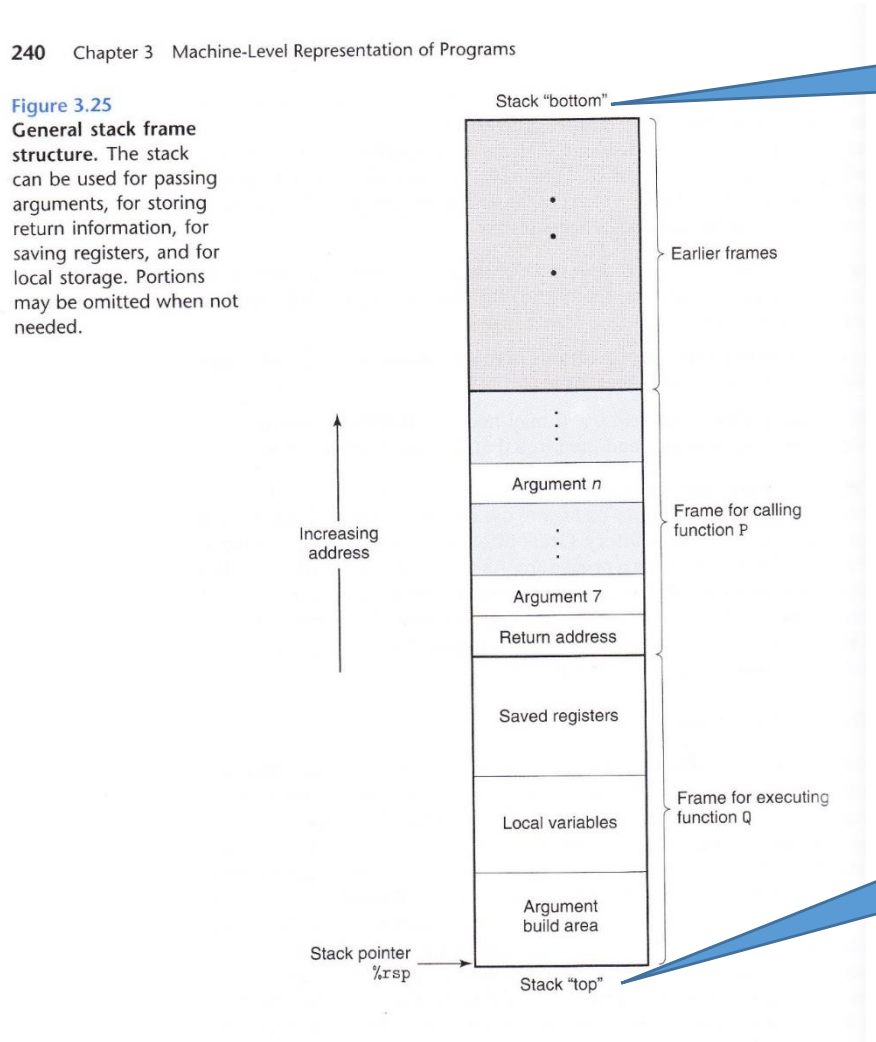
IA32 vs. IA64

Computer Systems 3.7

Naming Convention Conflict



I call this the "top of the stack"

I call this the "bottom of the stack"

Why 64 bit addresses?

- 32 bit address space
  - 0x0000 0000 – xFFFF FFFF
  - 2^{32} bytes = 4G
  - if you put 3200 chars on a page, that's a stack of paper about 140 meters

- 64 bit address space
  - 0x0000 0000 0000 0000 – 0xFFFF FFFF FFFF FFFF
  - 2^{64} bytes = 16 exbibytes (4G x 4G) =~ 1.84 x 10^{19}
  - Squares the amount of memory that can be addressed!
  - That's twice from the earth to the sun and back!

But that's not all that changed!

- IA64 has a whole new set of conventions!
- We won't study, but we will go through an example

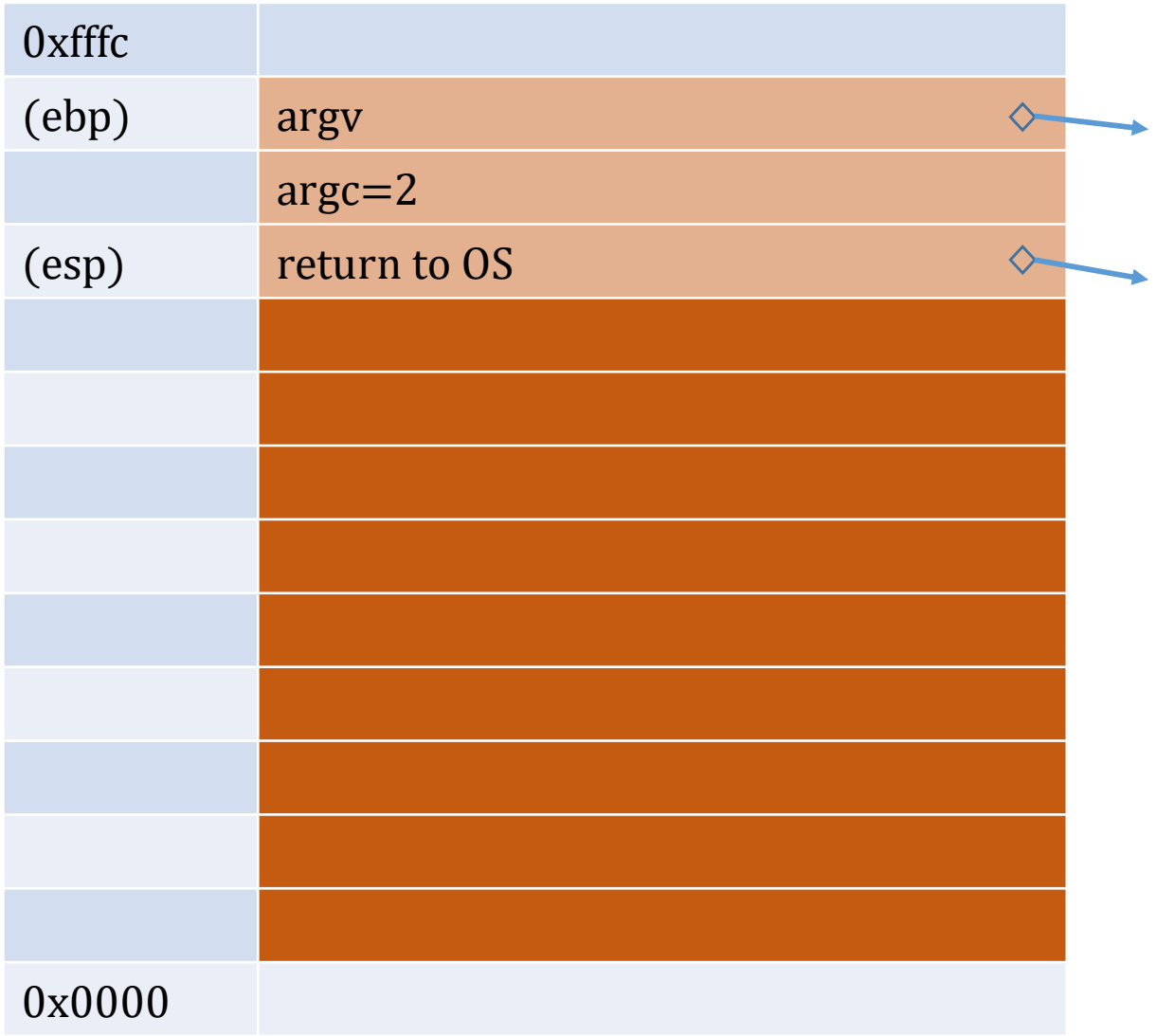
C Code…

See http://www.cs.binghamton.edu/~tbartens/CS220_Spring_2016/examples/xmp_swap/

```
int main(int argc, char **argv) {
    int x,y;
    x=atoi(argv[1]); y=atoi(argv[2]);
    swap(&x,&y);
}
```

