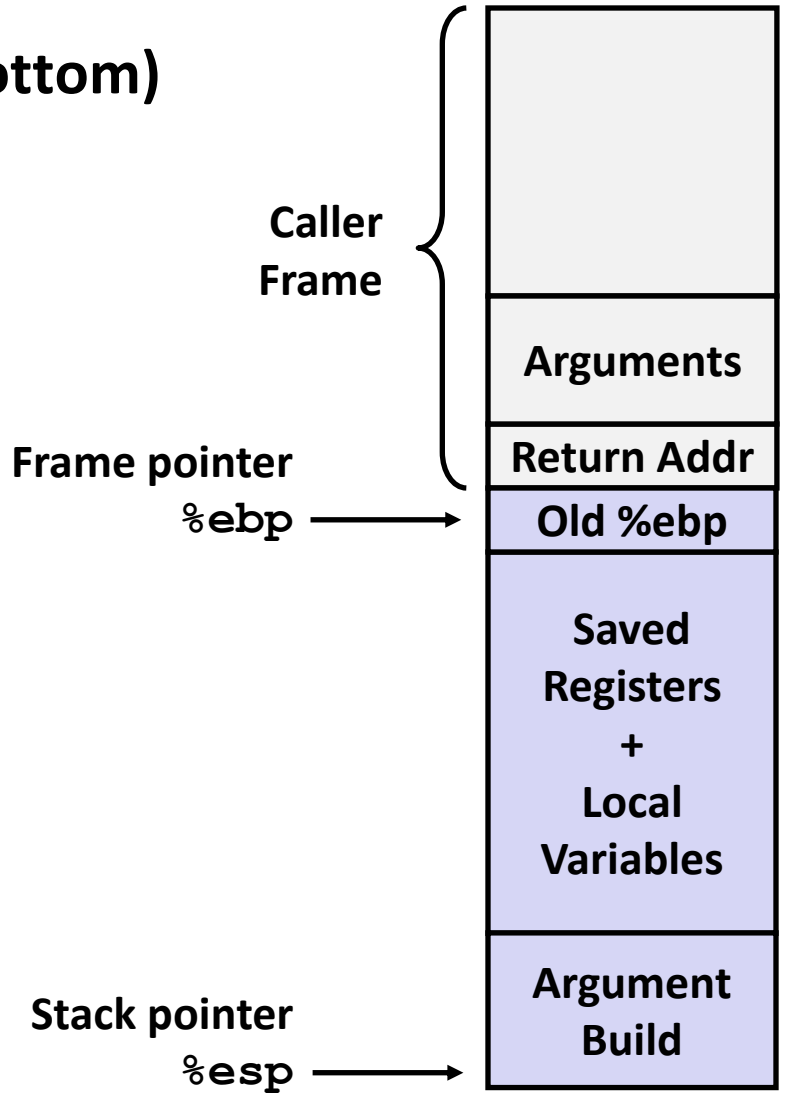
Section 5: Procedures & Stacks

- Stacks in memory and stack operations
- The stack used to keep track of procedure calls
- Return addresses and return values
- Stack-based languages
- The Linux stack frame
- Passing arguments on the stack
- Allocating local variables on the stack
- Register-saving conventions
- Procedures and stacks on x64 architecture

IA32/Linux Stack Frame

- **Current Stack Frame ("Top" to Bottom)**
  - "Argument build" area (parameters for function about to be called)
  - Local variables (if can't be kept in registers)
  - Saved register context (when reusing registers)
  - Old frame pointer (for caller)
- **Caller's Stack Frame**
  - Return address
    - Pushed by `call` instruction
  - Arguments for this call



Revisiting swap

```
int zip1 = 15213;
int zip2 = 98195;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

Revisiting swap

```
int zip1 = 15213;
int zip2 = 98195;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

