Structures in Assembly

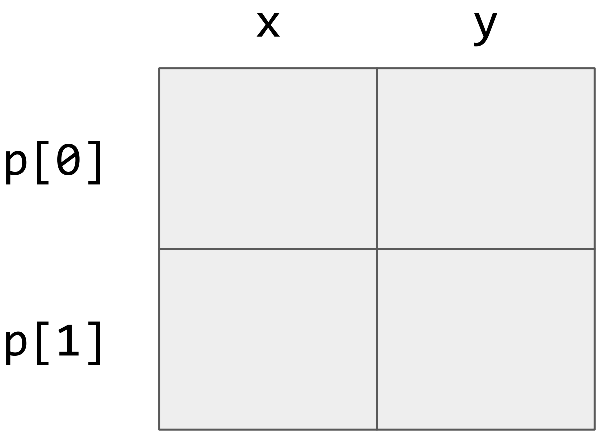
Adalbert **Gerald** Soosai Raj

C

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

	x	y
p[0]		
p[1]		

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

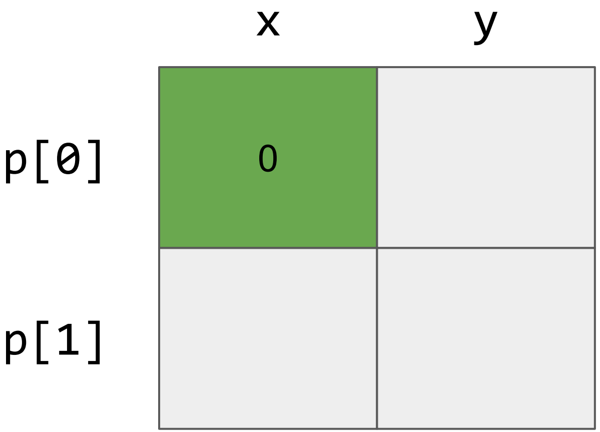


	x	y
p[0]		
p[1]		

```c
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

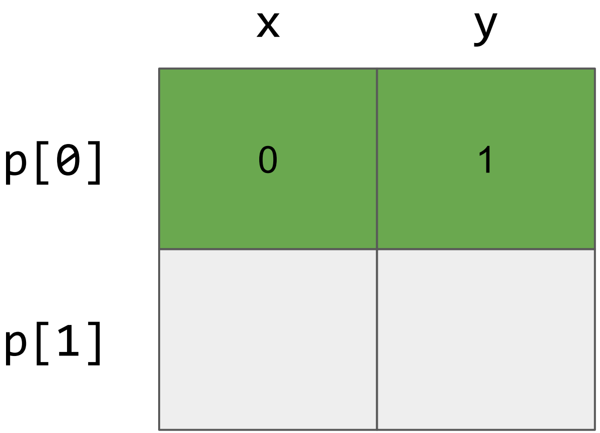
	x	y
p[0]	0	
p[1]		

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```



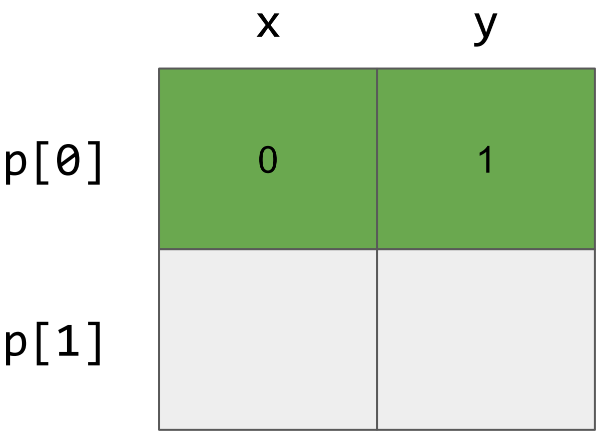
	x	y
p[0]	0	
p[1]		

```c
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```



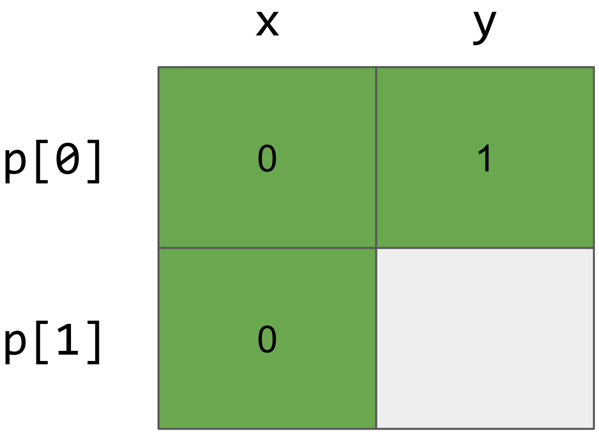
	x	y
p[0]	0	1
p[1]		

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```



	x	y
p[0]	0	1
p[1]		

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

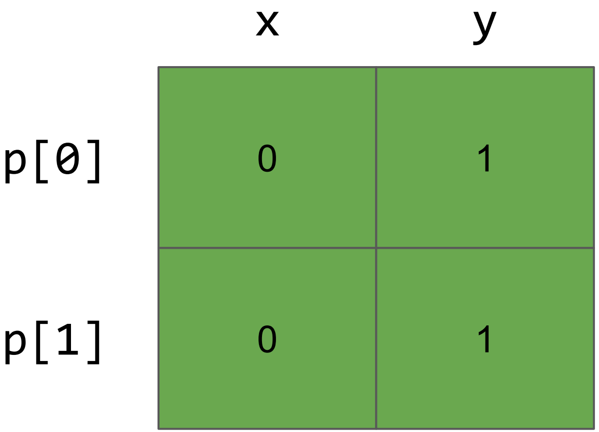


	x	y
p[0]	0	1
p[1]	0	

```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

	x	y
p[0]	0	1
p[1]	0	

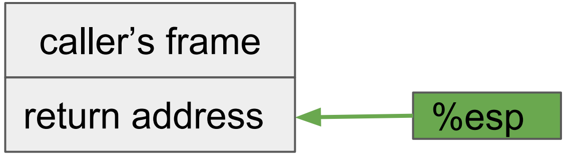
```
#define N 2

int func()
{
    struct point {
        int x;
        int y;
    };

    struct point p[N];
    int i;

    for (i = 0; i < N; ++i)
    {
        p[i].x = 0;
        p[i].y = 1;
    }
}
```

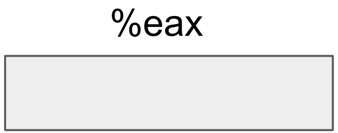


	x	y
p[0]	0	1
p[1]	0	1

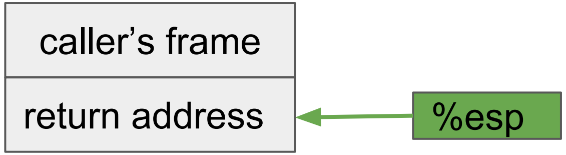
Assembly

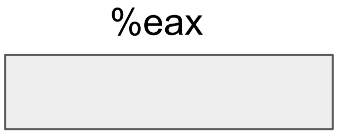
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



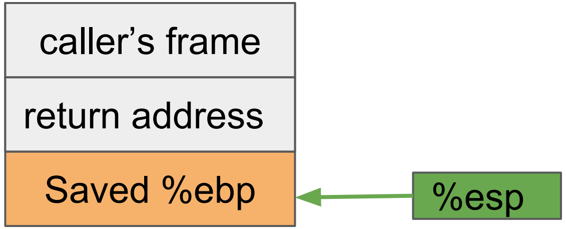


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```