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

Stack before OS transfers to main…

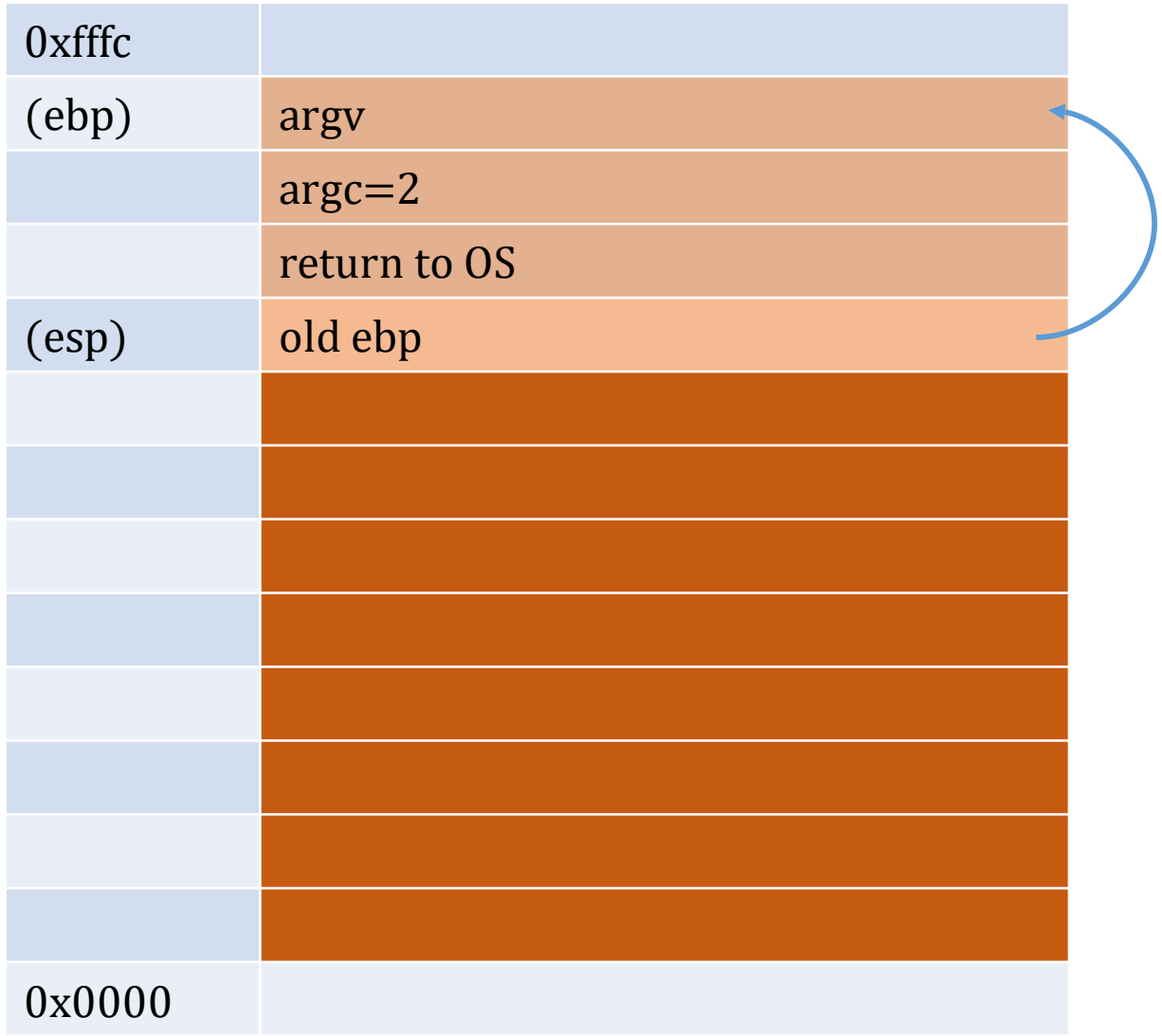


X86 prologue for “main”

```
main: pushl %ebp ; save base pointer on stack
      movl  %esp, %ebp ; Reset base pointer to current stack pointer
      andl  $-16, %esp ; esp = esp & 0xfff0 – start on qword boundary
      subl  $32, %esp; Allocate 32 bytes for locals (int x,y)
```

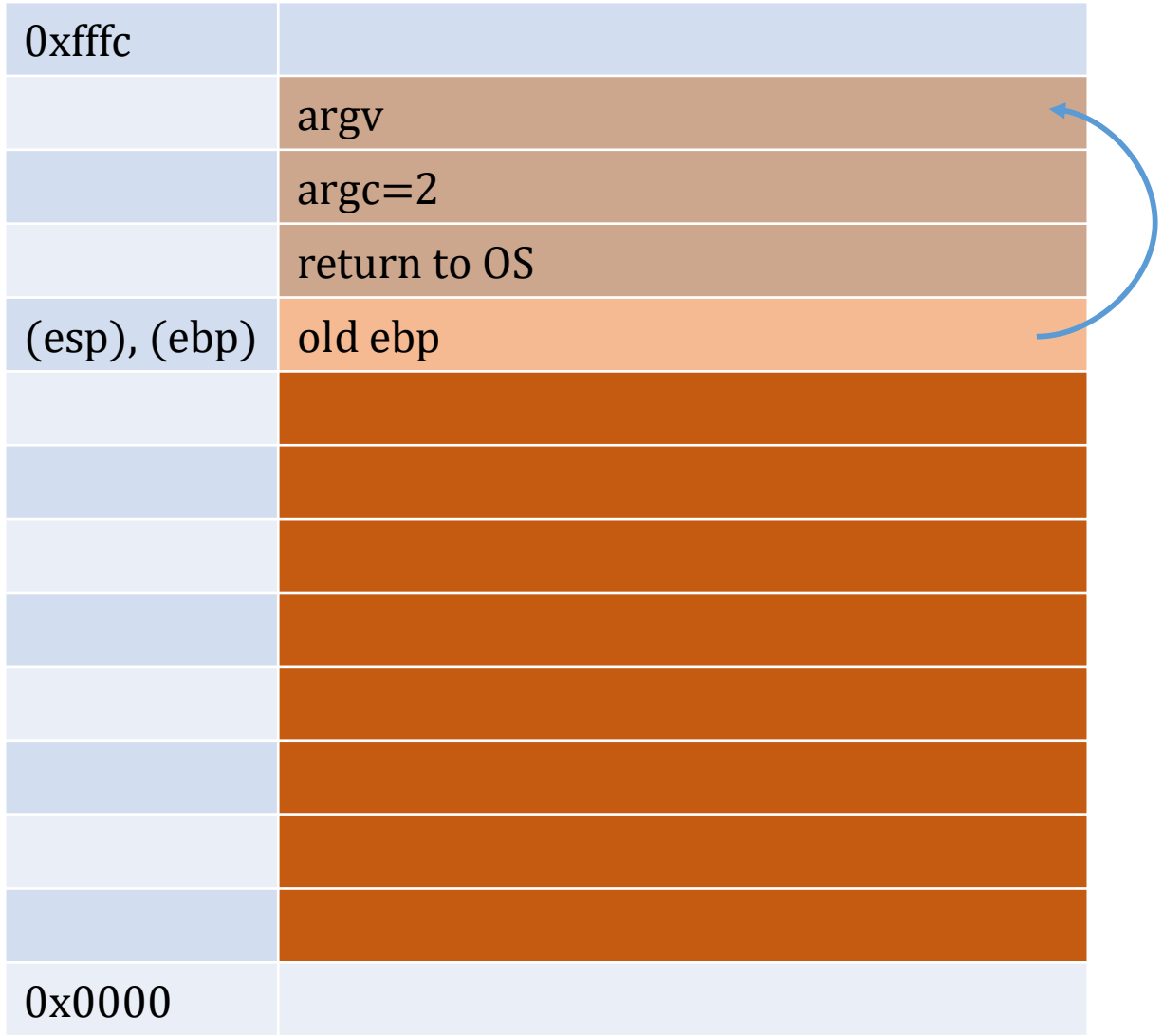
main preamble

```
main:   pushl   %ebp
        movl %esp, %ebp
        andl  $-16, %esp
        subl  $32, %esp
```