Calling swap from call_swap

```
call_swap:
    • • •
    pushl $zip2    # Global Var
    pushl $zip1    # Global Var
    call swap
    • • •
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

Revisiting swap

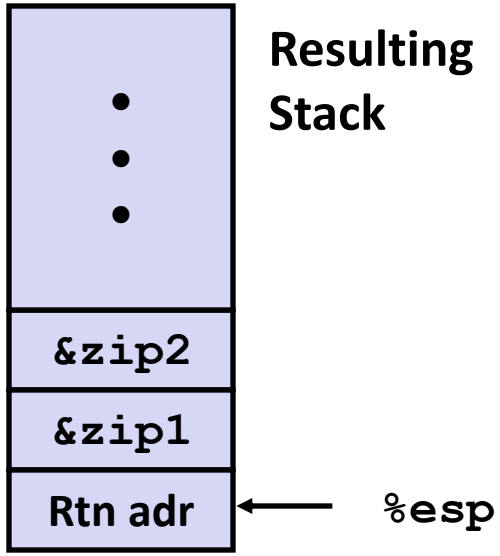
```
int zip1 = 15213;
int zip2 = 98195;

void call_swap()
{
  swap(&zip1, &zip2);
}
```

Calling swap from call_swap

```
call_swap:
    • • •
    pushl $zip2    # Global Var
    pushl $zip1    # Global Var
    call swap
    • • •
```

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```



Revisiting swap

```
void swap(int *xp, int *yp)
{
  int t0 = *xp;
  int t1 = *yp;
  *xp = t1;
  *yp = t0;
}
```

```
swap:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx

    movl 12(%ebp),%ecx
    movl 8(%ebp),%edx
    movl (%ecx),%eax
    movl (%edx),%ebx
    movl %eax,(%edx)
    movl %ebx,(%ecx)

    movl -4(%ebp),%ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

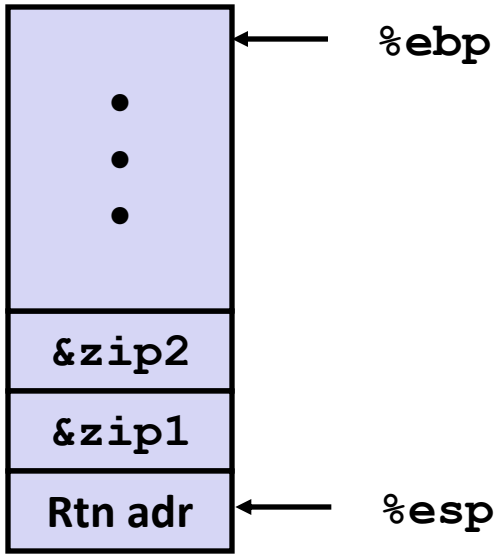
Set Up

Body

Finish

swap Setup #1

Entering Stack

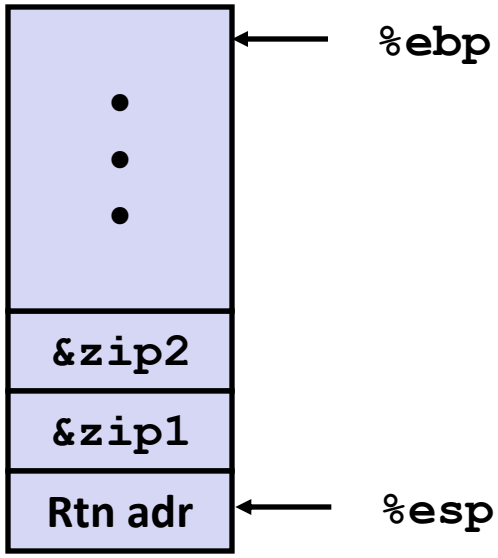


Resulting Stack?

