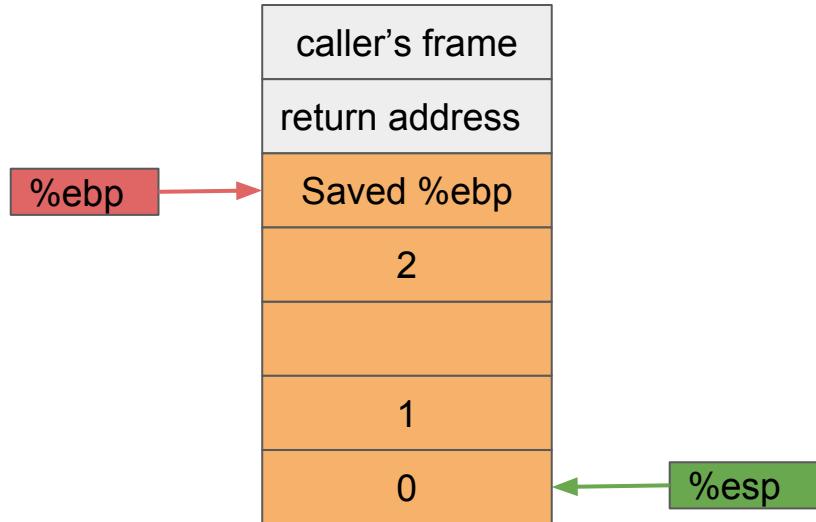
.L2:

```
cmpl $2, -4(%ebp)
```

jle .L3

leave

ret



%eax



%edx



func:

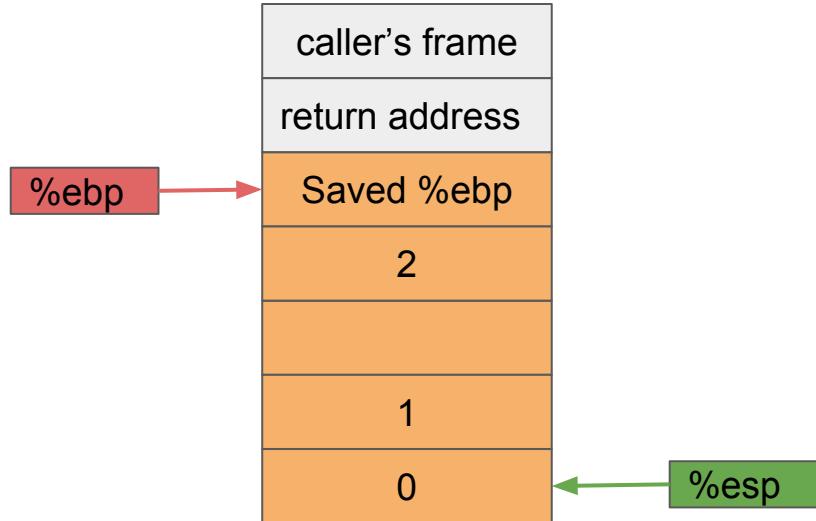
```
pushl %ebp  
movl %esp, %ebp  
subl $16, %esp  
movl $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

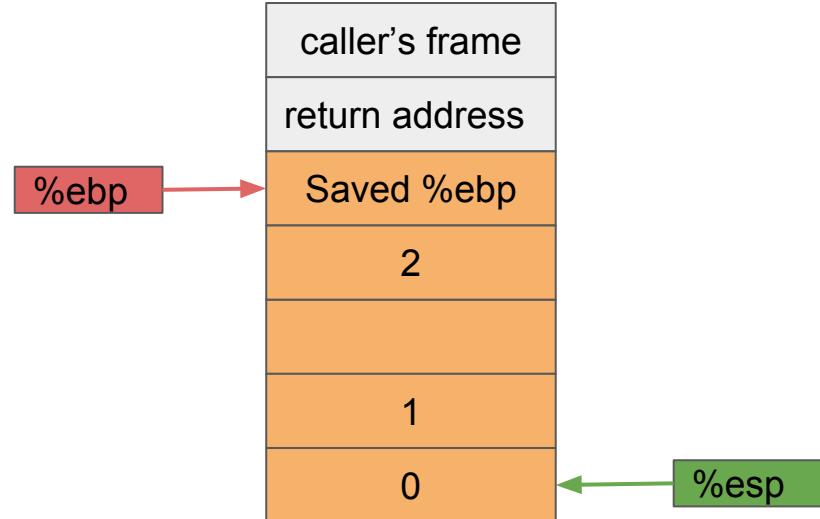
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

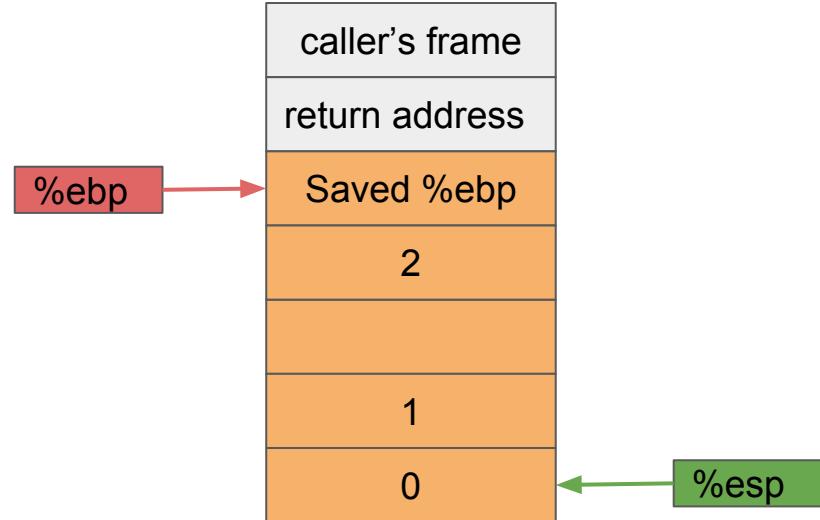
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

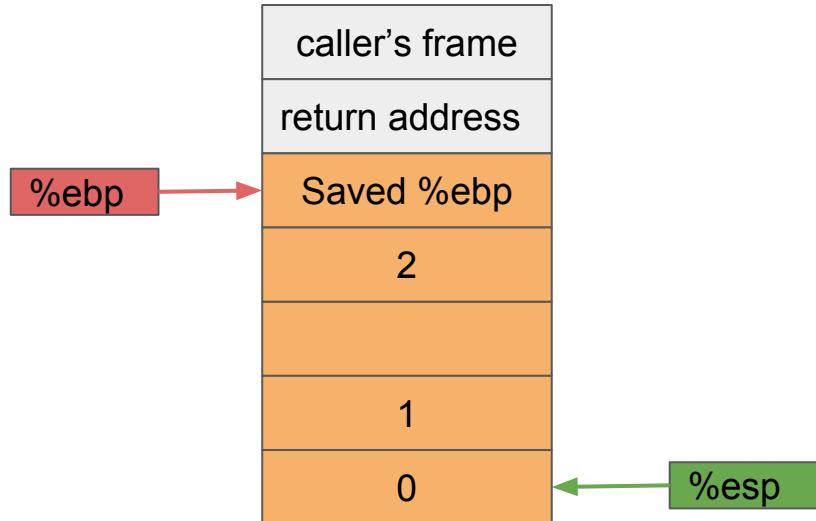
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

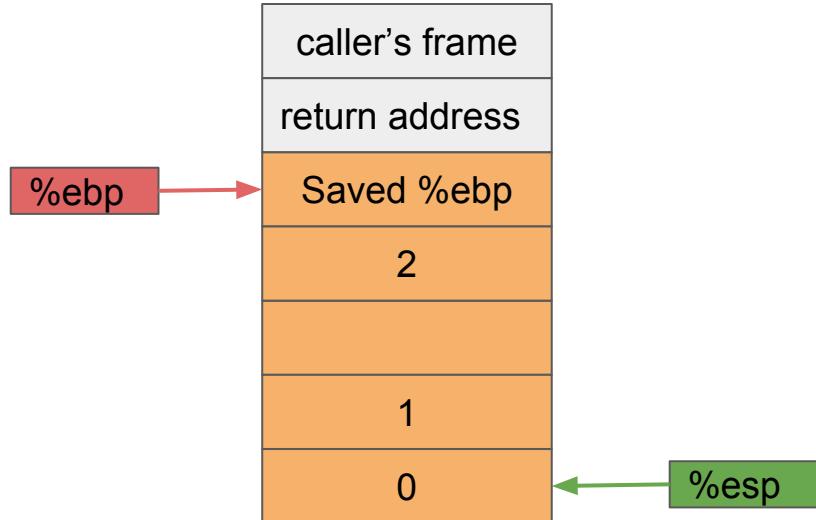
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