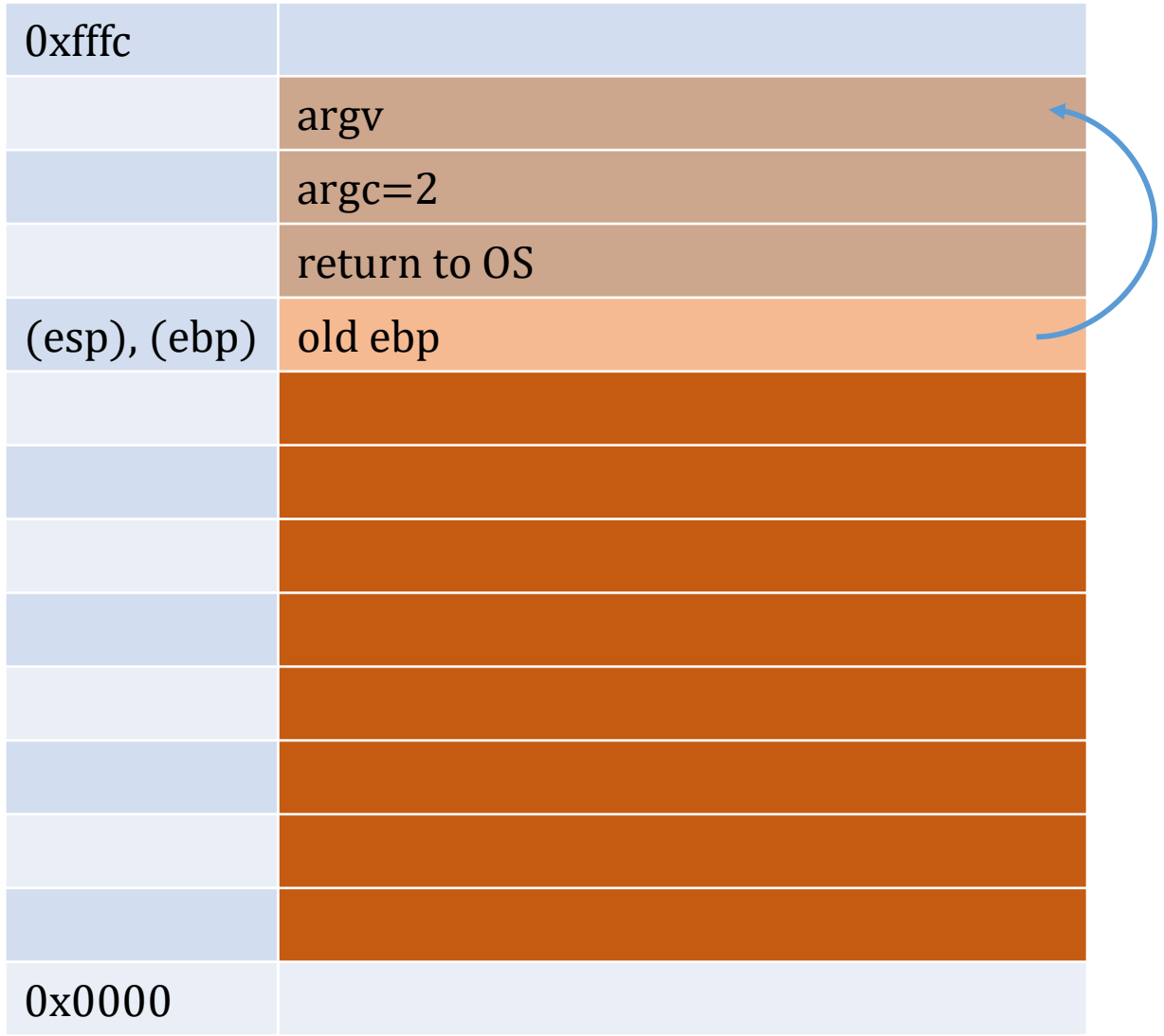
main preamble

```
main:   pushl   %ebp
        movl %esp, %ebp
        andl  $-16, %esp
        subl  $32, %esp
```