```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

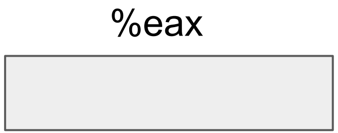


caller’s frame

return address

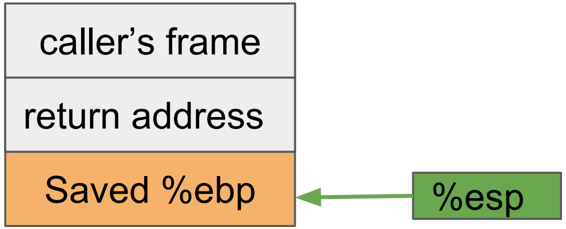
Saved %ebp

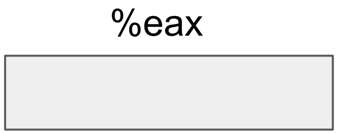
%esp



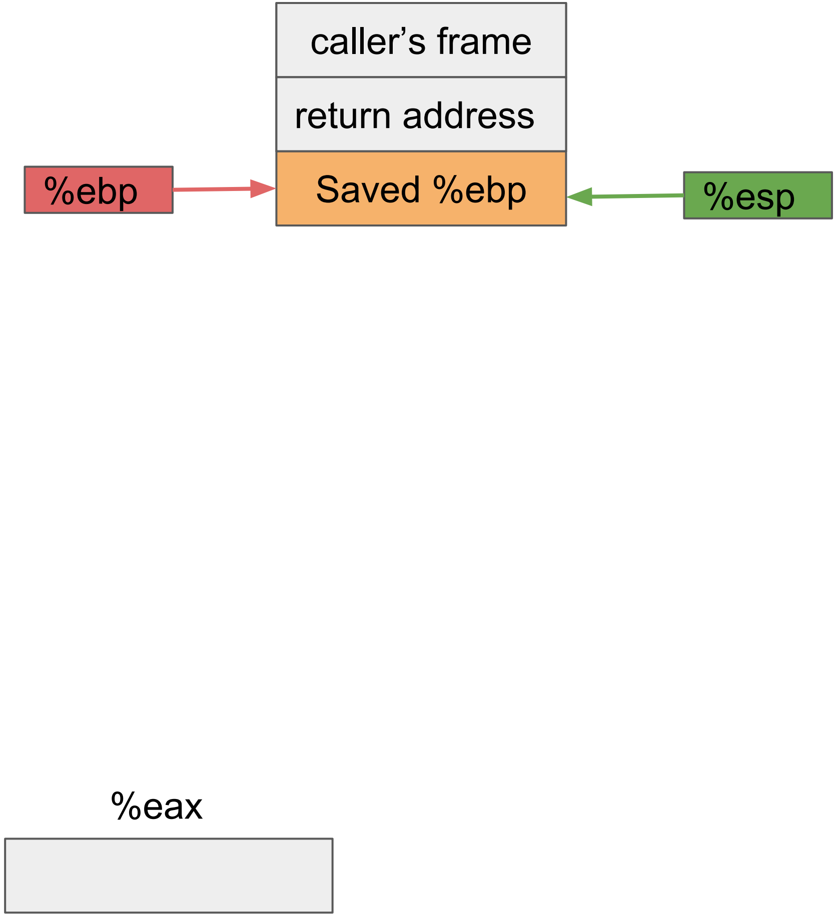
%eax

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

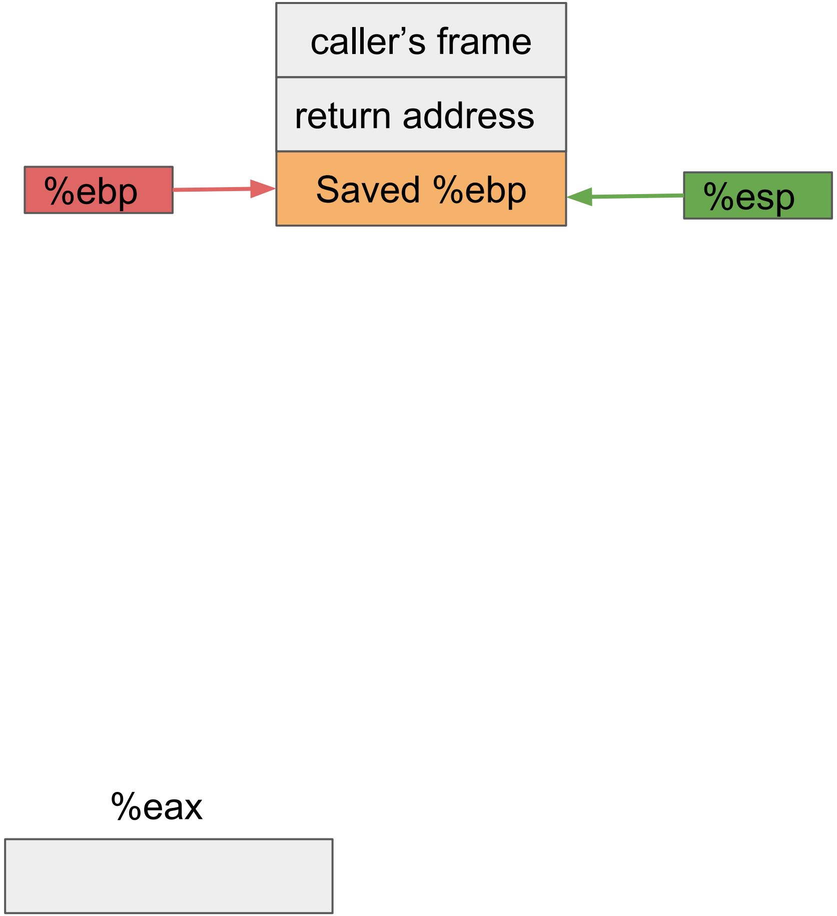




```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

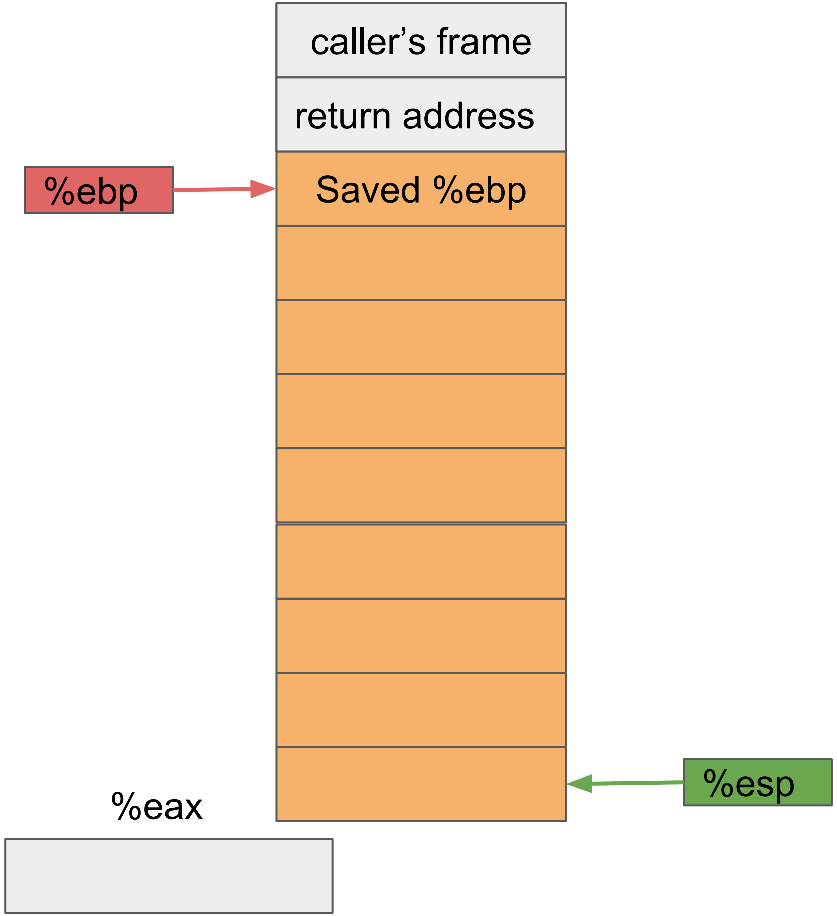
%ebp

Saved %ebp

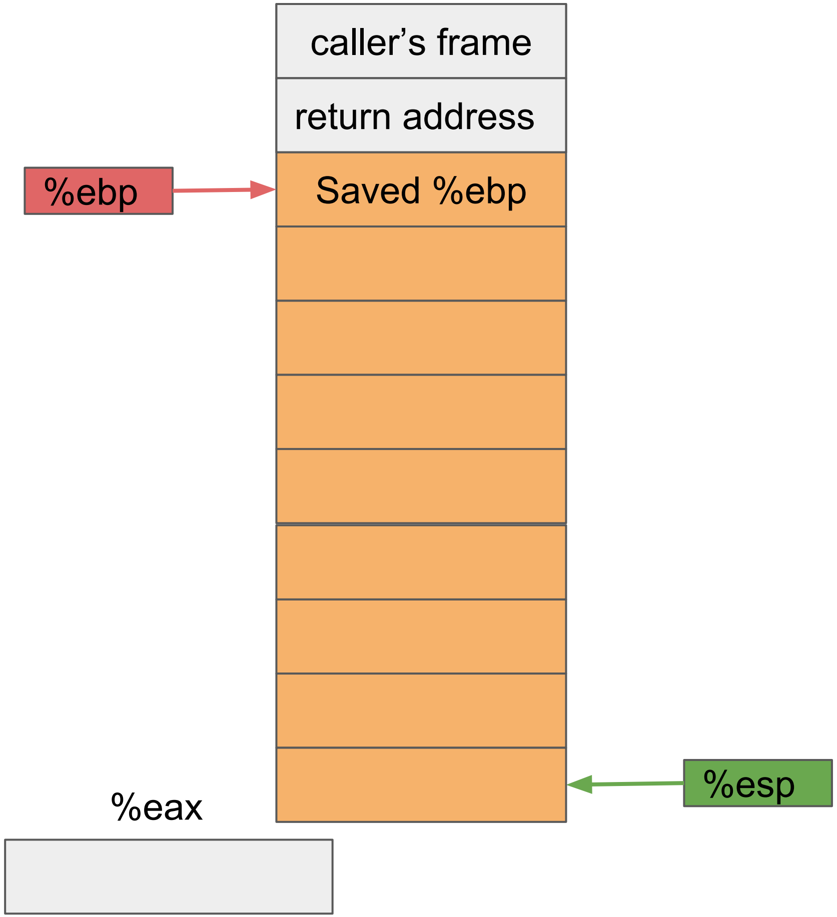
%esp

%eax

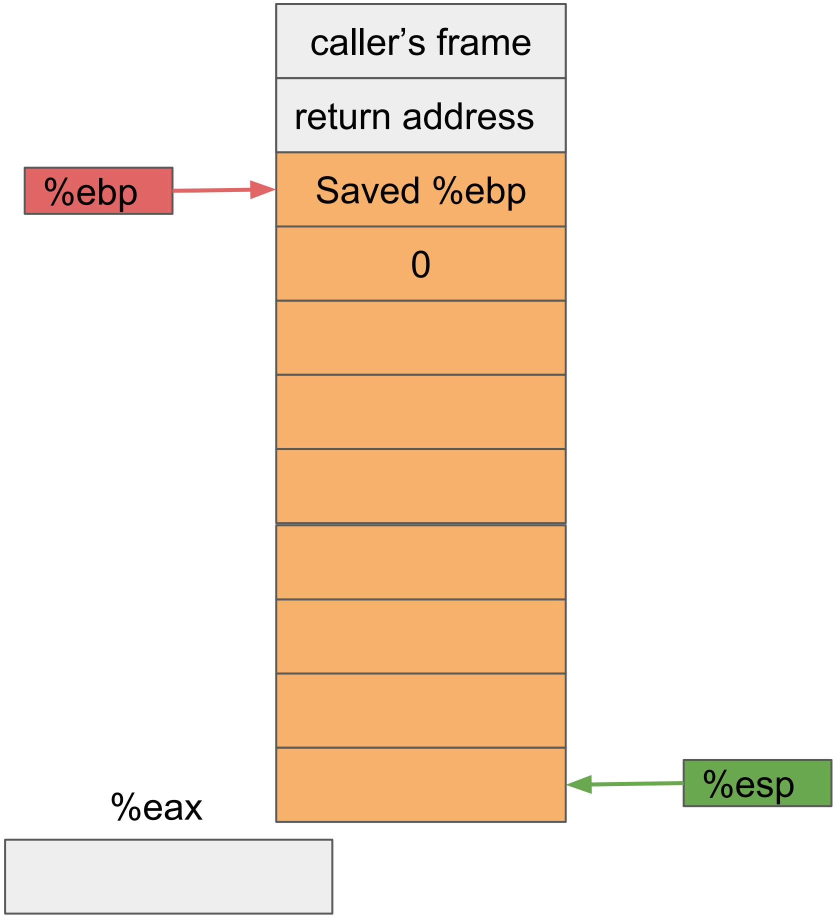
```
func:
    pushl    %ebp
    movl     %esp, %ebp
    subl     $32, %esp
    movl     $0, -4(%ebp)
    jmp .L2
.L3:
    movl     -4(%ebp), %eax
    movl     $0, -20(%ebp,%eax,8)
    movl     -4(%ebp), %eax
    movl     $1, -16(%ebp,%eax,8)
    addl     $1, -4(%ebp)
.L2:
    cmpl     $1, -4(%ebp)
    jle .L3
    leave
    ret
```



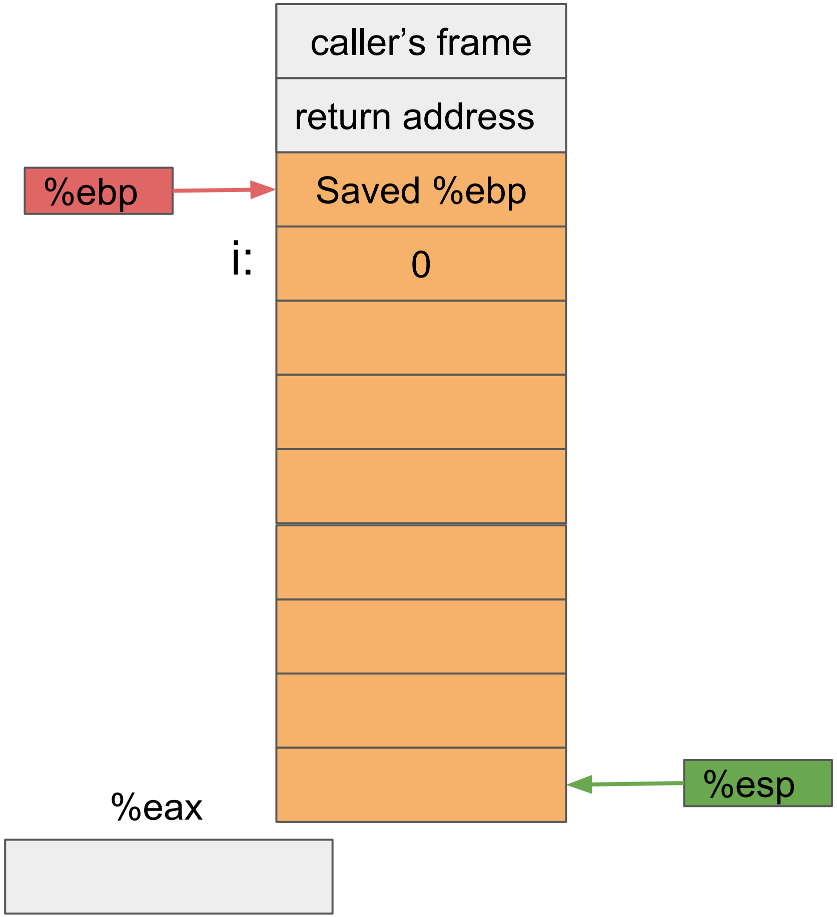
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



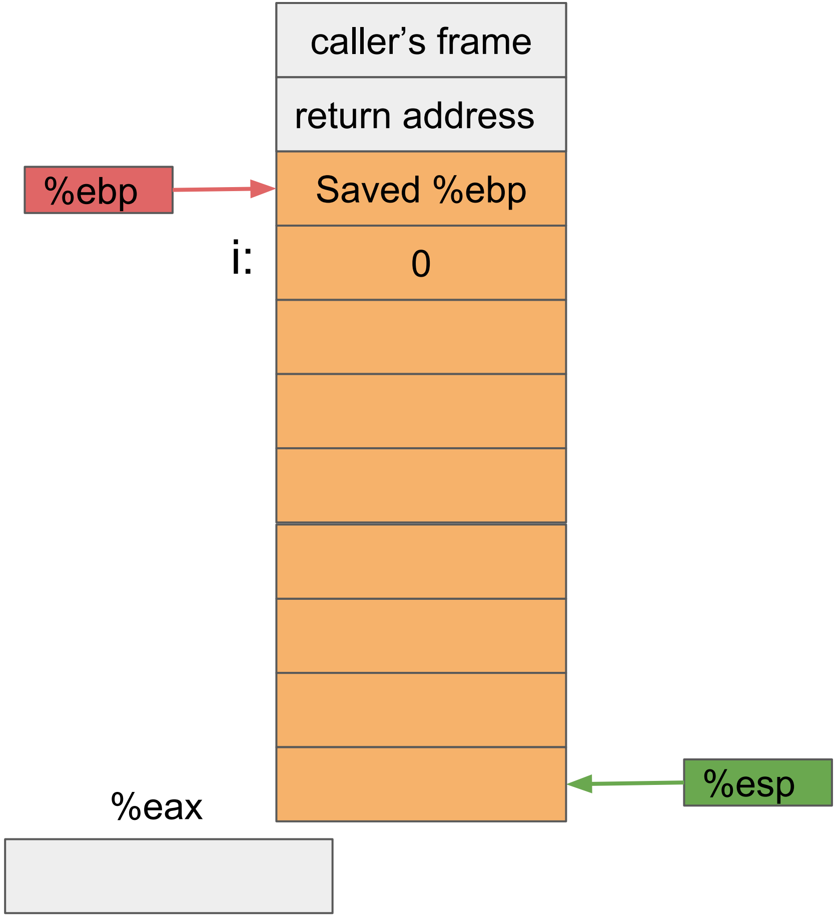
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



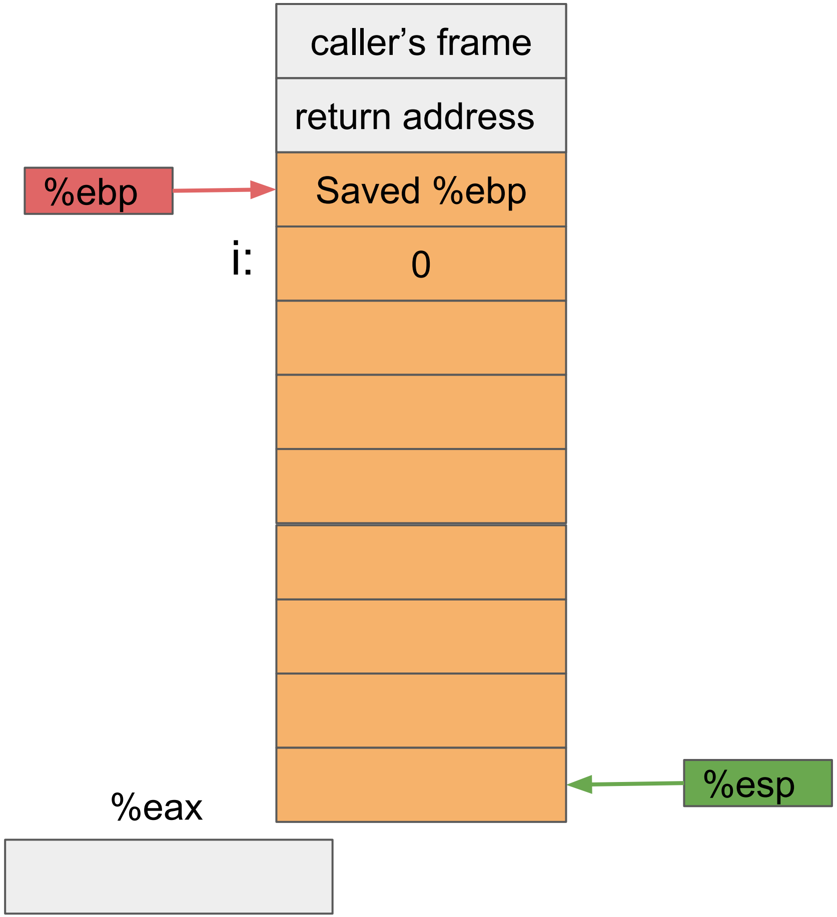
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



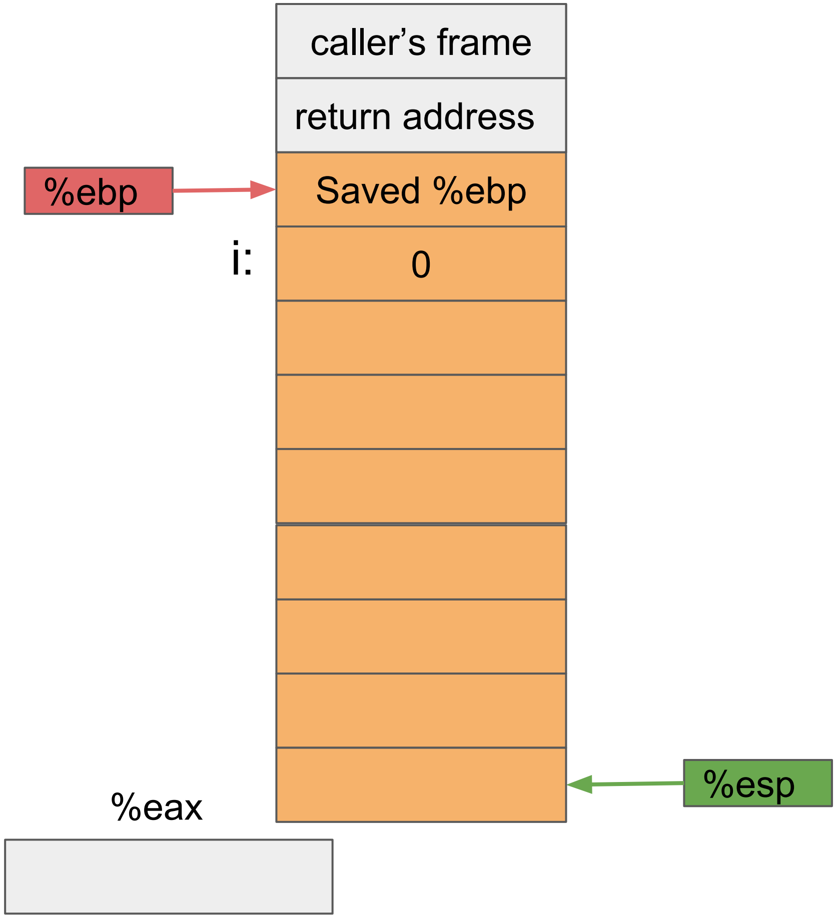
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

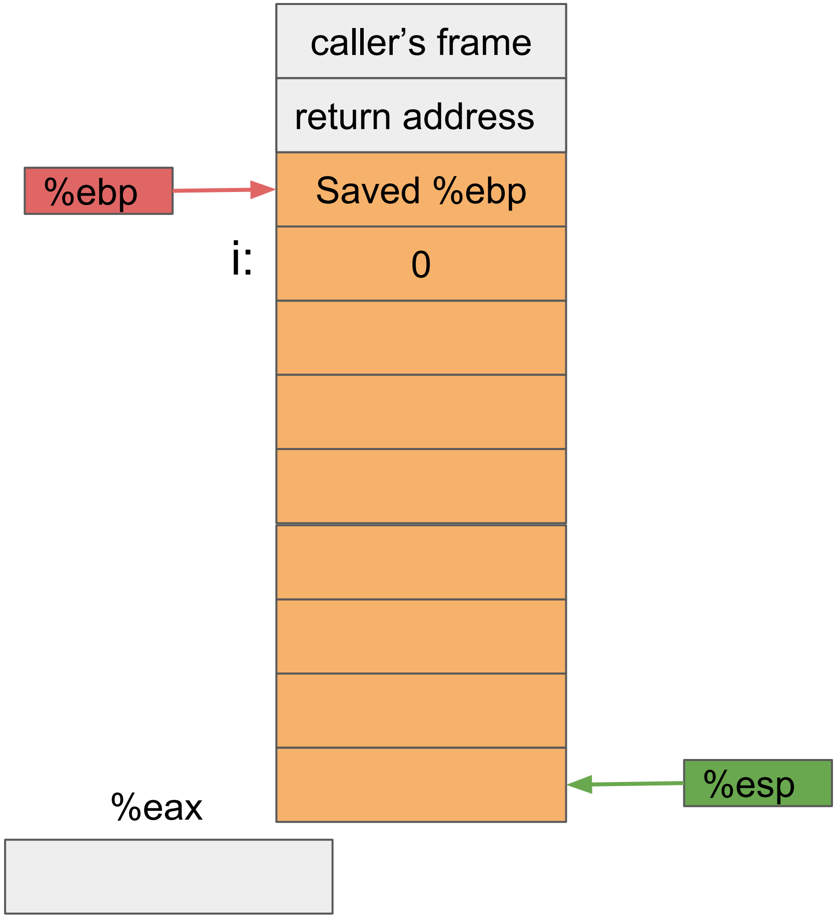


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