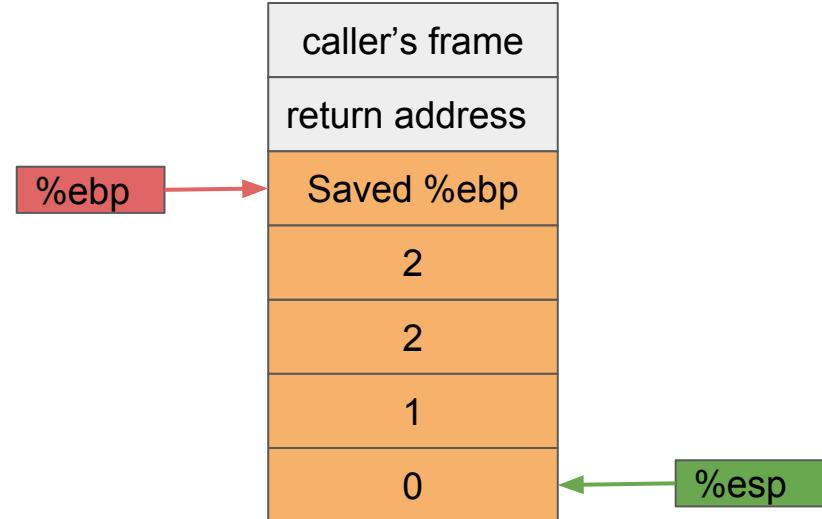
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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:

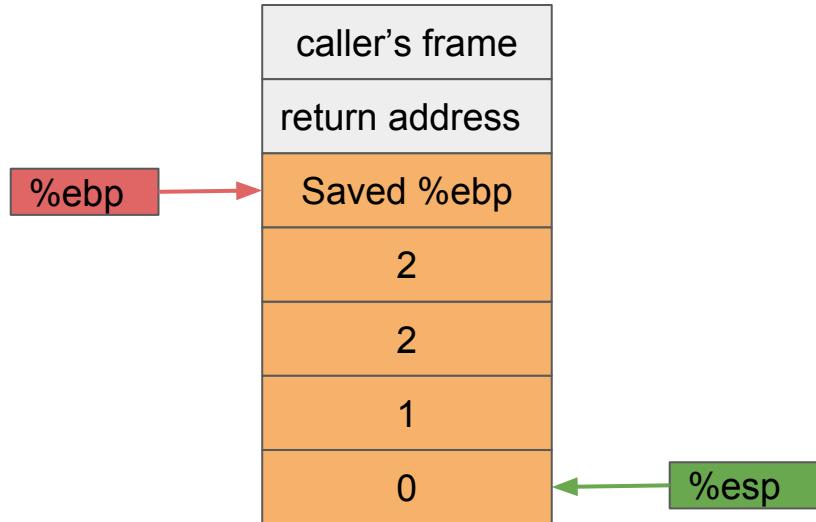
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



%eax



%edx



func:

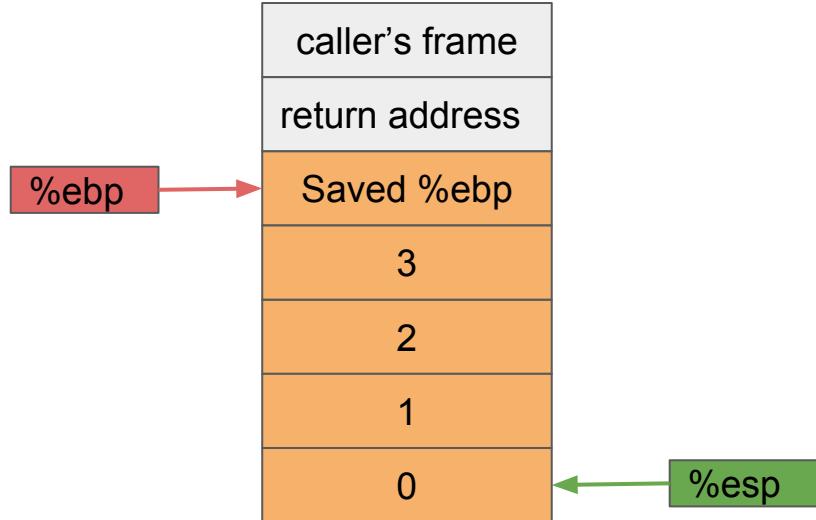
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.L3:

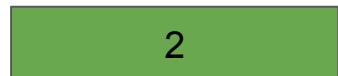
```
movl    -4(%ebp), %eax  
movl    -4(%ebp), %edx  
movl    %edx, -16(%ebp,%eax,4)  
addl    $1, -4(%ebp)
```

.L2:

```
cmpl    $2, -4(%ebp)  
jle .L3  
leave  
ret
```



%eax



%edx



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)
```

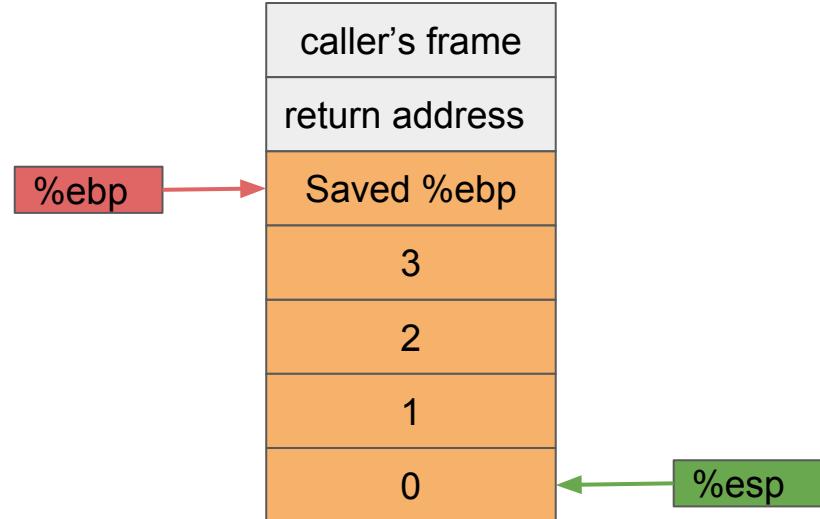
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax

2

%edx

2

func:

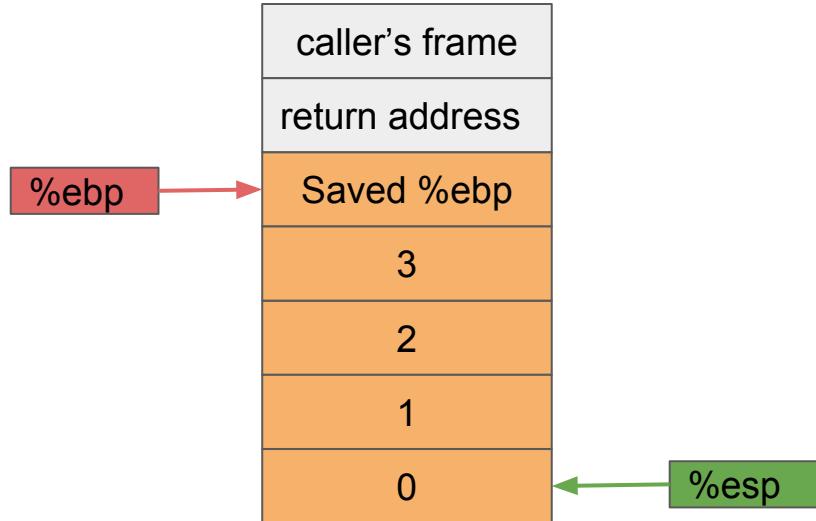
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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
```



Compare 3 and 2

%eax

2

%edx

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)
```

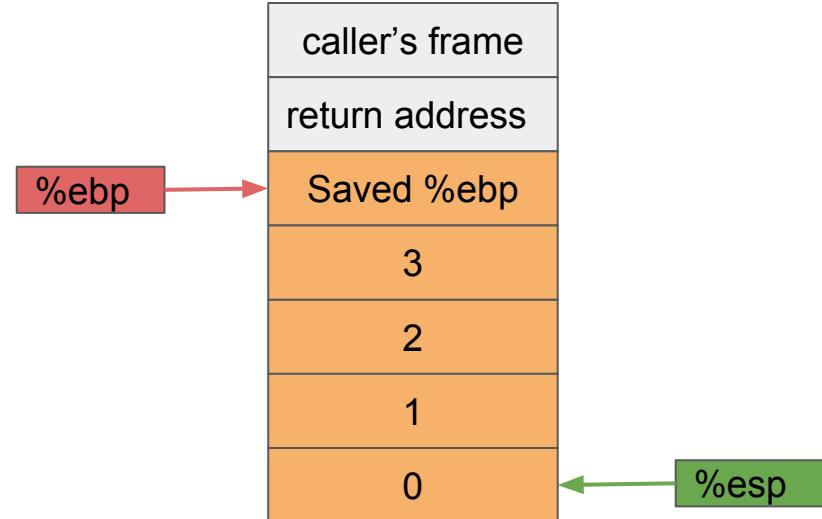
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



%eax

2

%edx

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)
```

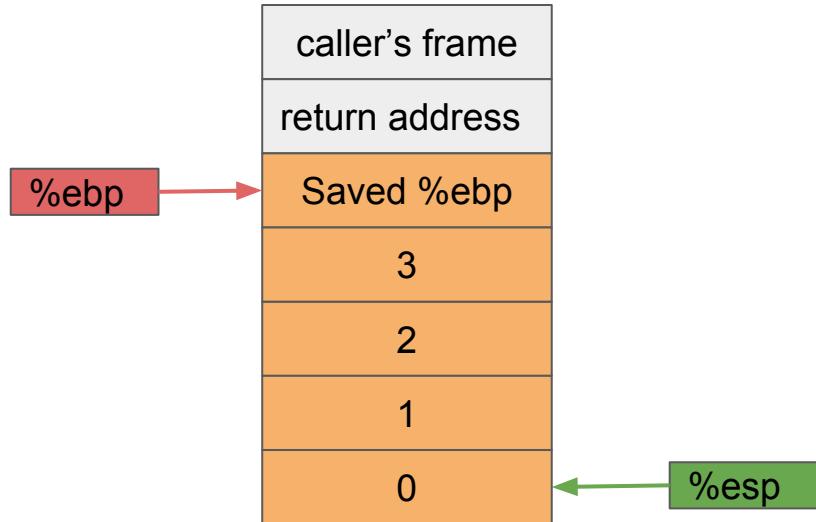
.L2:

```
cmpl    $2, -4(%ebp)
```

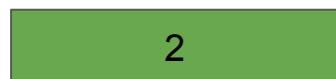
jle .L3

leave

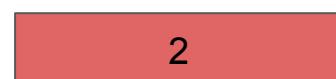
ret



%eax



%edx



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)
```