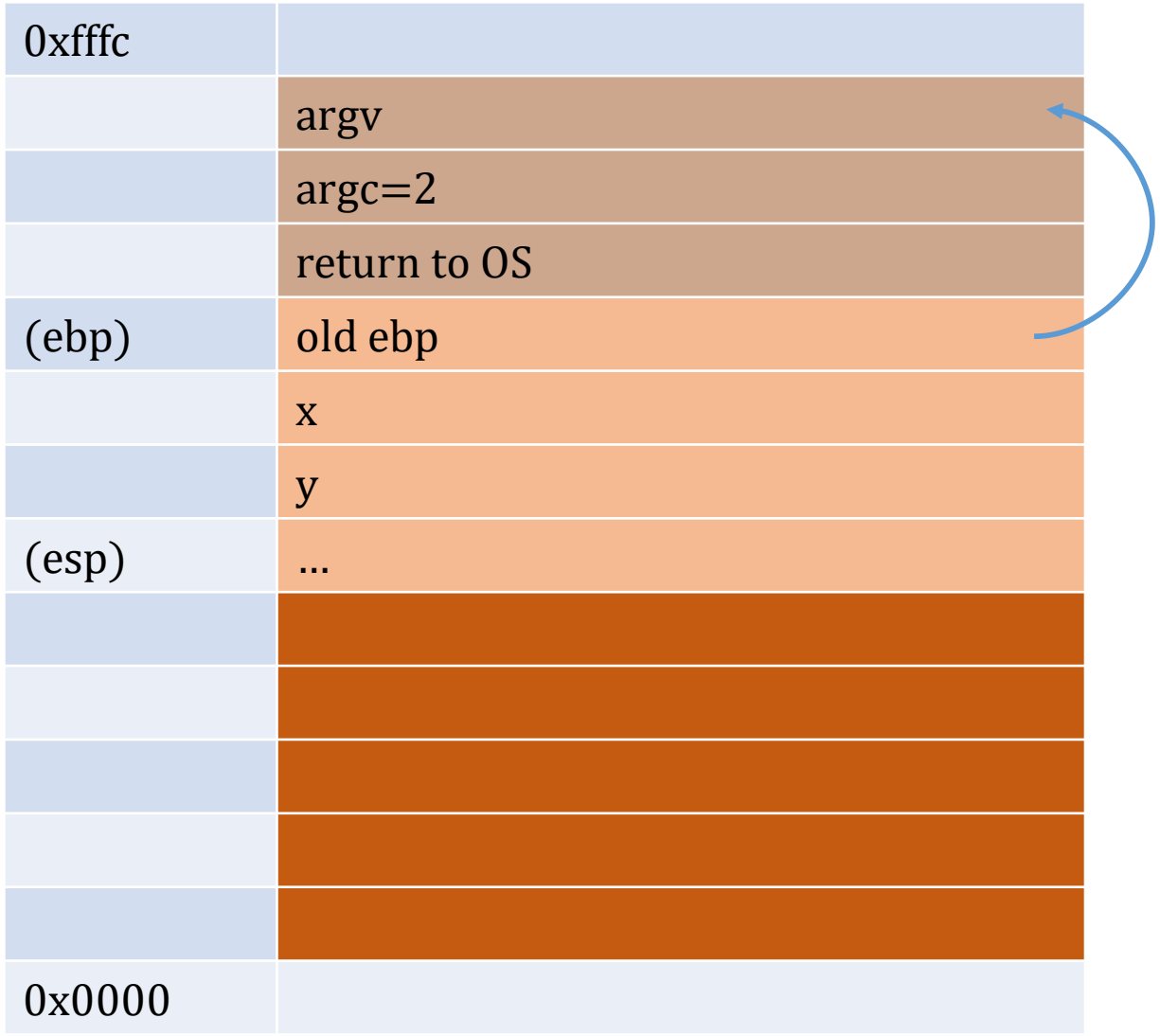
main preamble

```
main:   pushl   %ebp
        movl %esp, %ebp
        andl  $-16, %esp
        subl  $32, %esp
```

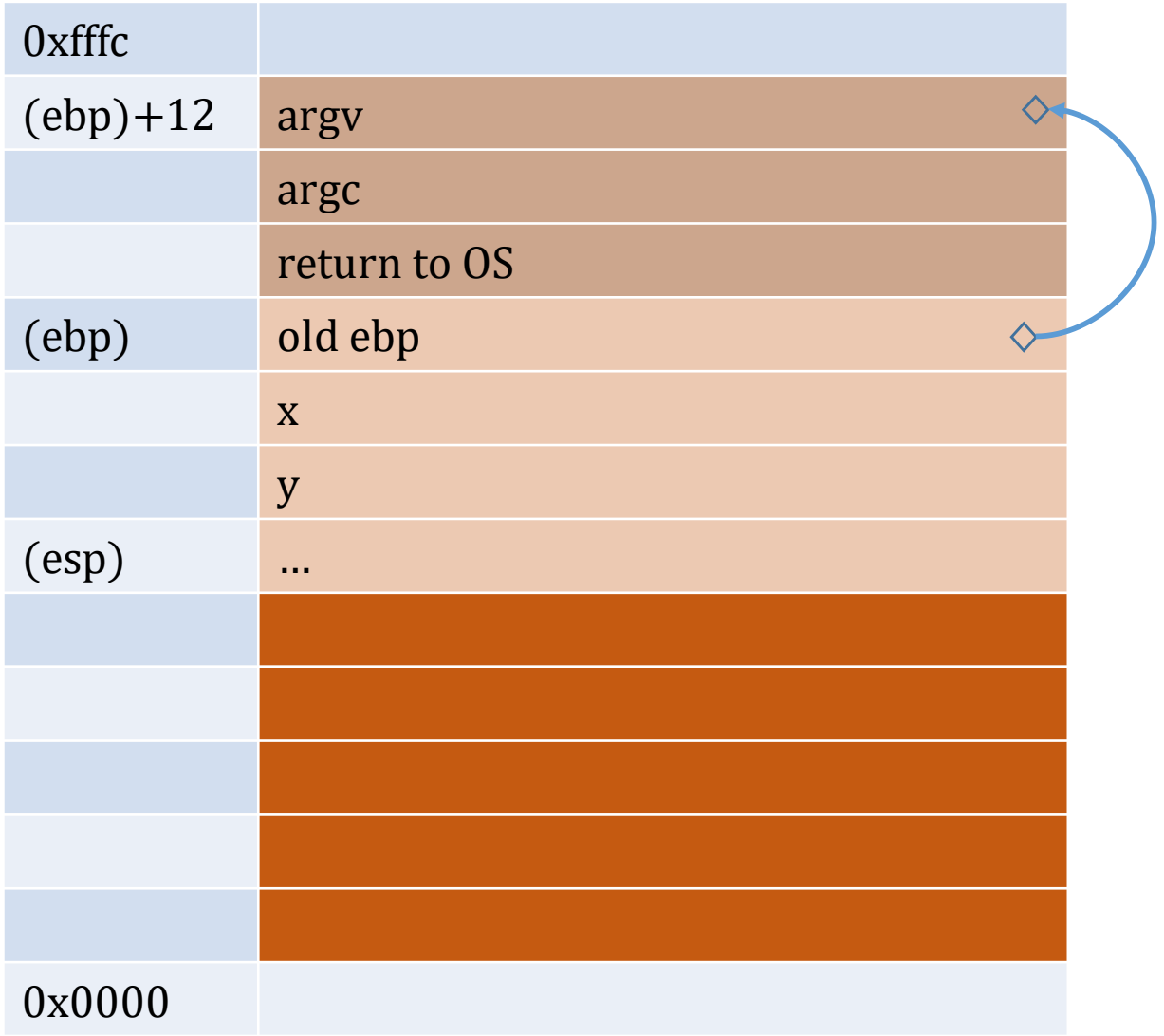


main preamble

```
main:   pushl   %ebp
        movl %esp, %ebp
        andl  $-16, %esp
        subl  $32, %esp
```