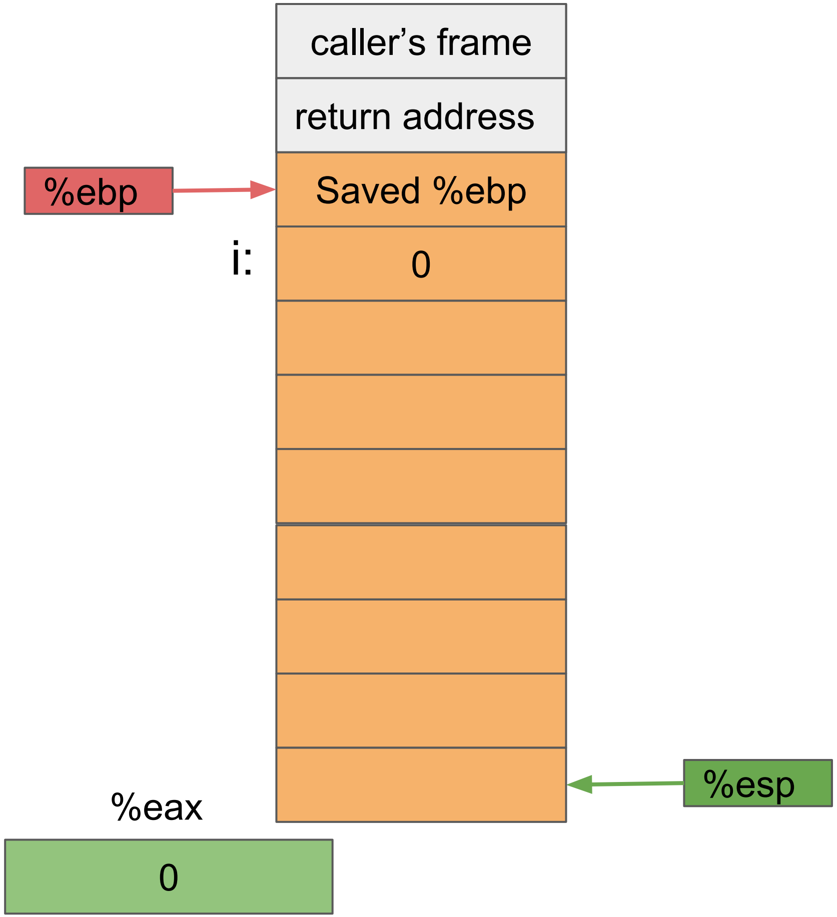


Jump if 0 <= 1

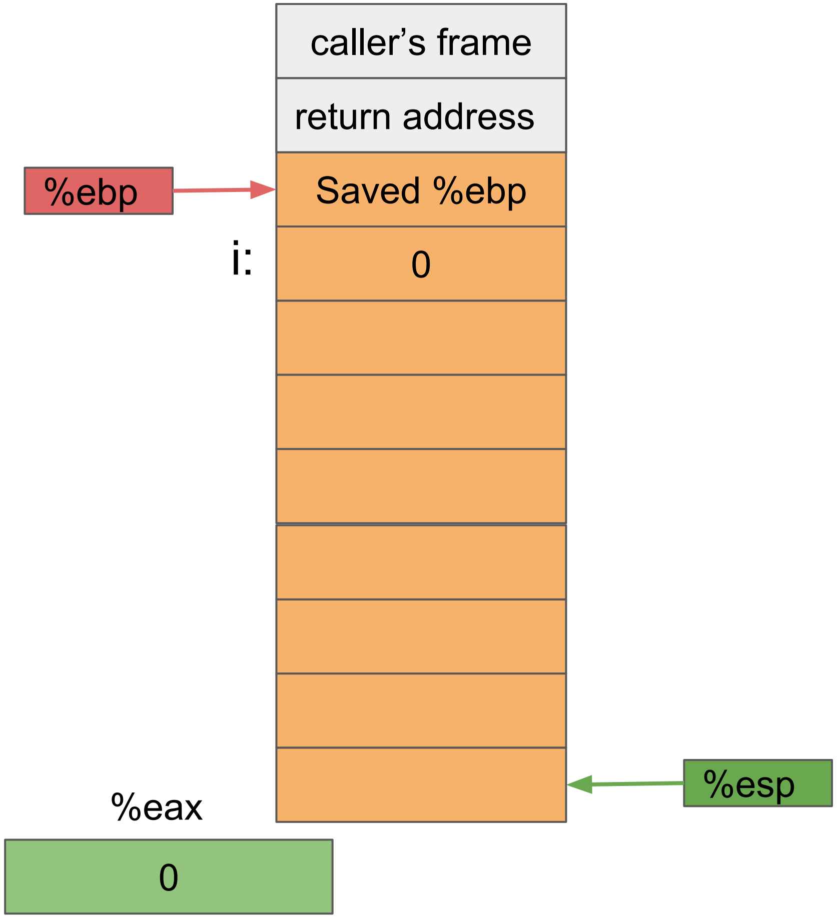
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



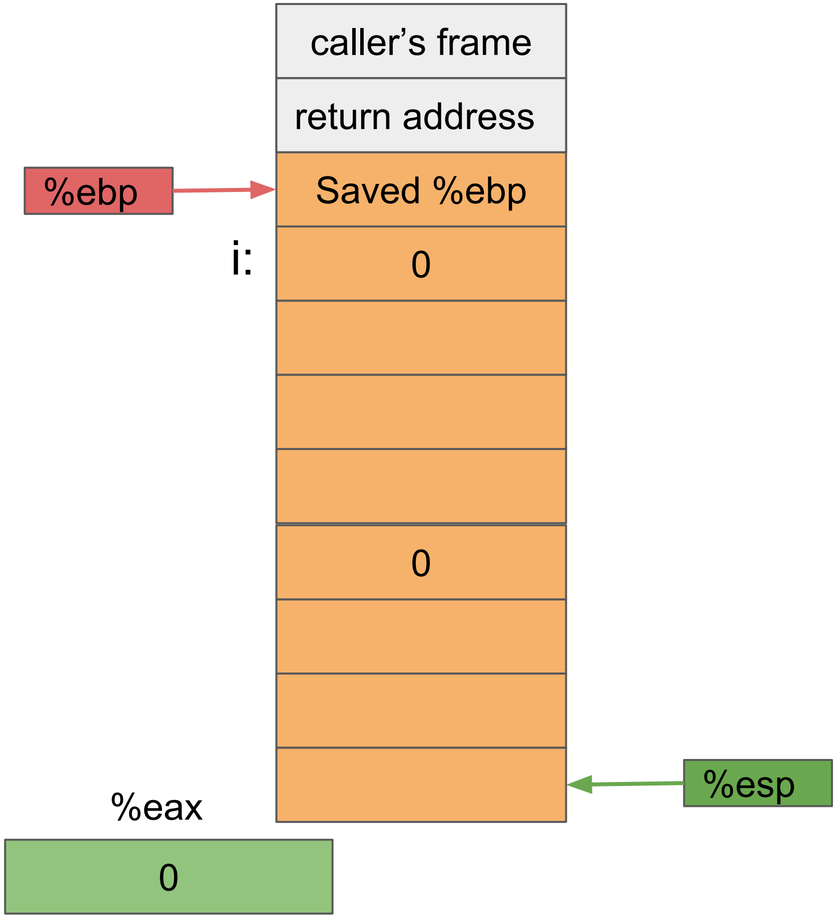
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



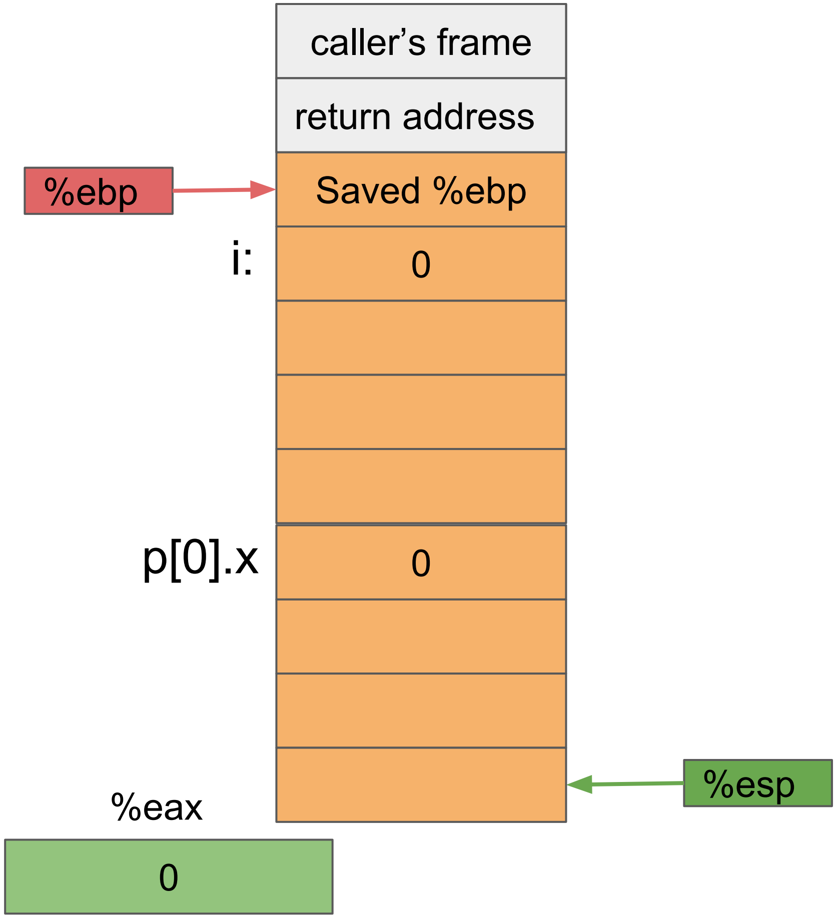
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp

Saved %ebp

i:

0

p[0].x

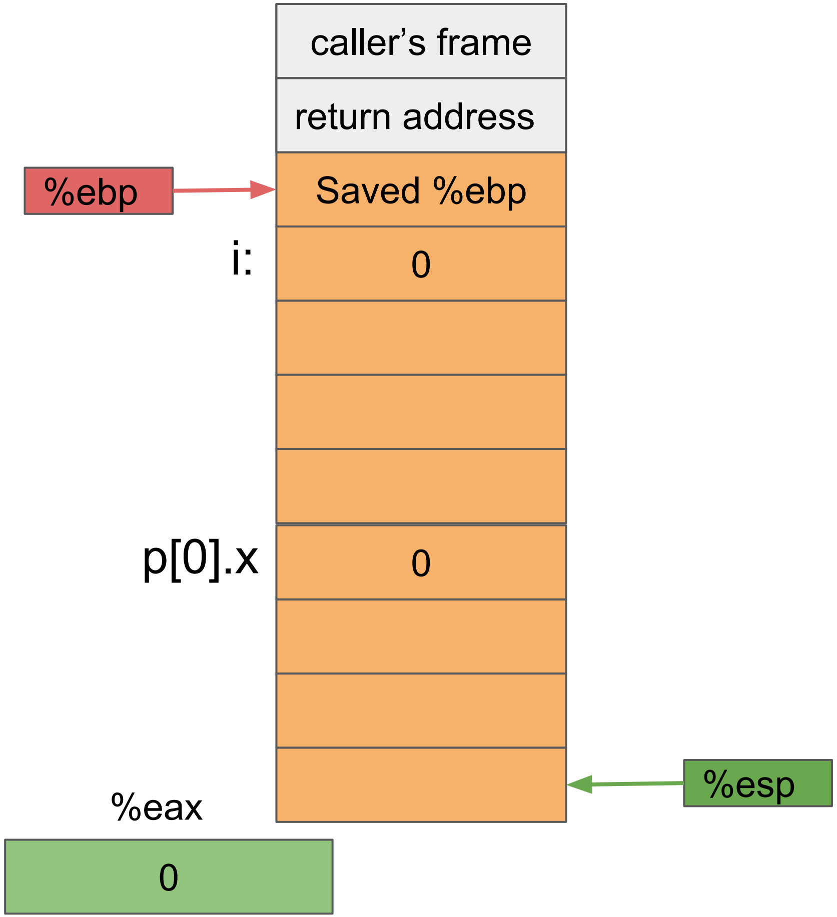
0

%esp

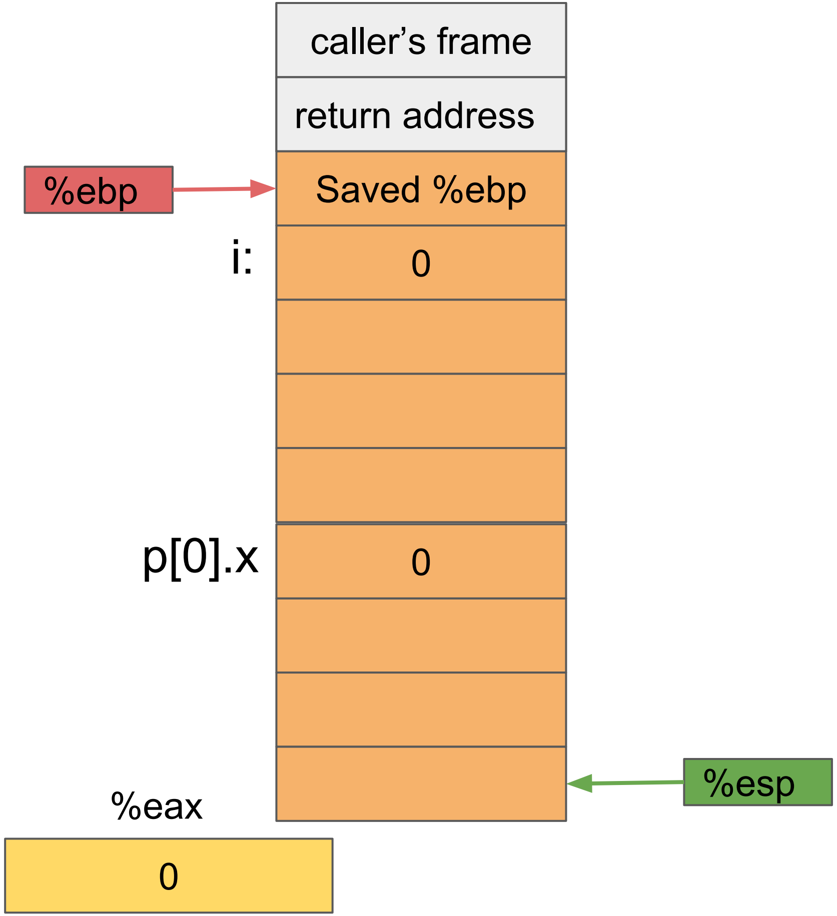
%eax

0

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp

Saved %ebp

i:

0

p[0].x

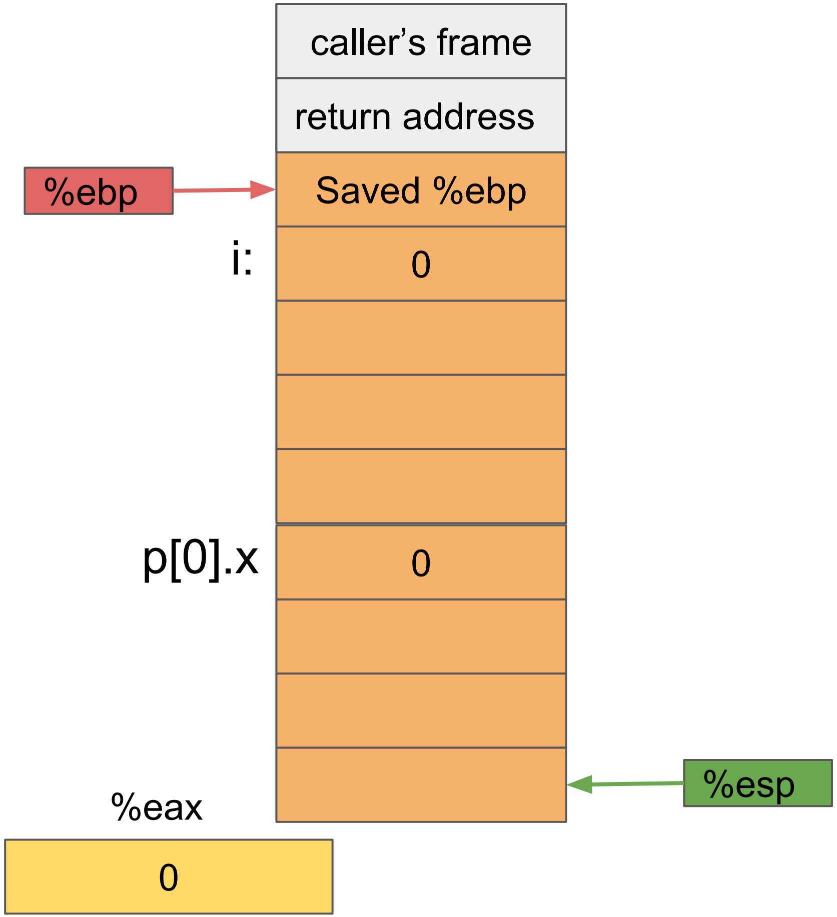
0

%esp

%eax

0

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp

Saved %ebp

i:

0

p[0].x

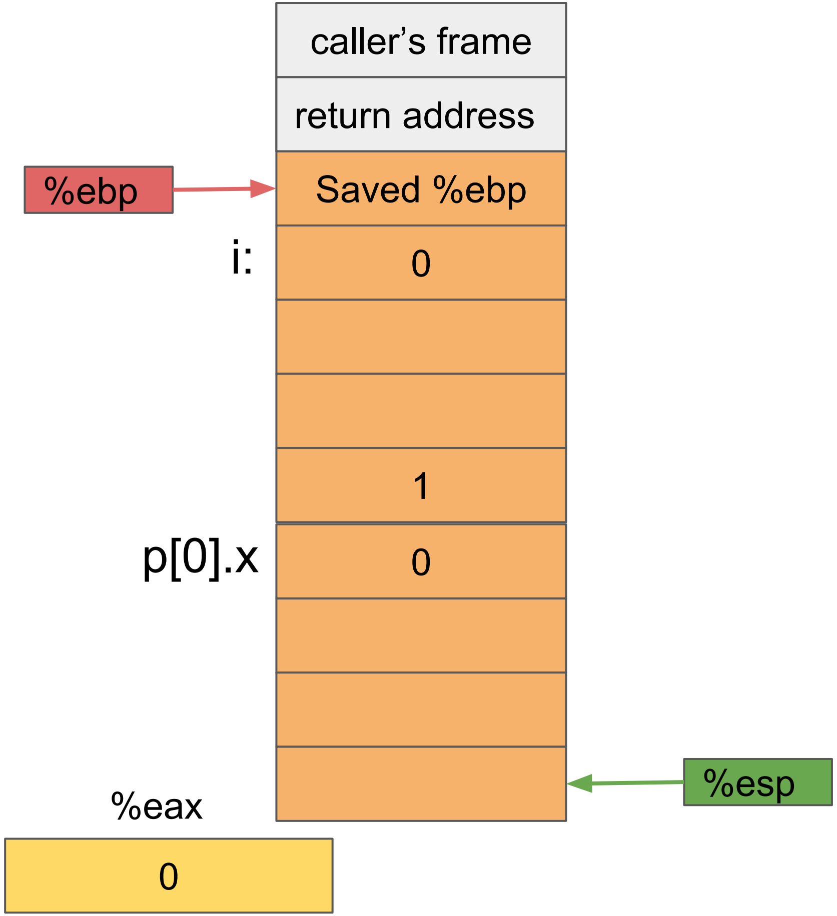