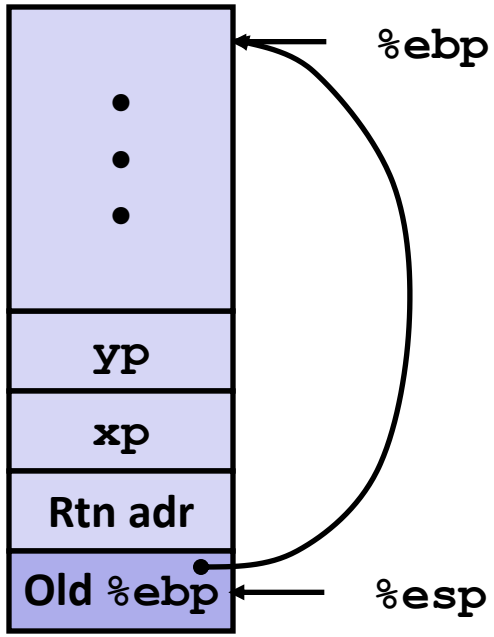
```
swap:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
```

swap Setup #1

Entering Stack



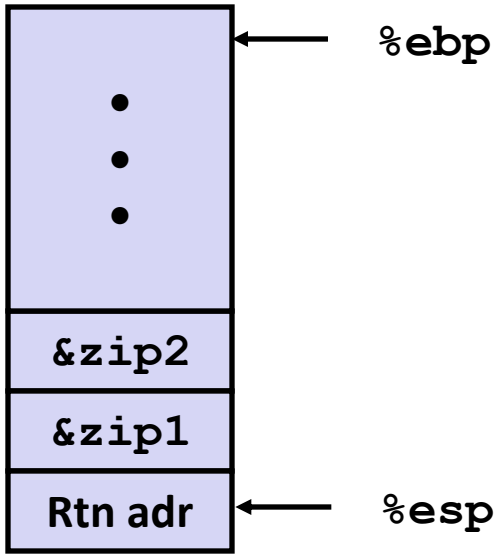
Resulting Stack



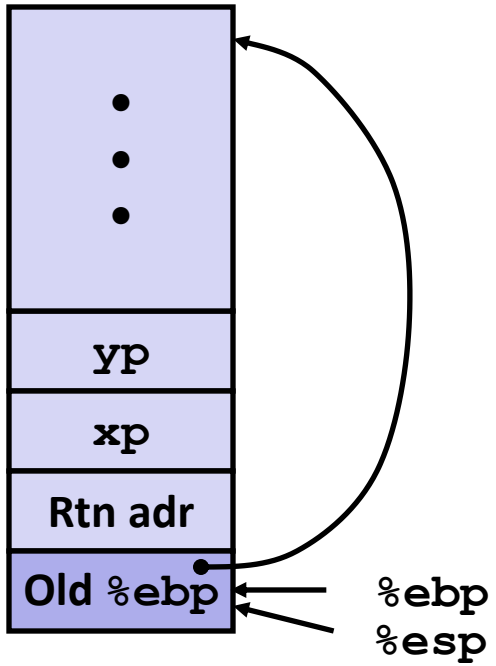

```
swap:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
```