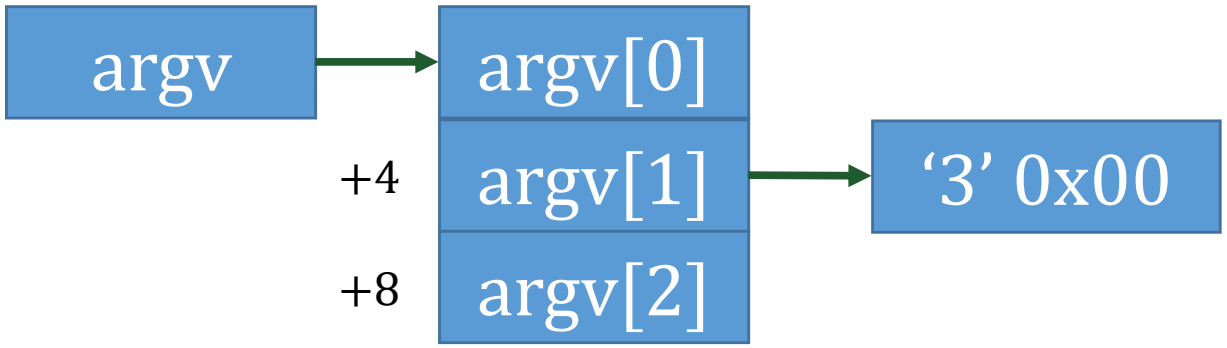


Stack after main preamble…



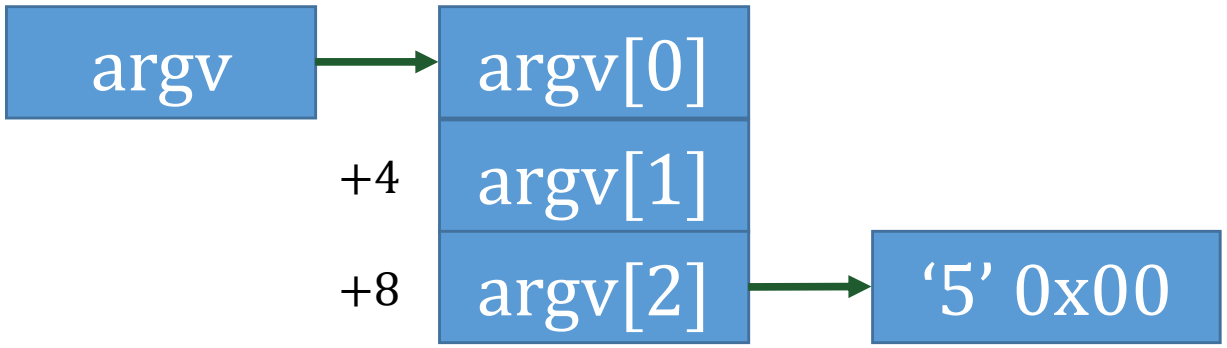
x=atoi(argv[1])

```
movl  12(%ebp), %eax
addl  $4, %eax
movl  (%eax), %eax
movl  %eax, (%esp)
call  atoi
movl  %eax, 28(%esp)
```

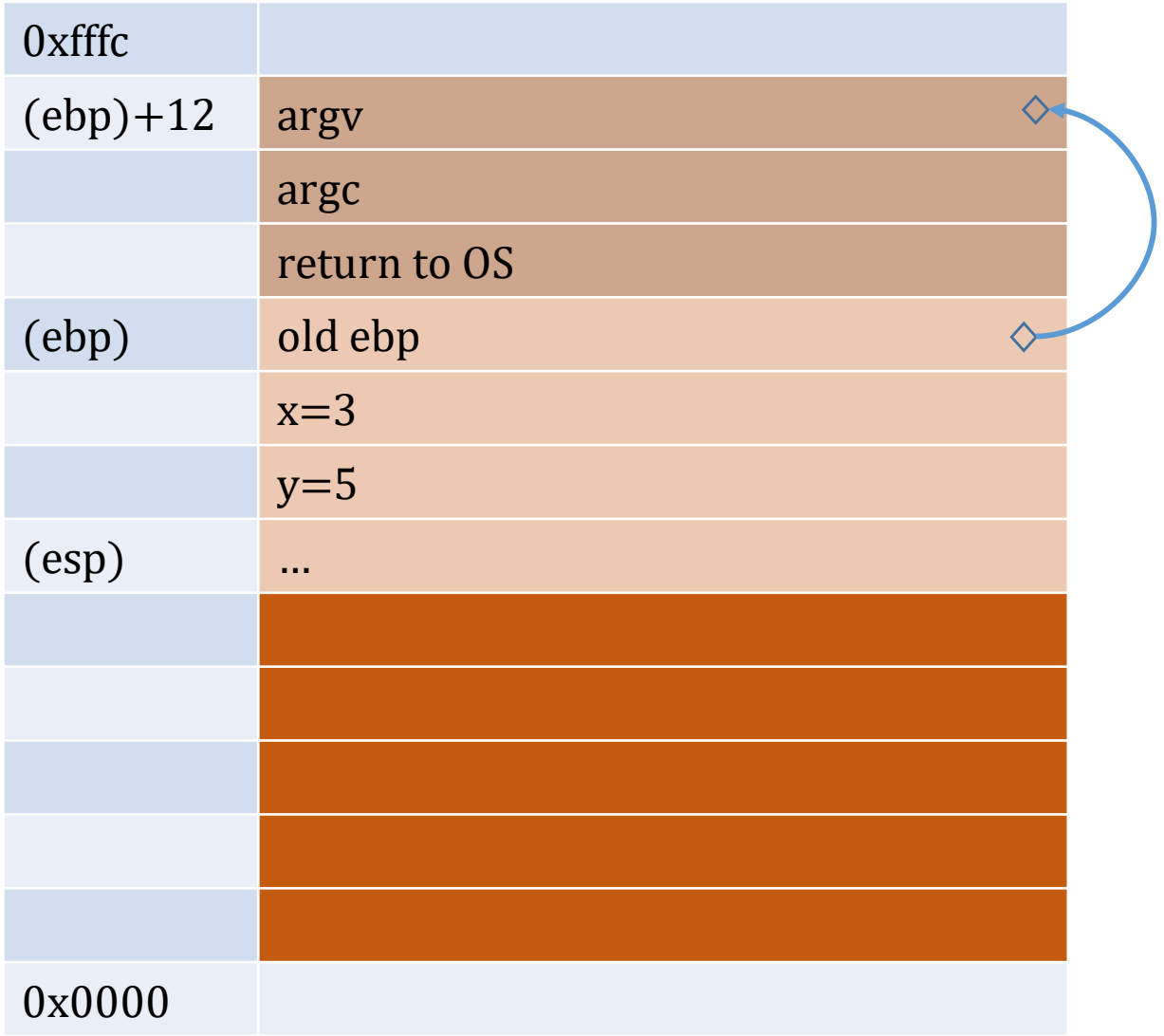


y=atoi(argv[2]);

```
movl  12(%ebp), %eax
addl  $8, %eax
movl  (%eax), %eax
movl  %eax, (%esp)
call  atoi
movl  %eax, 24(%esp)
```