0

%esp

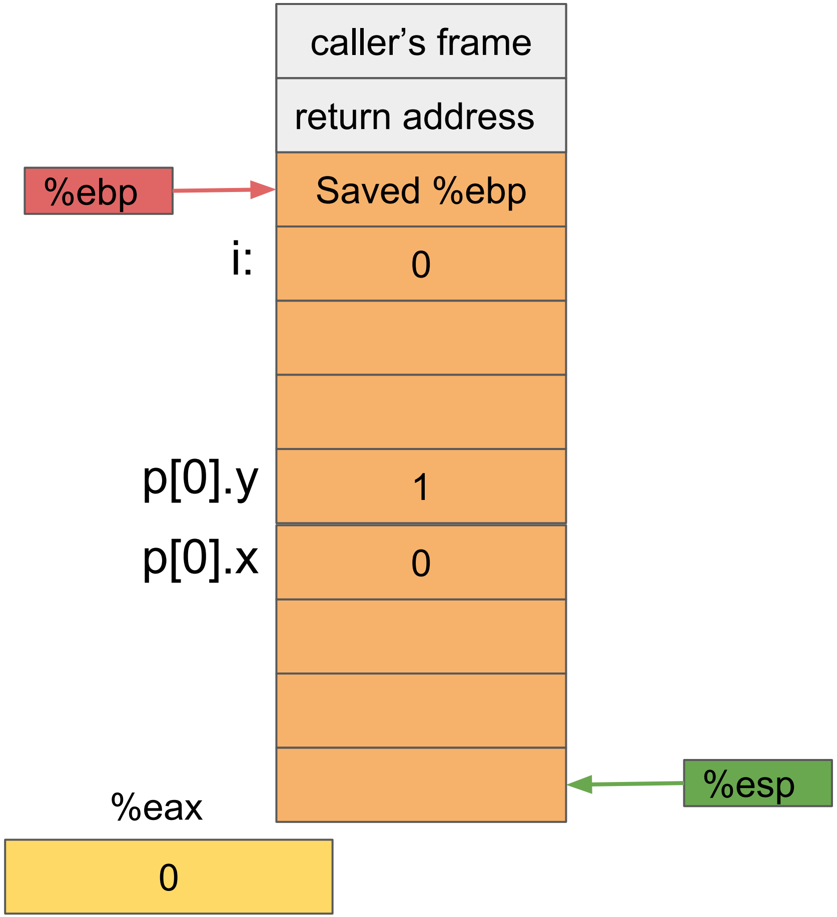
%eax

0

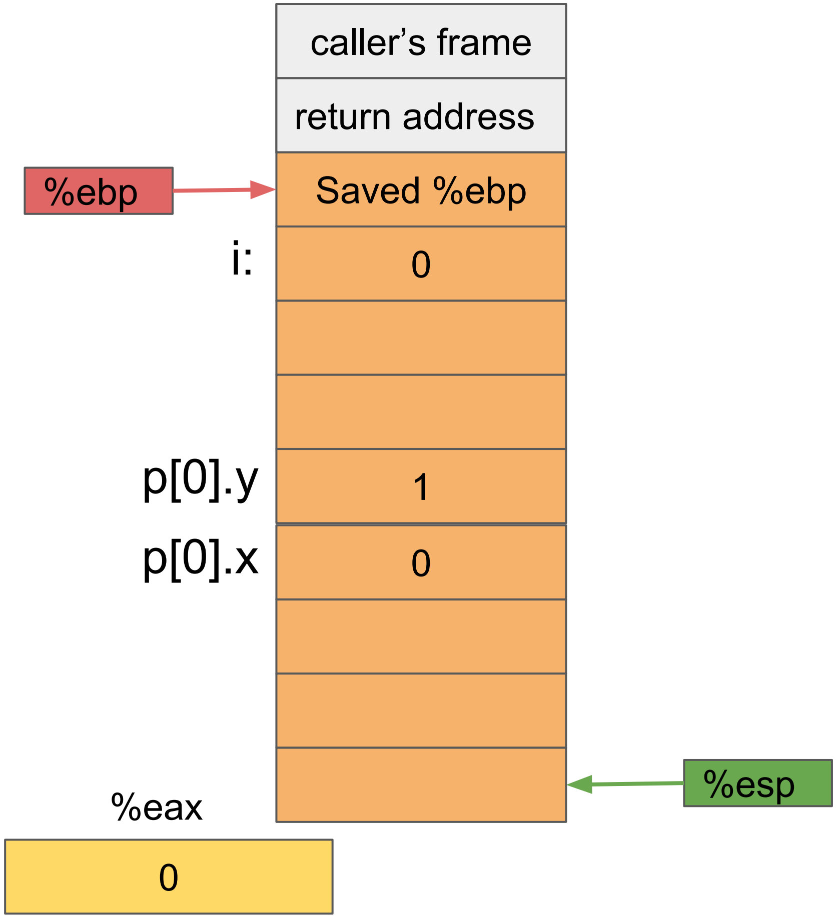
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



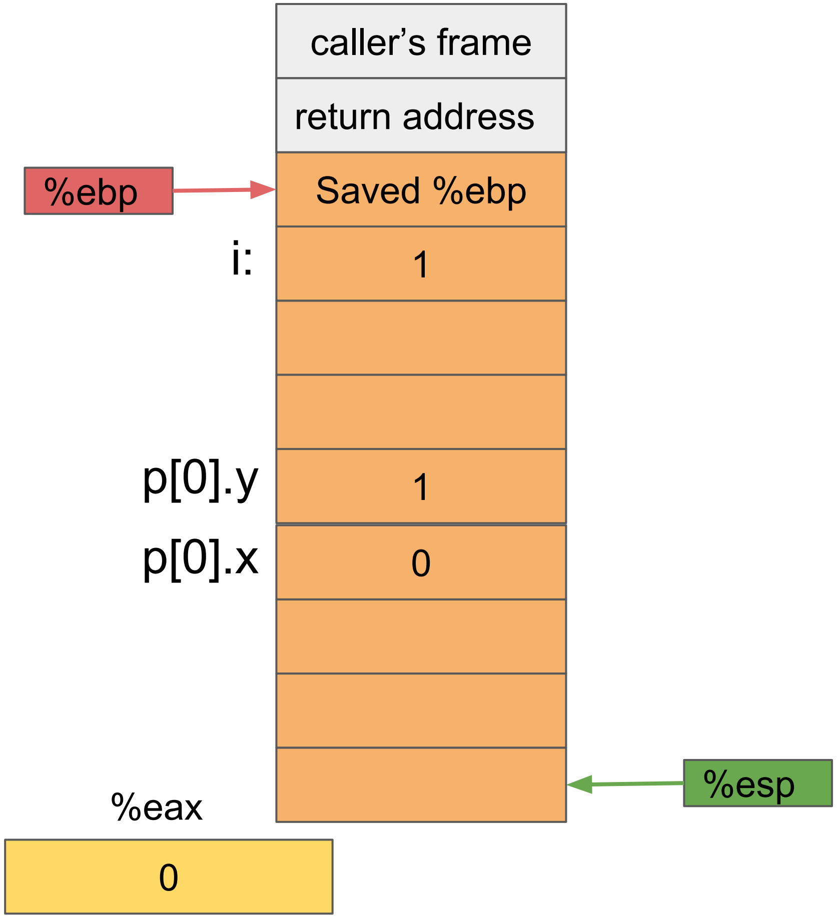
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp

Saved %ebp

i: 1

p[0].y 1

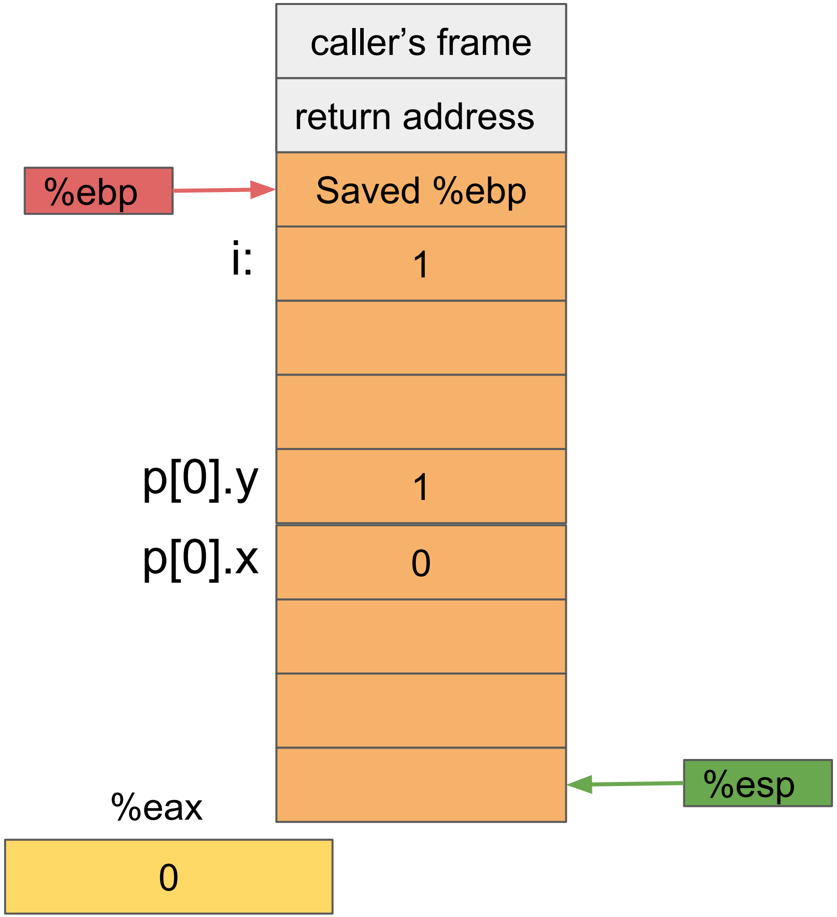
p[0].x 0

%esp

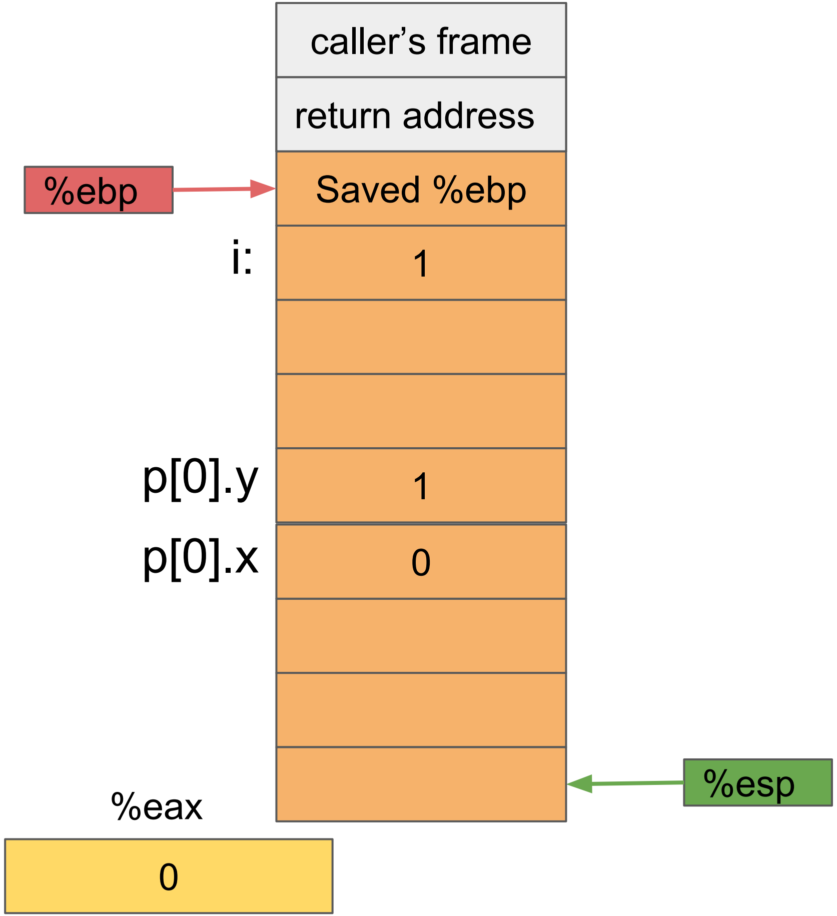
%eax

0

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

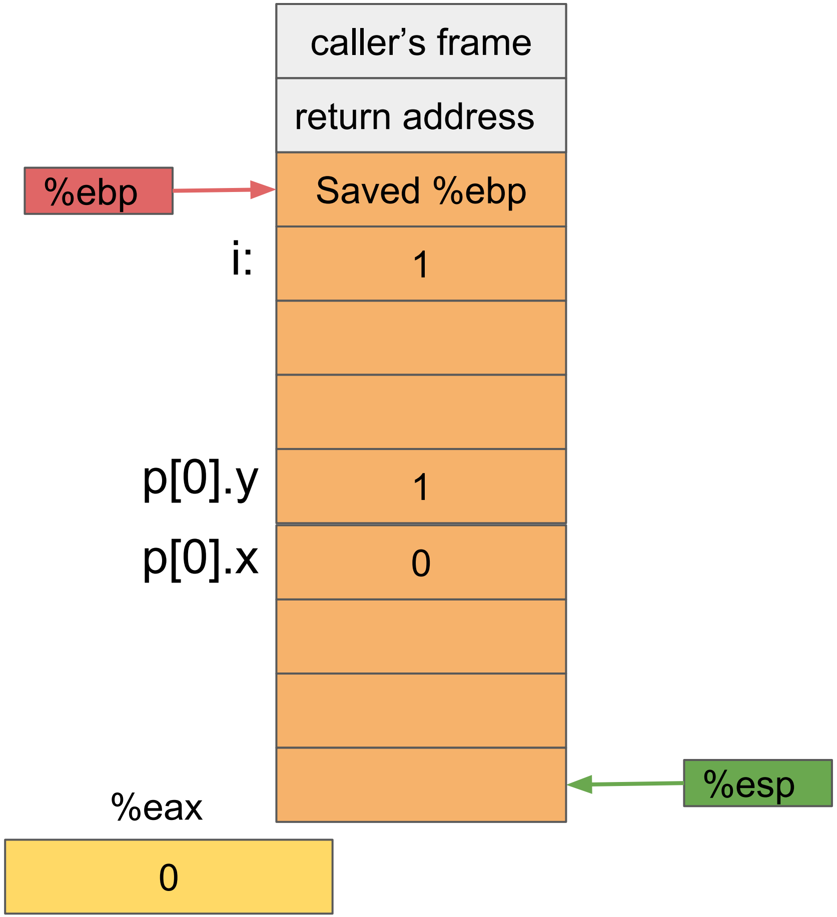


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



Jump if 1 <= 1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp

Saved %ebp

i: 1

p[0].y 1

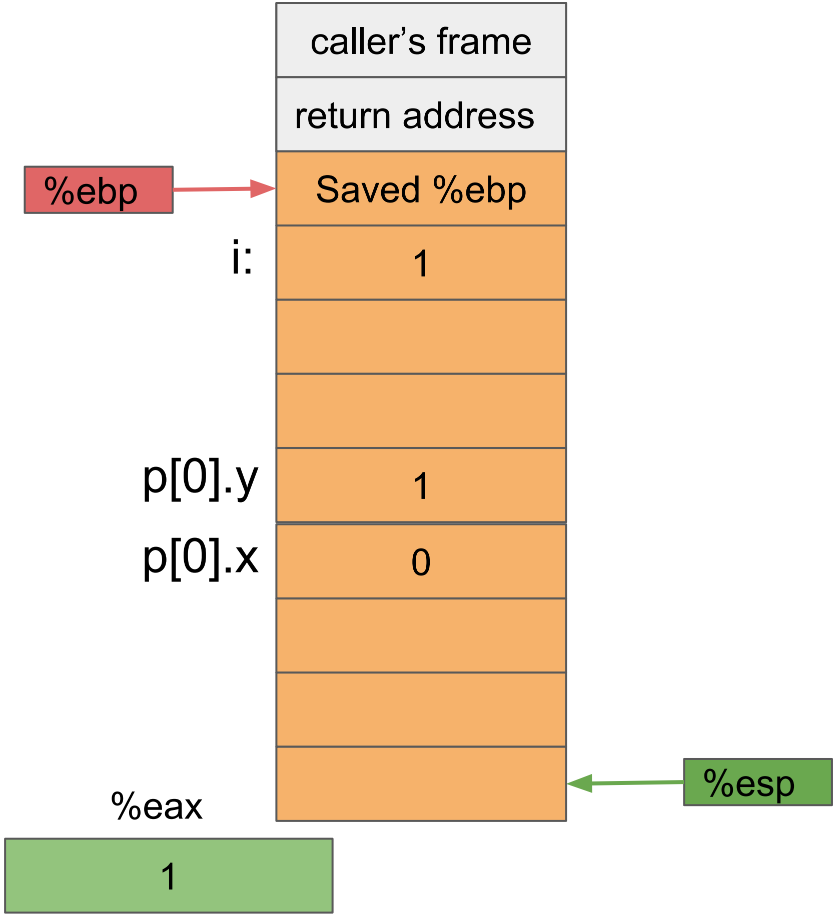
p[0].x 0