swap Setup #2

Entering Stack



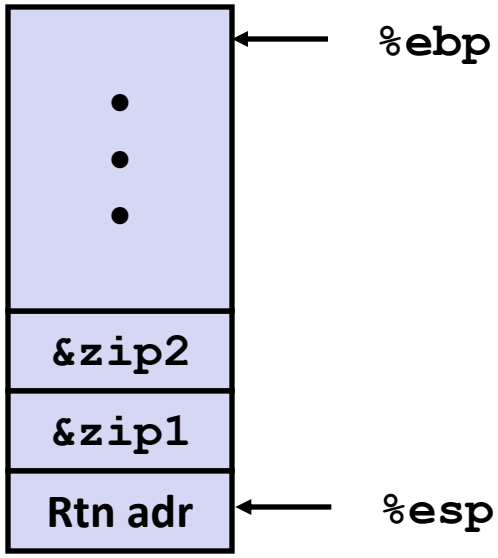
Resulting Stack



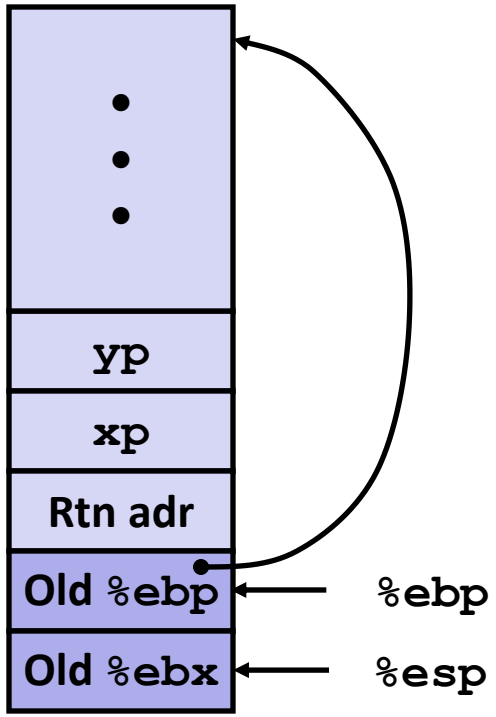

```
swap:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
```