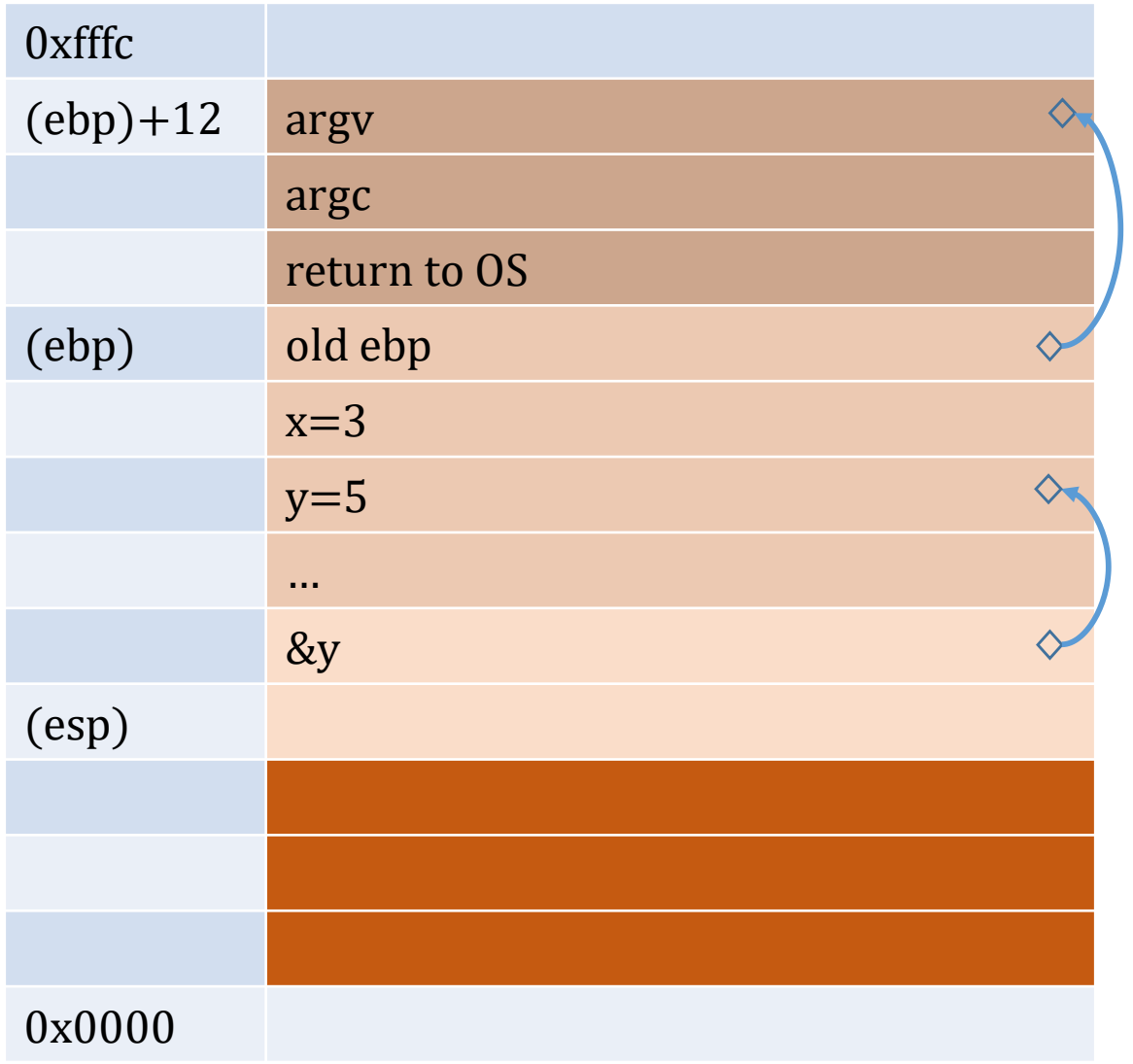


Stack after atoi calls…



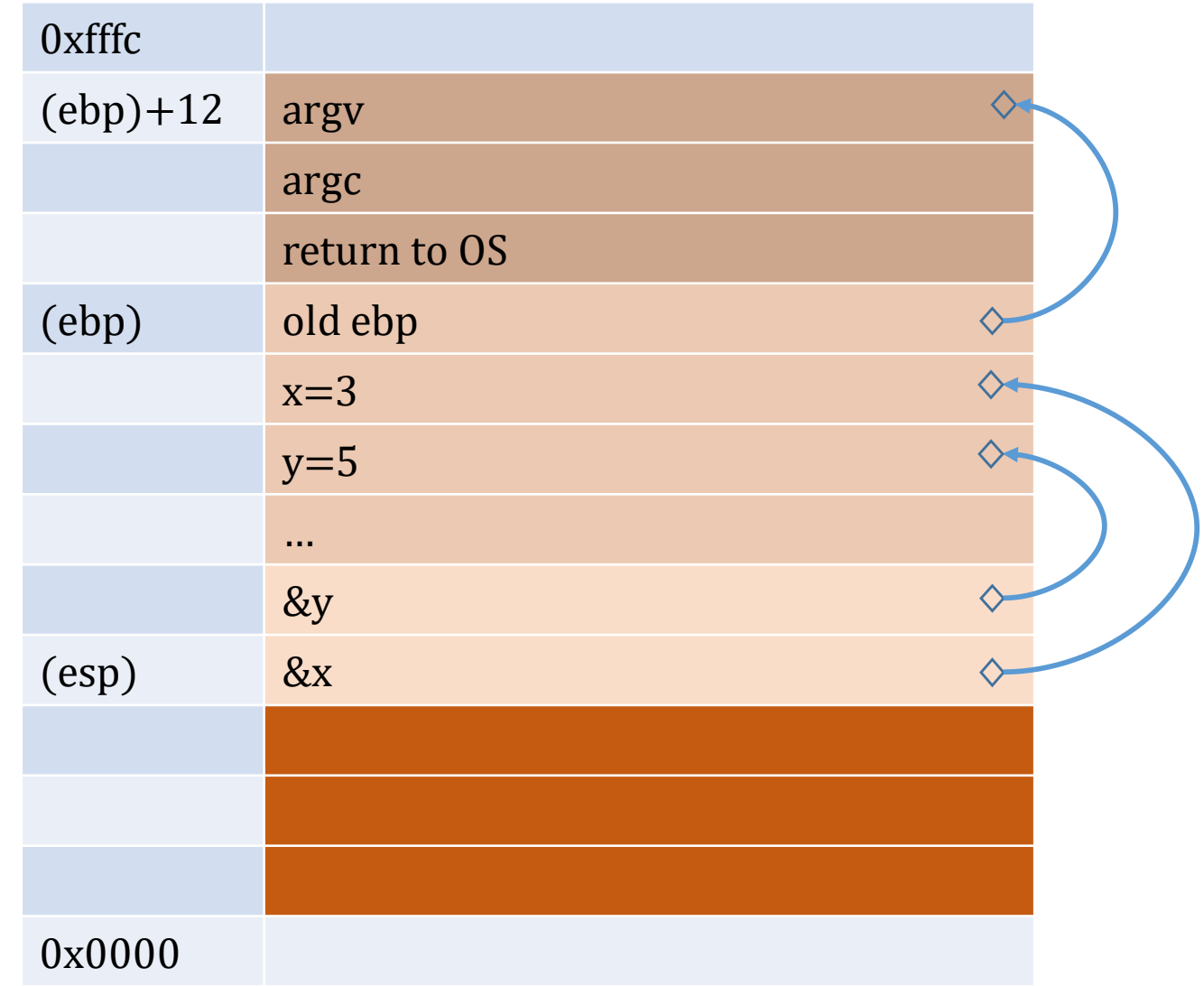
main invocation of swap

```
leal   24(%esp), %eax
movl %eax, 4(%esp)
leal   28(%esp), %eax
movl %eax, (%esp)
call   swap
```