%esp

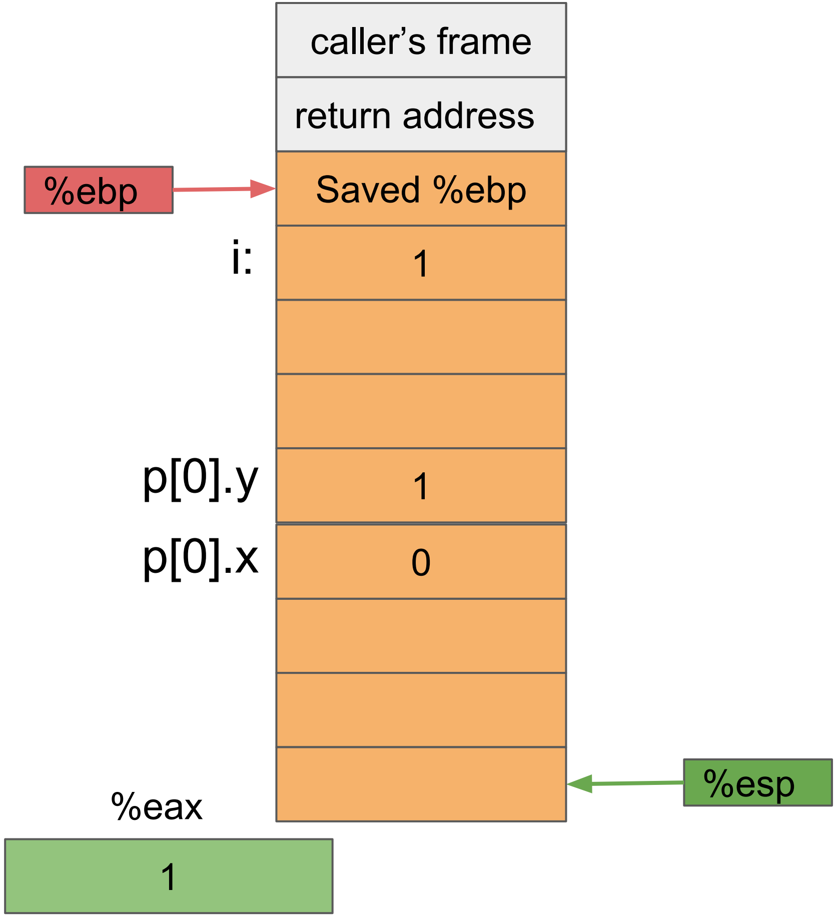
%eax

0

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp → Saved %ebp

i: 1

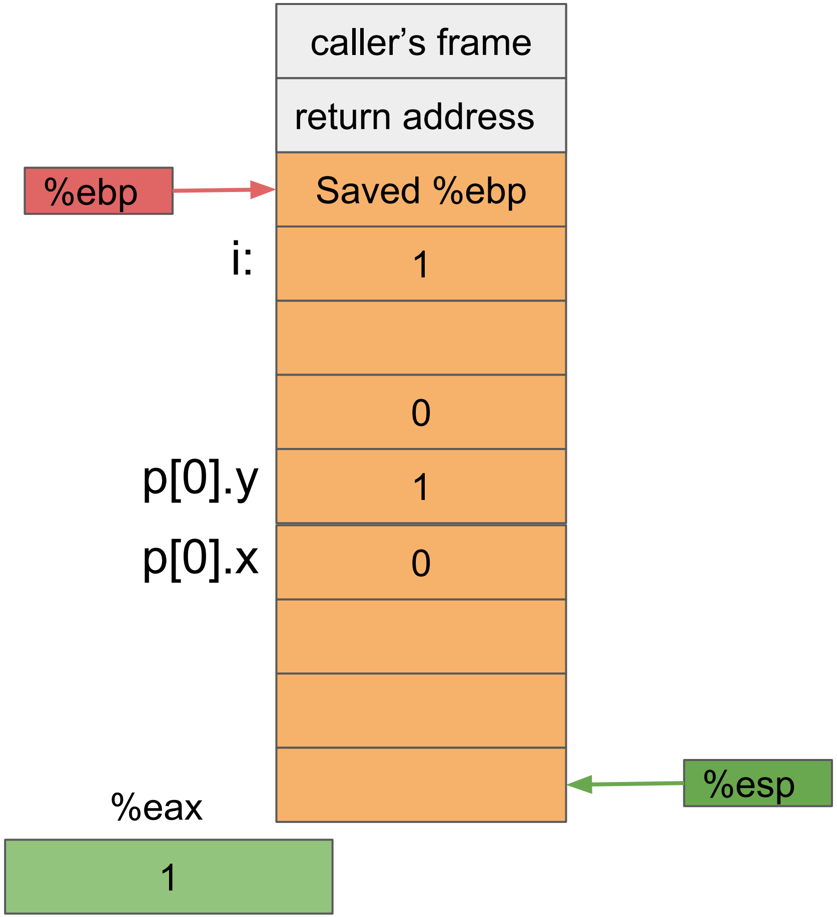
p[0].y 1

p[0].x 0

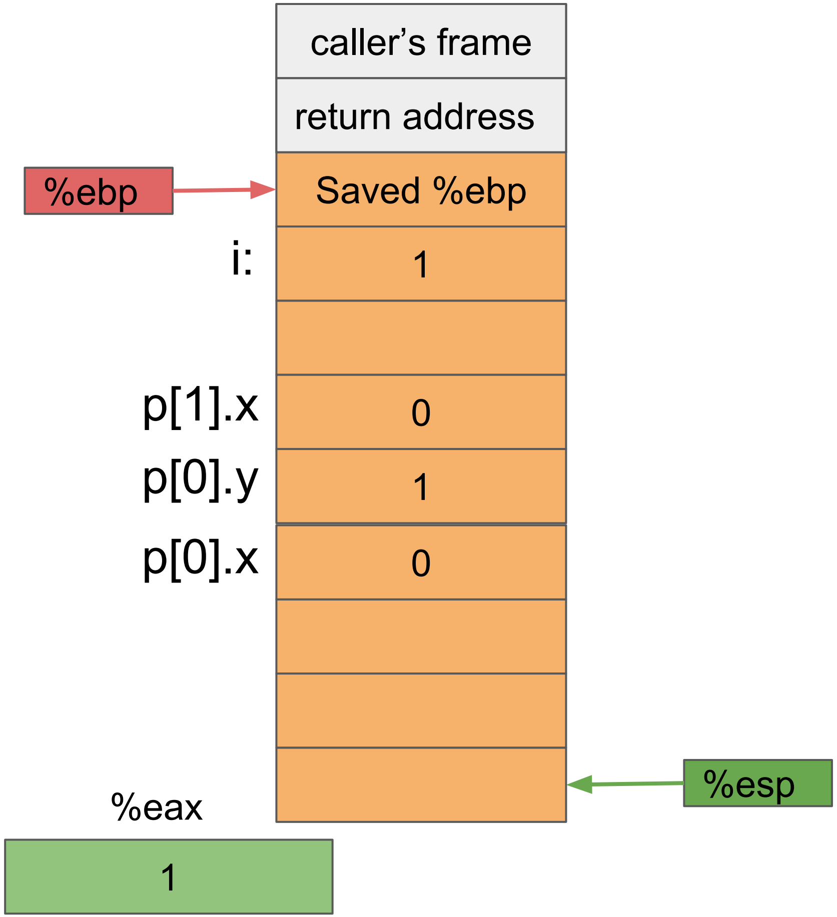
%esp

%eax 1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp

Saved %ebp

i: 1

p[1].x 0

p[0].y 1

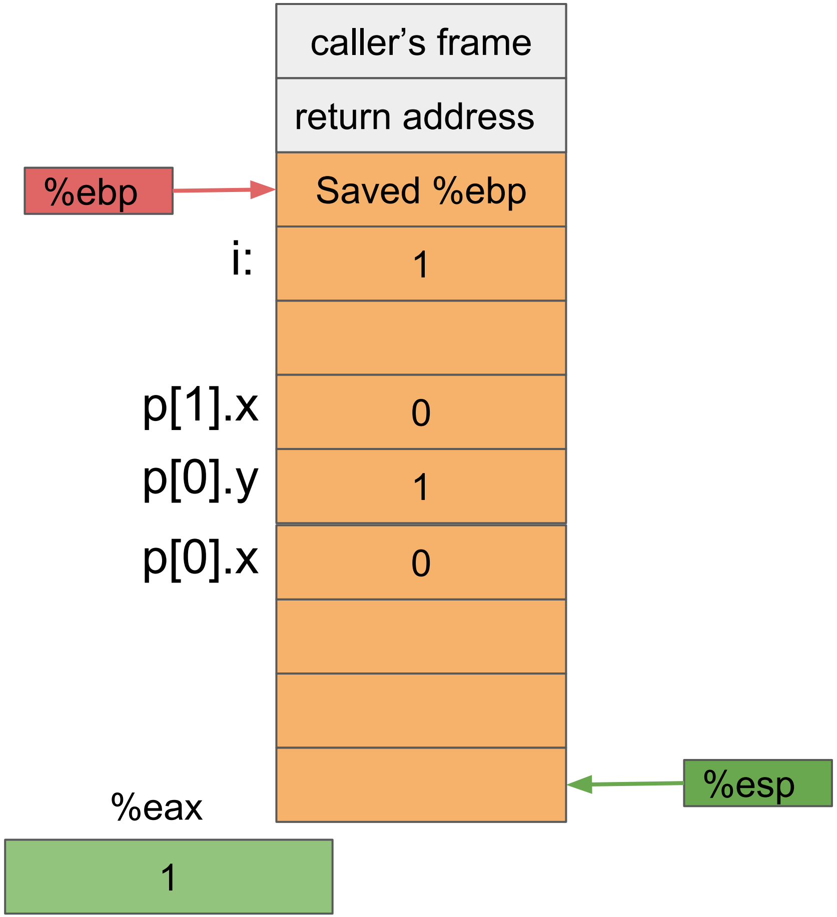
p[0].x 0

%esp

%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp

Saved %ebp

i: 1

p[1].x 0

p[0].y 1

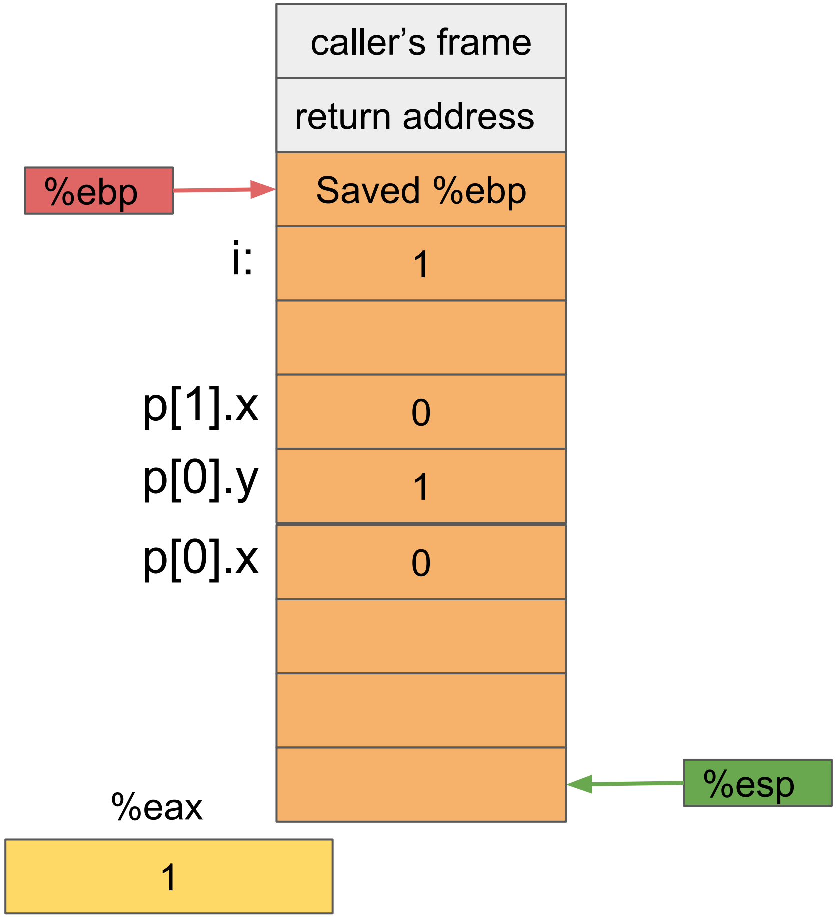
p[0].x 0

%esp

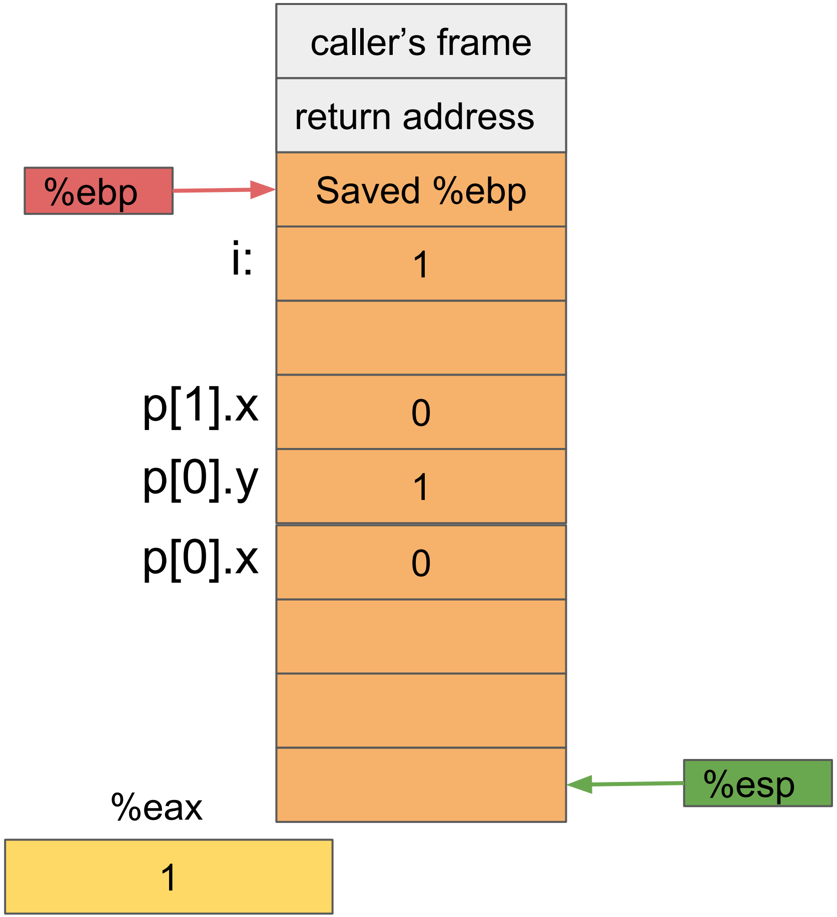
%eax

1

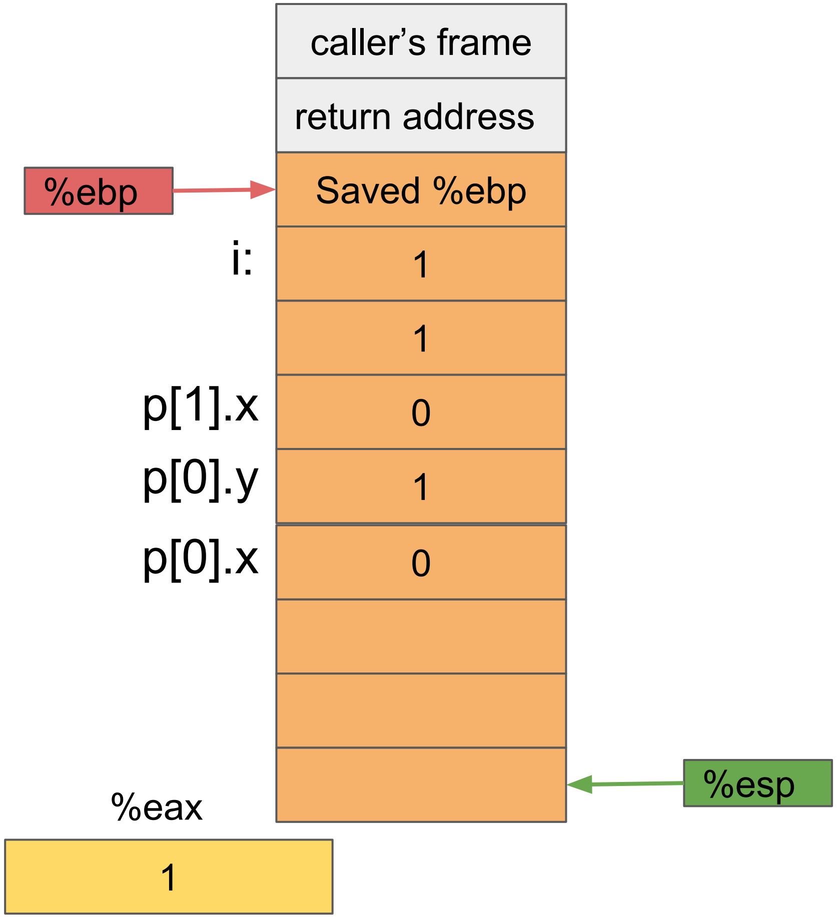
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp → Saved %ebp

i: 1

1