swap Setup #3

Entering Stack

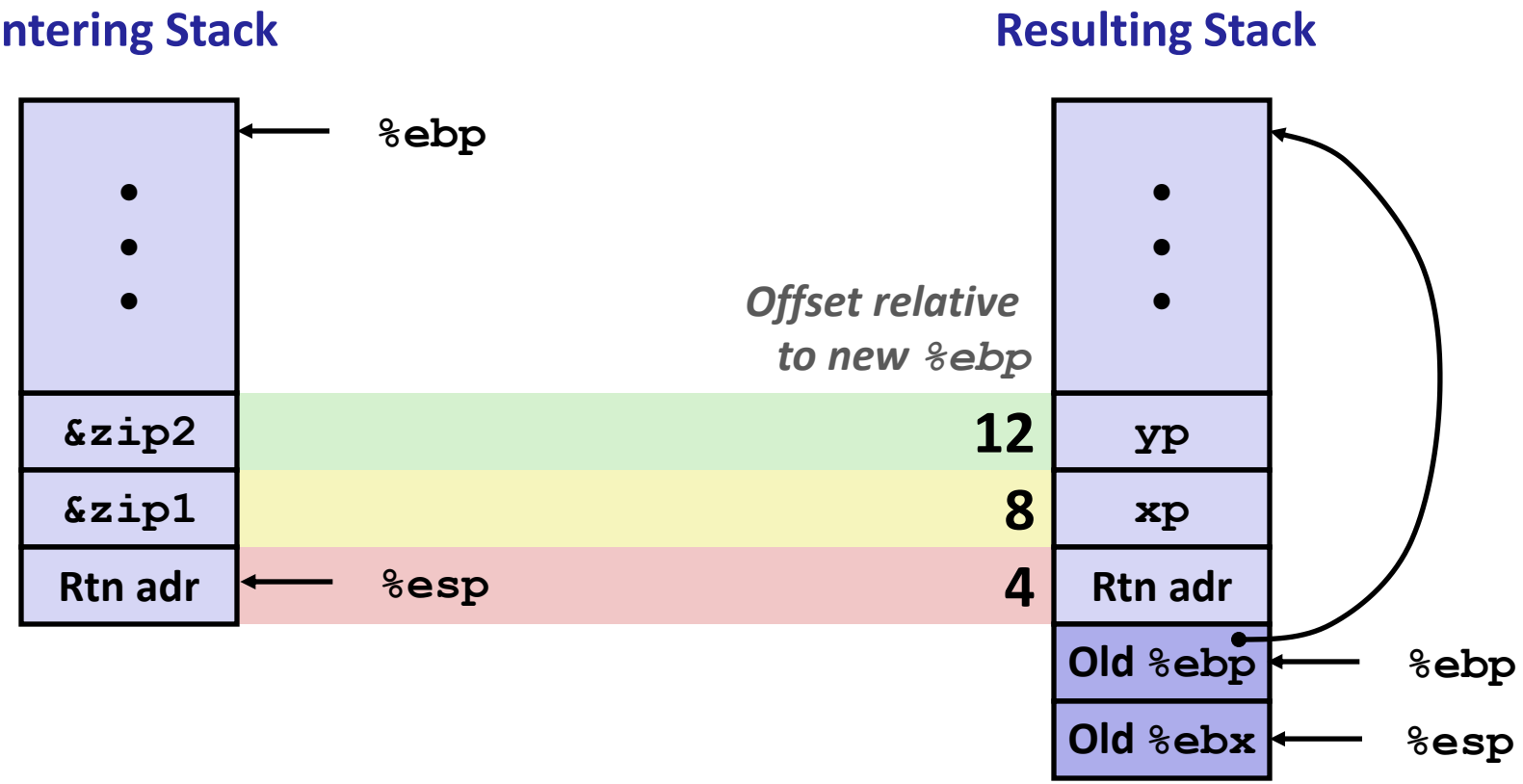


Resulting Stack




```
swap:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
```

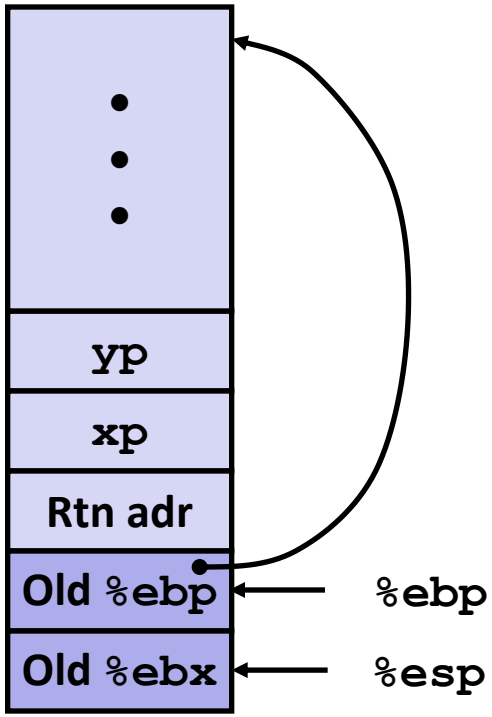
swap Body




```
movl 12(%ebp),%ecx   # get yp
movl  8(%ebp),%edx   # get xp
. . .
```