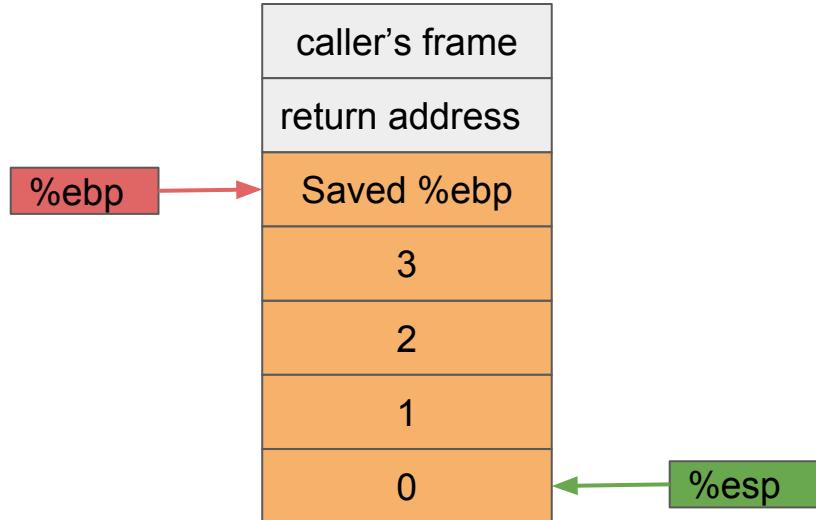
.L2:

```
cmpl $2, -4(%ebp)
```

```
jle .L3
```

```
leave
```

```
ret
```



Jump if $3 \leq 2$

%eax

2

%edx

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)
```

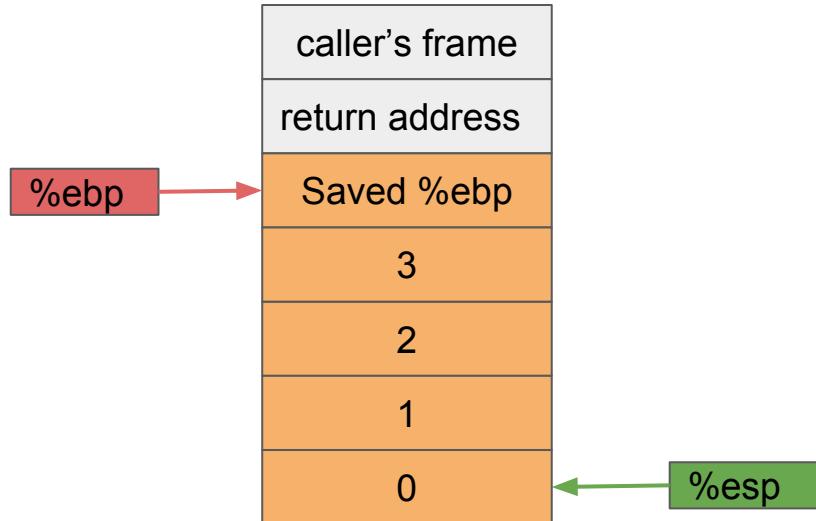
.L2:

```
cmpl    $2, -4(%ebp)
```

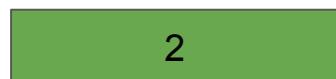
jle .L3

leave

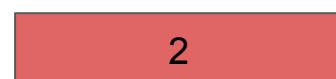
ret



%eax



%edx



func:

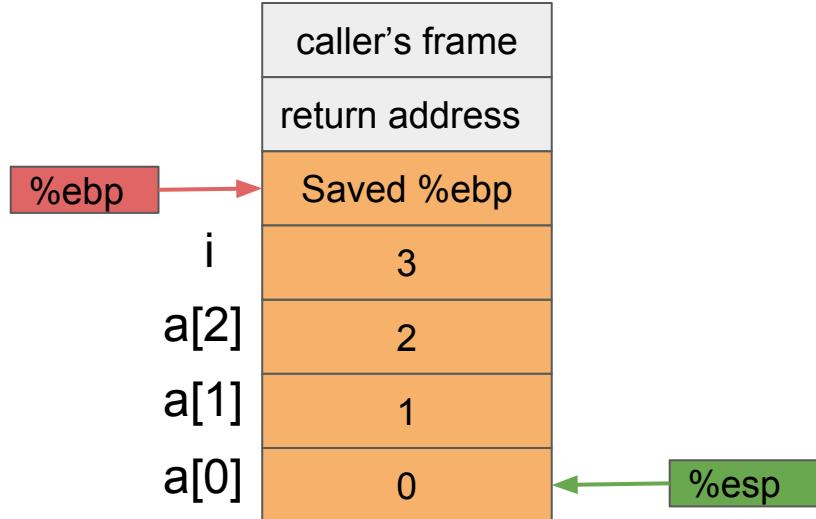
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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)
```

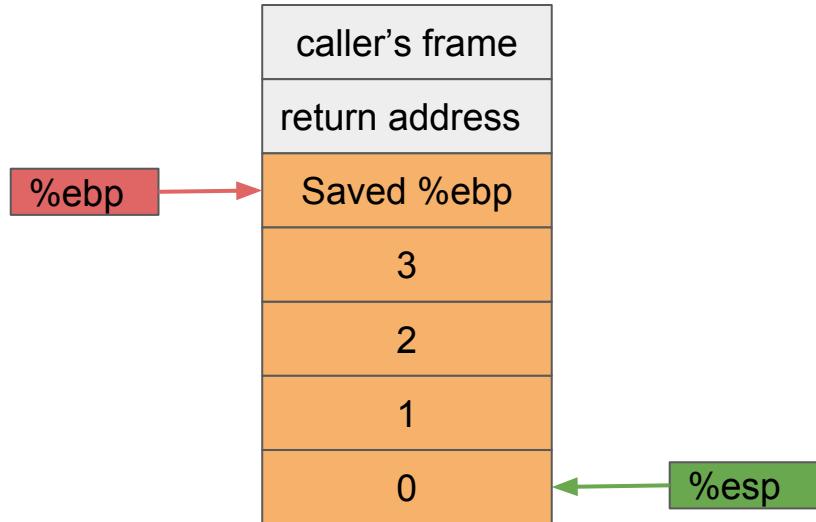
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

leave

ret



%eax

2

%edx

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)
```

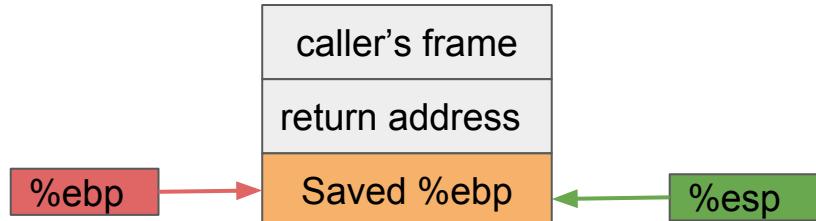
.L2:

```
cmpl    $2, -4(%ebp)
```

```
jle .L3
```

leave

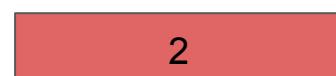
ret



%eax



%edx



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



%eax

%edx

2

2

func:

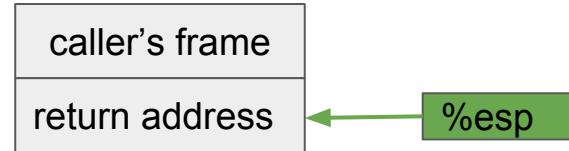
```
pushl    %ebp  
movl    %esp, %ebp  
subl    $16, %esp  
movl    $0, -4(%ebp)  
jmp .L2
```

.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

caller's frame

%esp

2

%eax

2

%edx