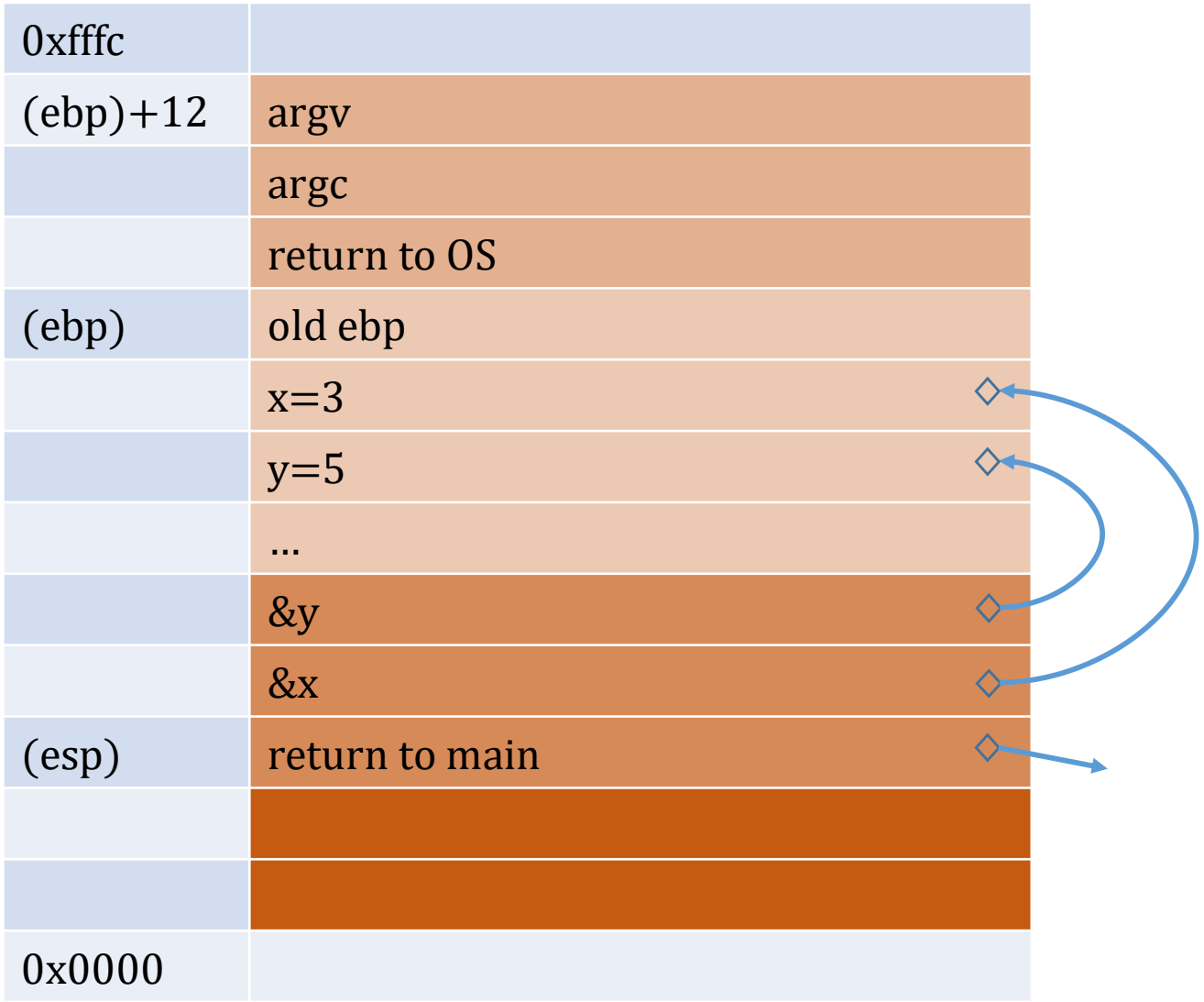


main invocation of swap

```
leal   24(%esp), %eax
movl %eax, 4(%esp)
leal   28(%esp), %eax
movl %eax, (%esp)
call   swap
```



Stack at start of swap…

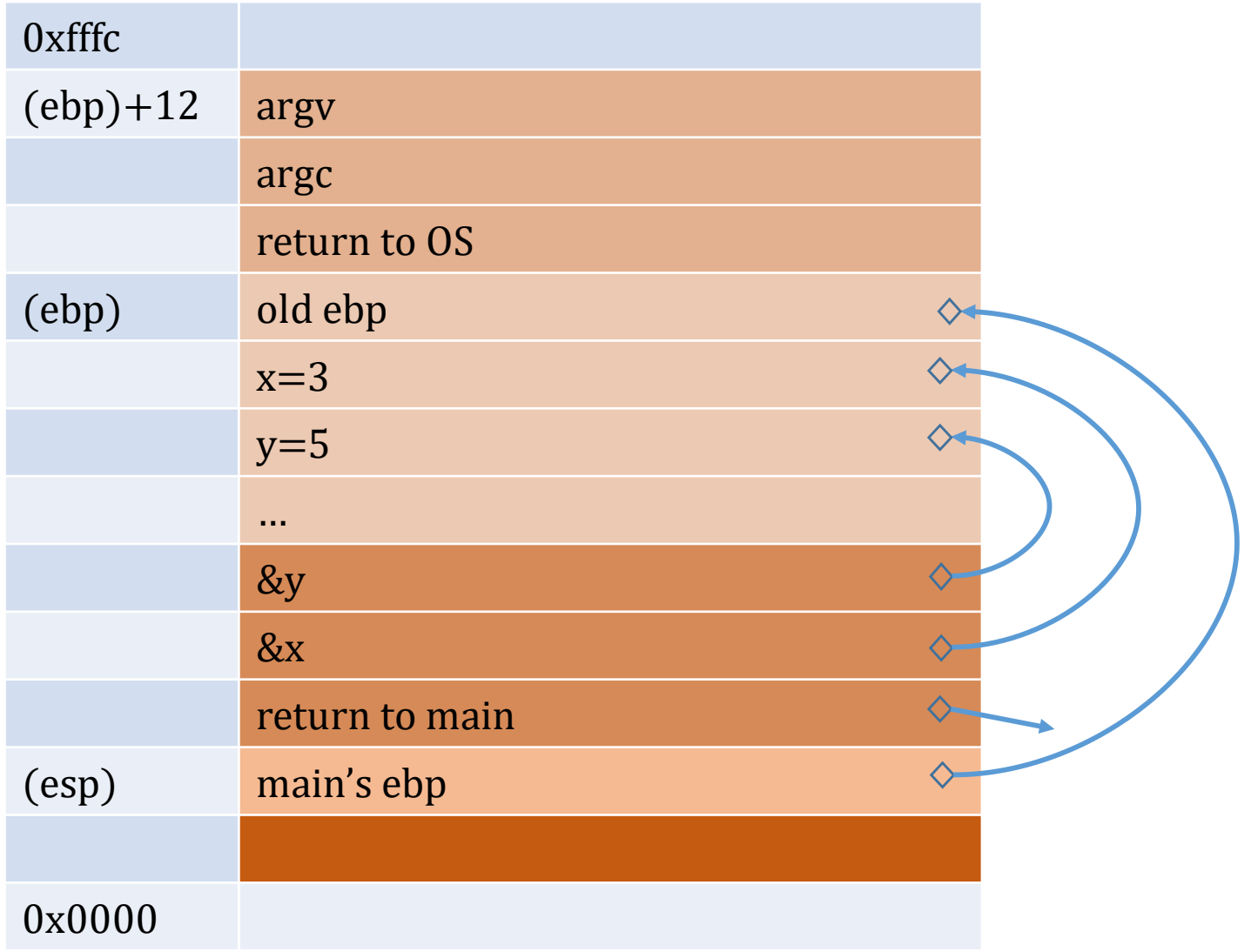


swap prologue

```
swap:  pushl %ebp
       movl  %esp, %ebp
       subl   $16, %esp
```