swap Finish #1

swap's Stack

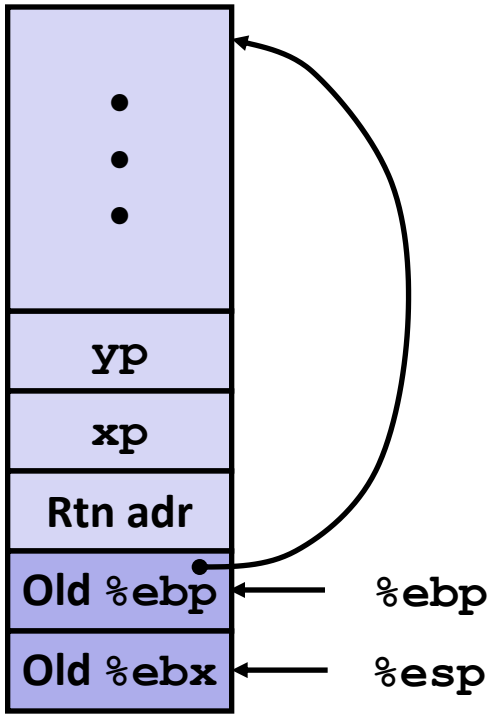



```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

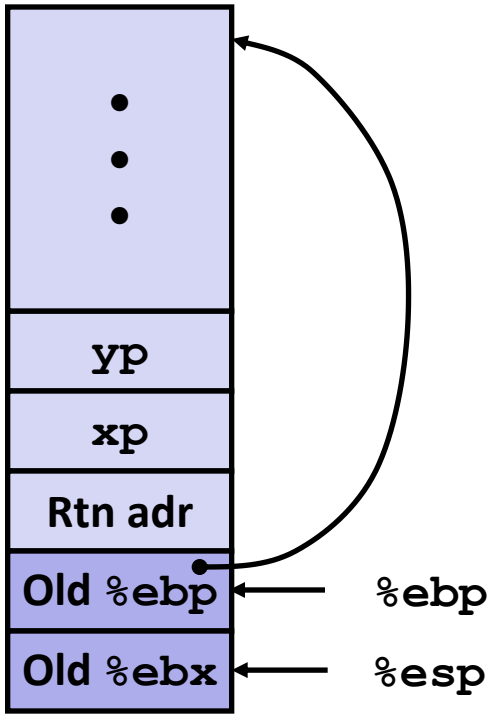
Resulting Stack?

swap Finish #1

swap's Stack



Resulting Stack

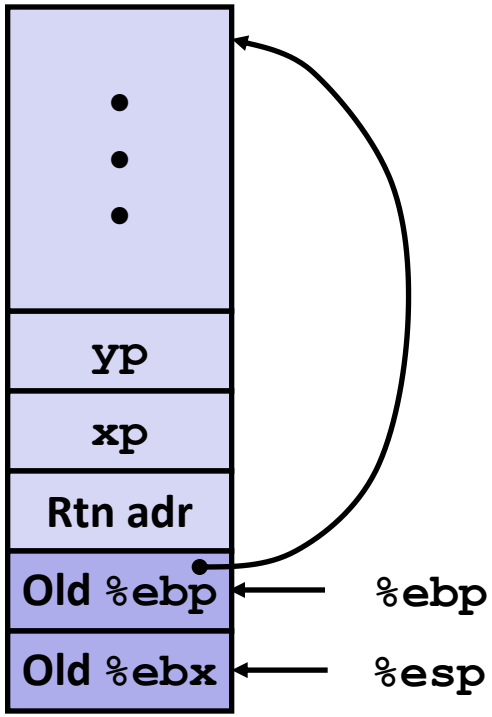



```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

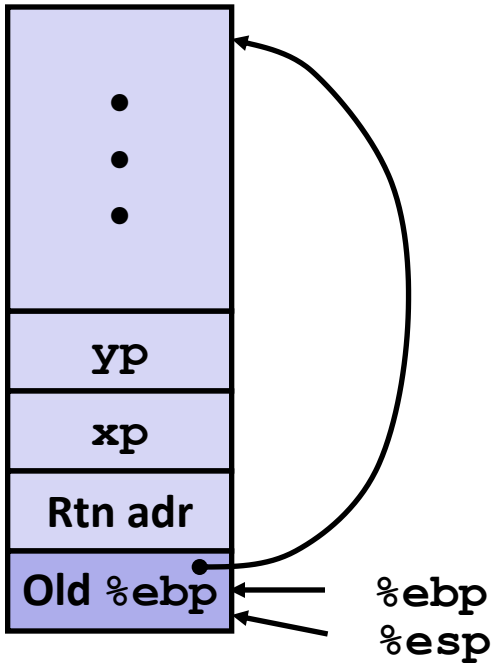
Observation: Saved and restored register %ebx