p[1].x 0

p[0].y 1

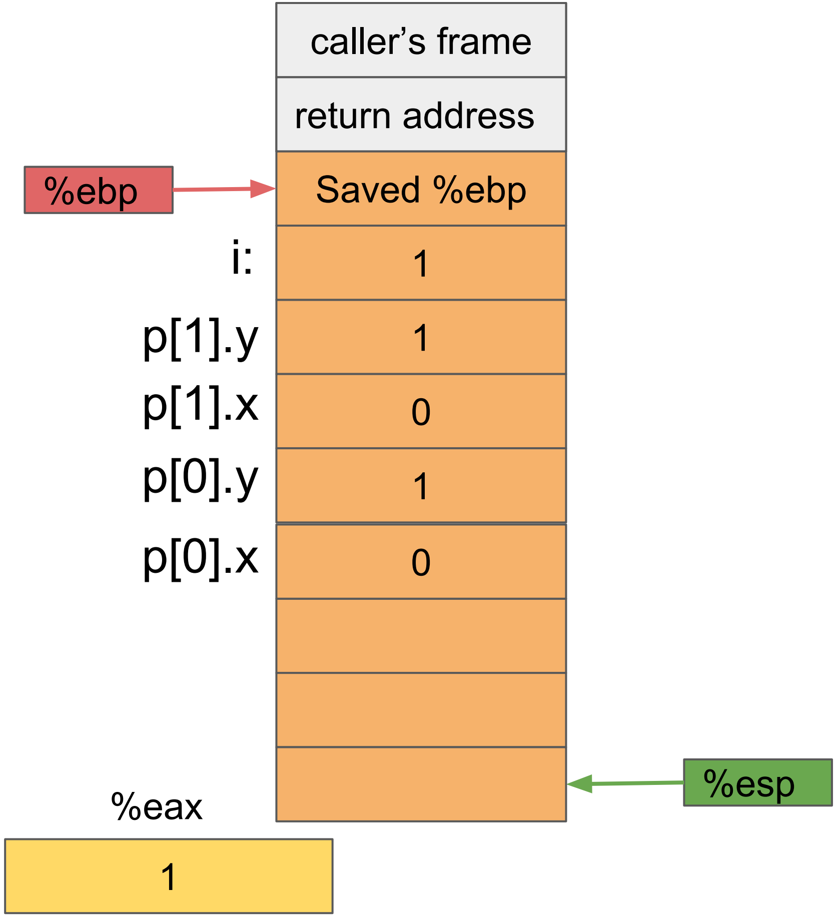
p[0].x 0

%esp →

%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

%ebp → Saved %ebp

i: 1

p[1].y 1

p[1].x 0

p[0].y 1

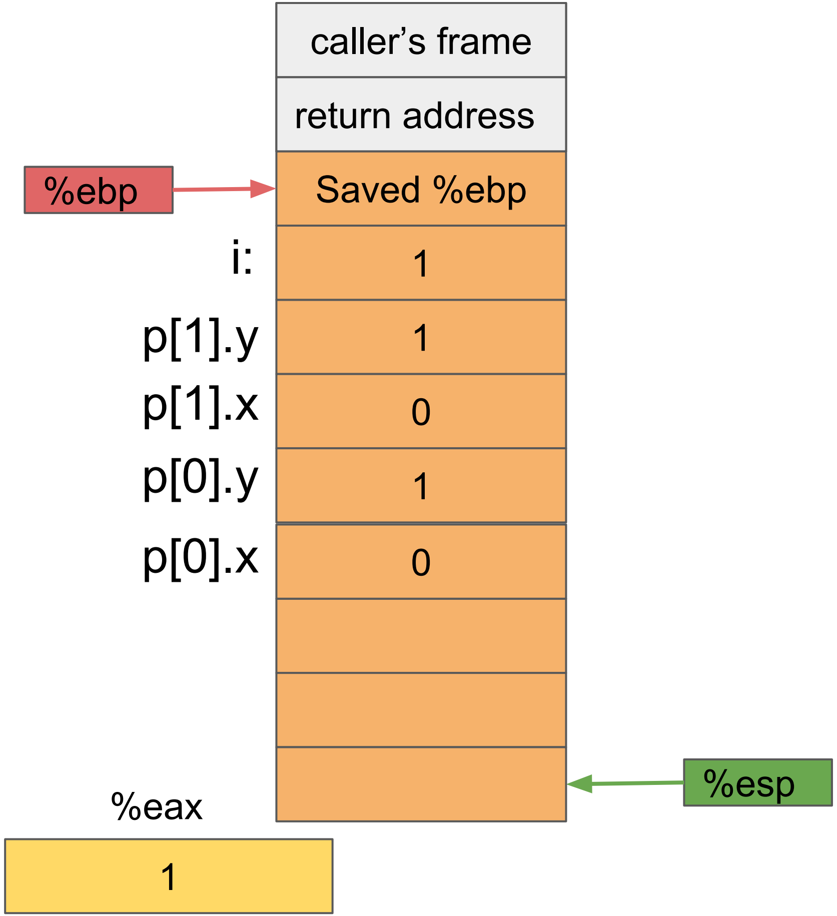
p[0].x 0

%esp

%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp

Saved %ebp

i: 1

p[1].y 1

p[1].x 0

p[0].y 1

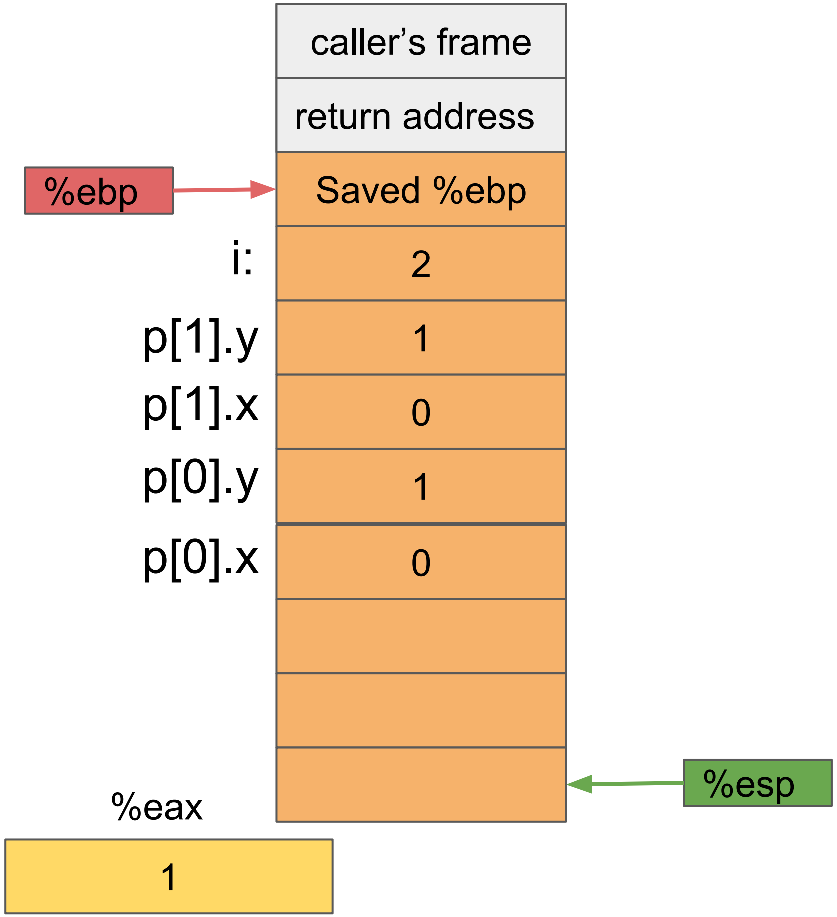
p[0].x 0

%esp

%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp → Saved %ebp

i: 2

p[1].y 1

p[1].x 0

p[0].y 1

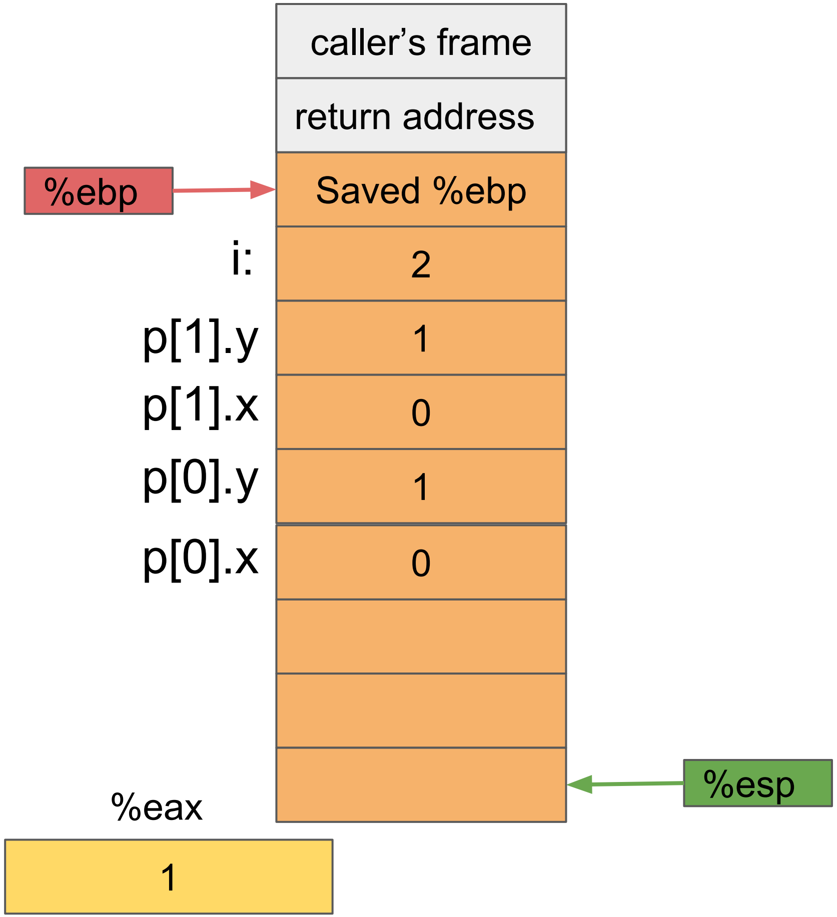
p[0].x 0

%esp →

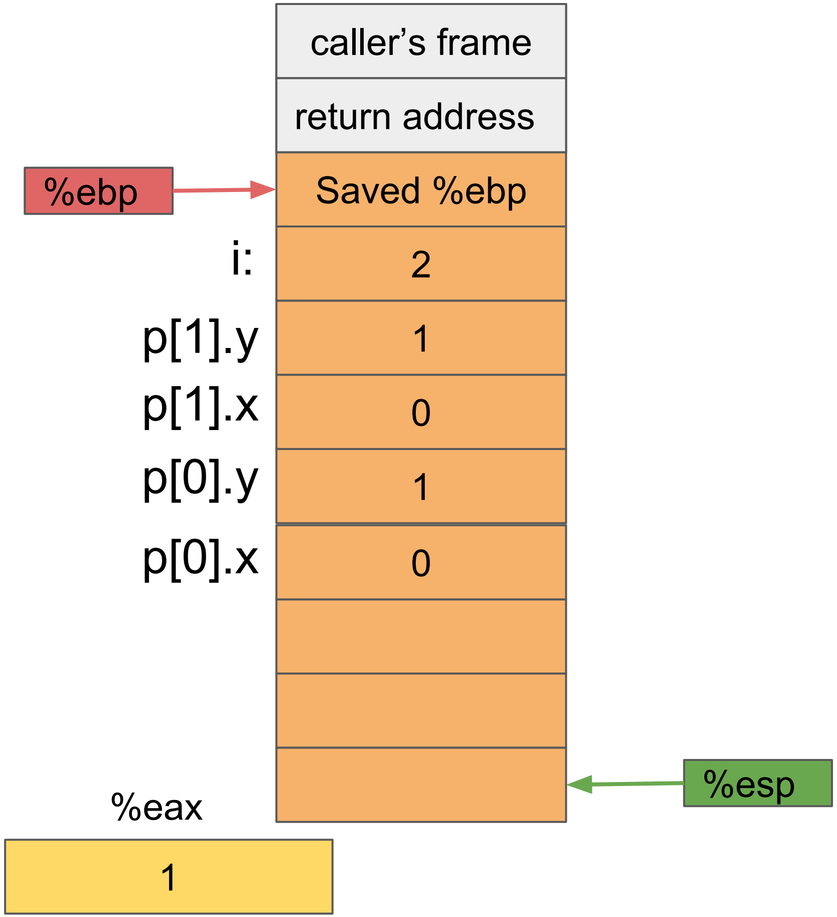
%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

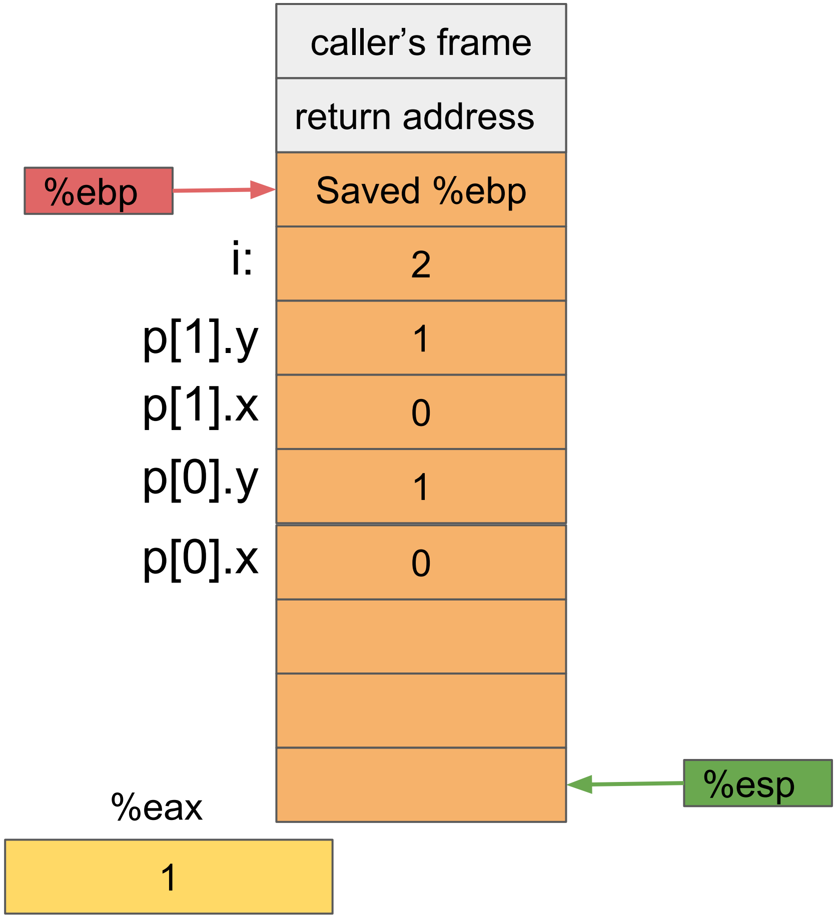