swap Finish #2

swap's Stack



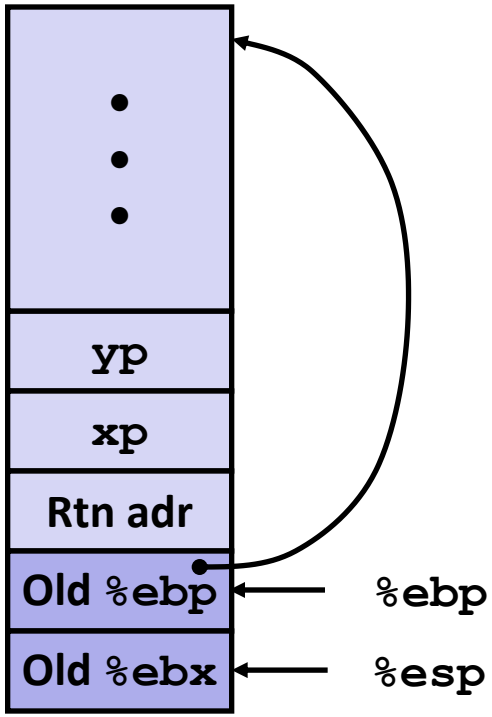
Resulting Stack



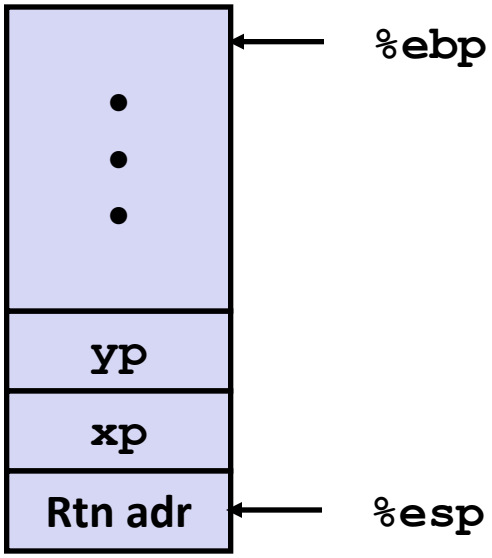

```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

swap Finish #3

swap's Stack



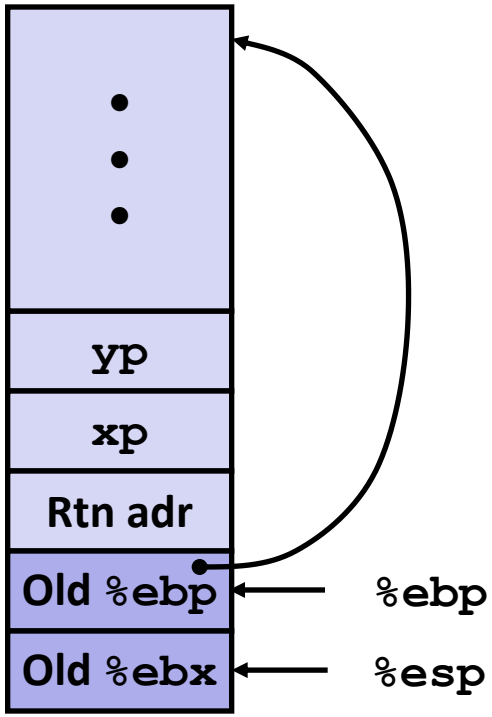
Resulting Stack



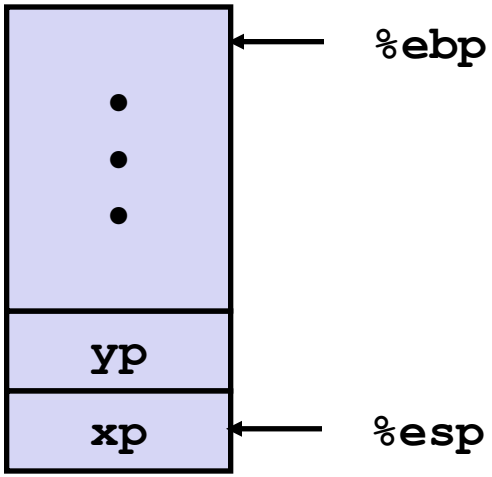

```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

swap Finish #4

swap's Stack



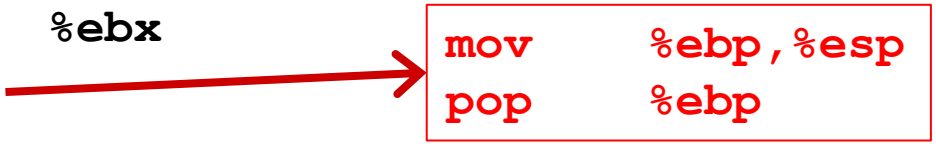
Resulting Stack




```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

Disassembled swap

```
080483a4 <swap>:
 80483a4:    55              push   %ebp
 80483a5:    89 e5           mov    %esp,%ebp
 80483a7:    53              push   %ebx
 80483a8:    8b 55 08        mov    0x8(%ebp),%edx
 80483ab:    8b 4d 0c        mov    0xc(%ebp),%ecx
 80483ae:    8b 1a           mov    (%edx),%ebx
 80483b0:    8b 01           mov    (%ecx),%eax
 80483b2:    89 02           mov    %eax,(%edx)
 80483b4:    89 19           mov    %ebx,(%ecx)
 80483b6:    5b              pop    %ebx
 80483b7:    c9              leave
 80483b8:    c3              ret
```

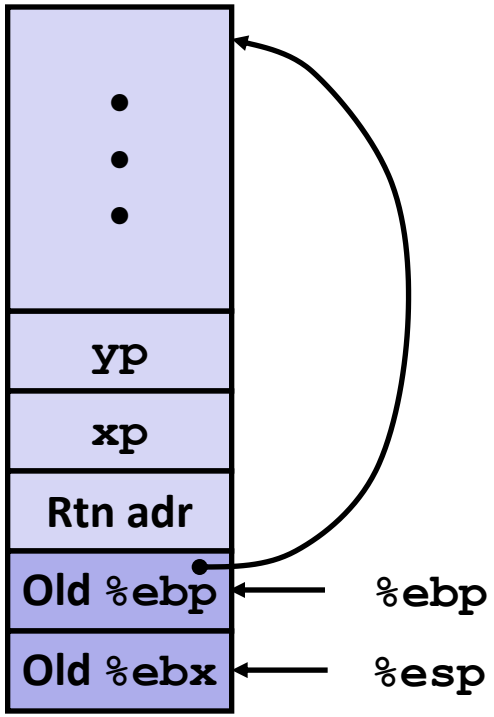


Calling Code

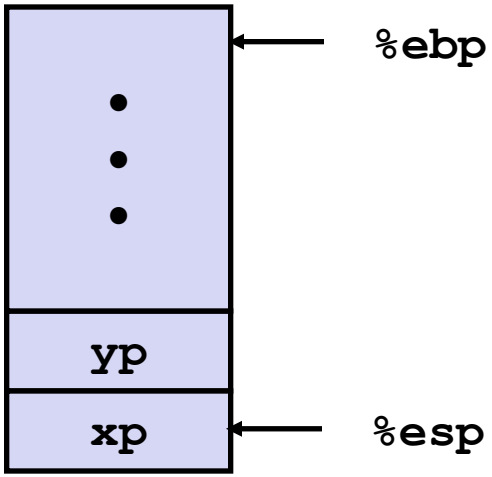
```
 8048409:    e8 96 ff ff ff    call 80483a4 <swap>
 804840e:    8b 45 f8          mov  0xfffffff8(%ebp),%eax
```

swap Finish #4

swap's Stack



Resulting Stack




```
movl -4(%ebp),%ebx
movl %ebp,%esp
popl %ebp
ret
```

- **Observation**
  - Saved & restored register **%ebx**
  - Didn't do so for **%eax**, **%ecx**, or **%edx**