```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



Jump if 2 <= 1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller's frame

return address

%ebp → Saved %ebp

i: 2

p[1].y 1

p[1].x 0

p[0].y 1

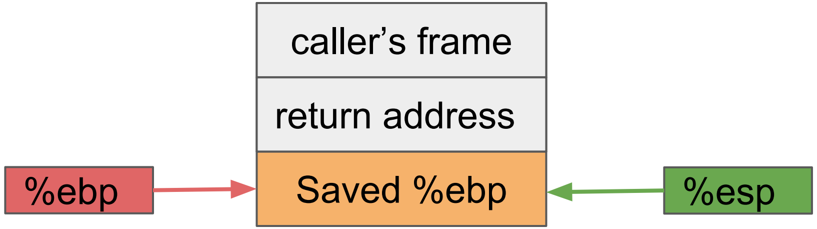
p[0].x 0

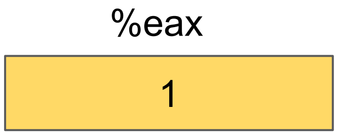
%esp

%eax

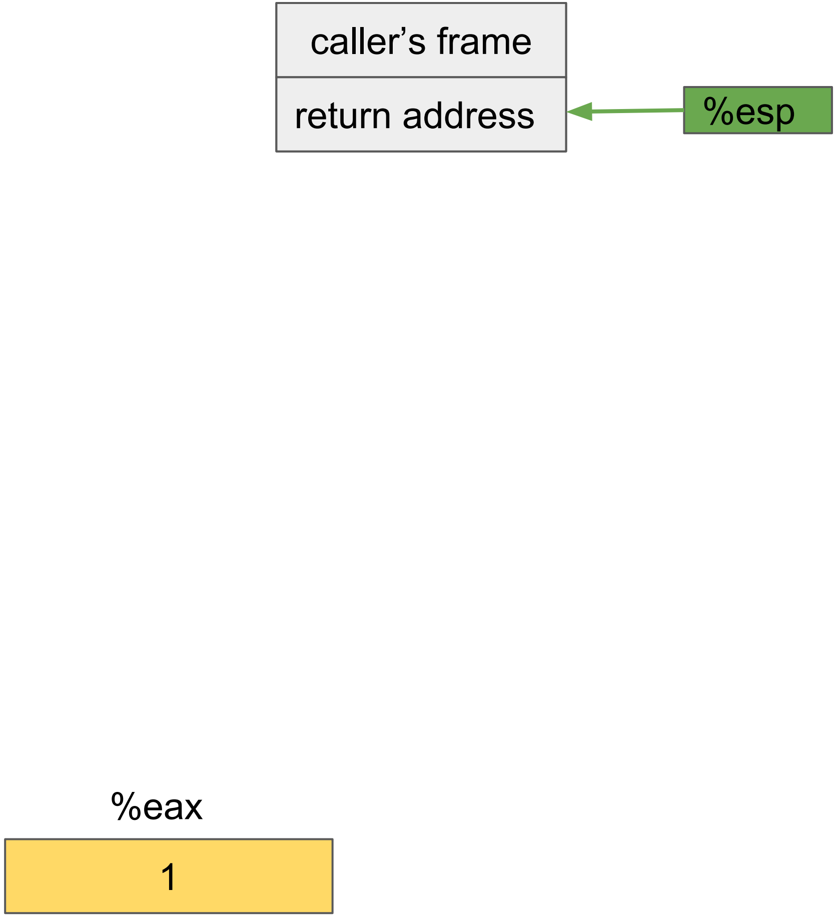
1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```

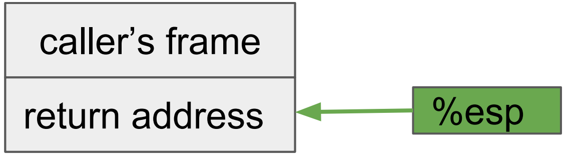




```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



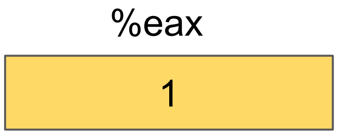
```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



caller’s frame

return address

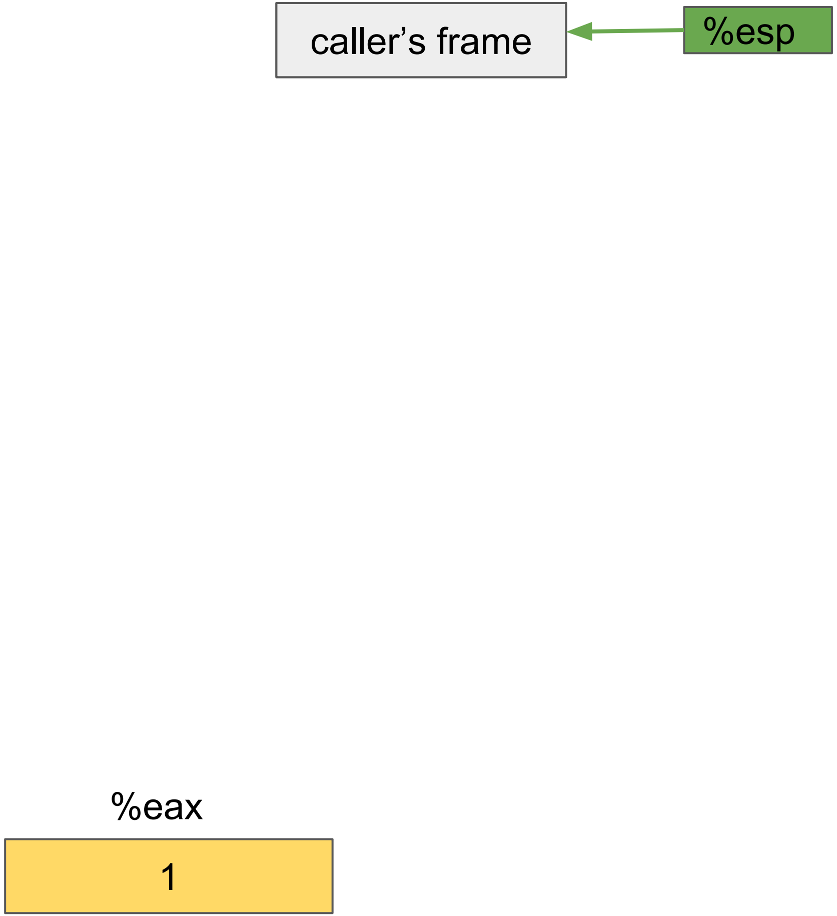
%esp



%eax

1

```
func:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $32, %esp
    movl    $0, -4(%ebp)
    jmp .L2
.L3:
    movl    -4(%ebp), %eax
    movl    $0, -20(%ebp,%eax,8)
    movl    -4(%ebp), %eax
    movl    $1, -16(%ebp,%eax,8)
    addl    $1, -4(%ebp)
.L2:
    cmpl    $1, -4(%ebp)
    jle .L3
    leave
    ret
```



Questions?