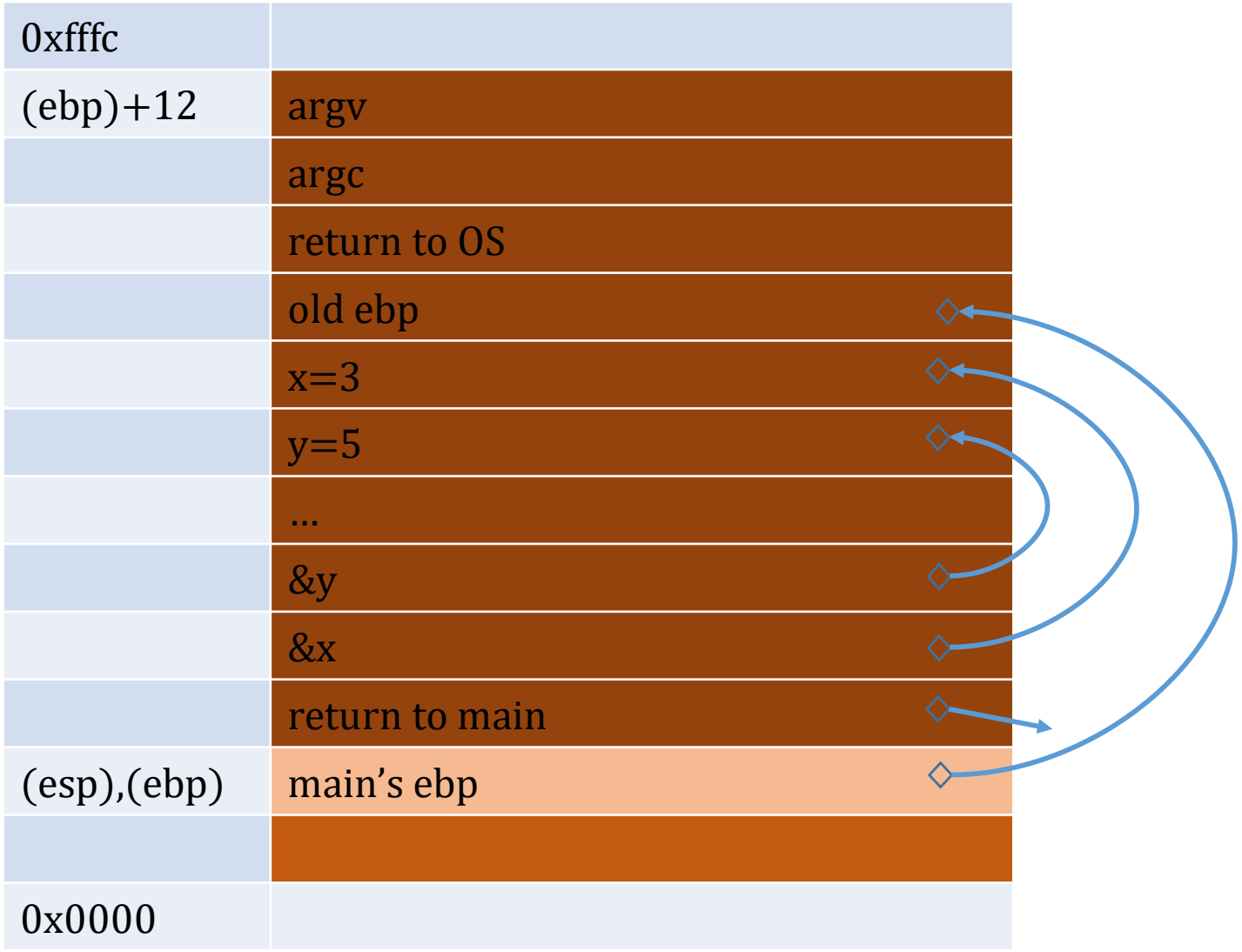
swap prologue

```
swap:  pushl %ebp
       movl  %esp, %ebp
       subl  $16, %esp
```

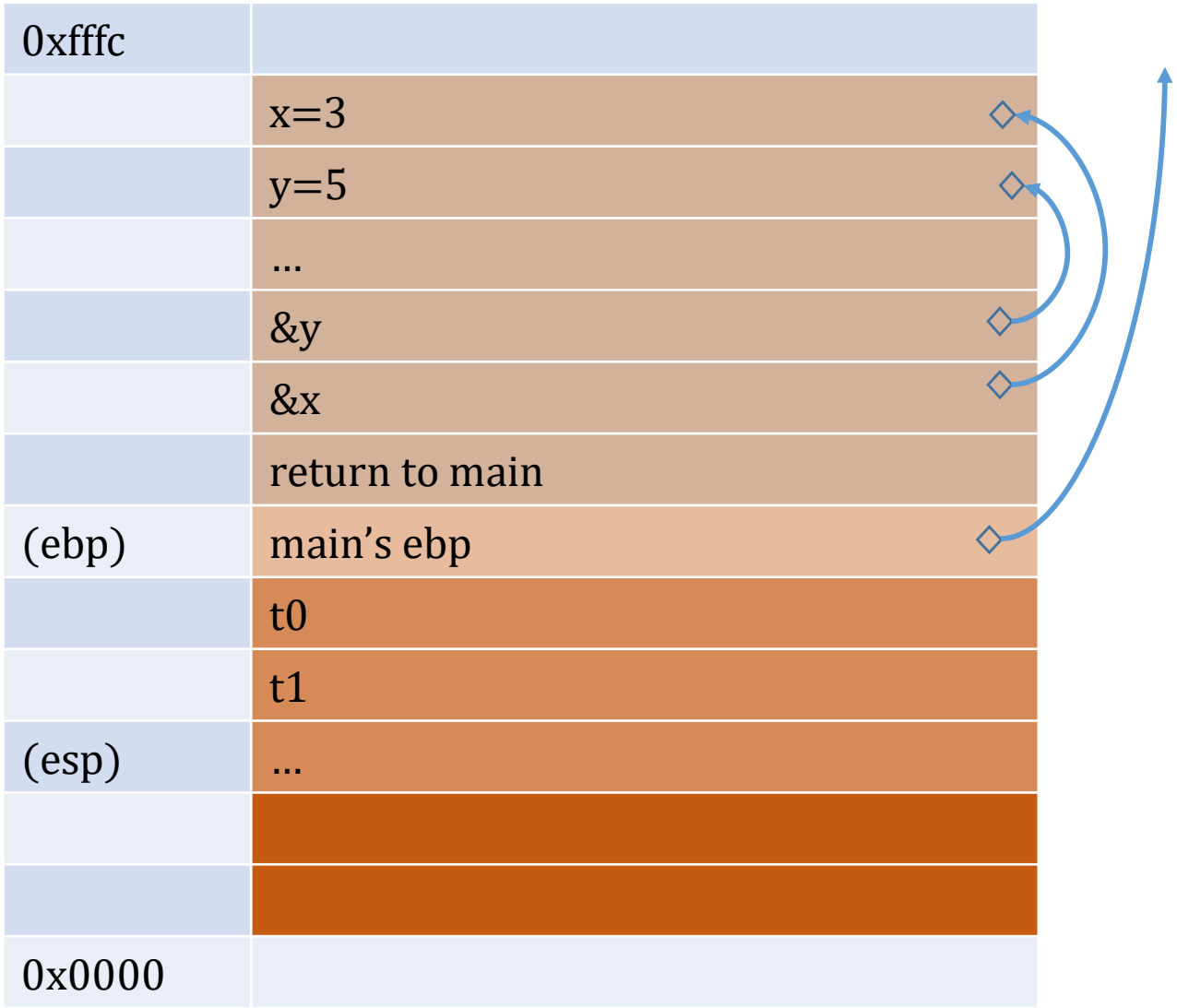


swap prologue

```
swap:  pushl %ebp
       movl  %esp, %ebp
       subl   $16, %esp
```



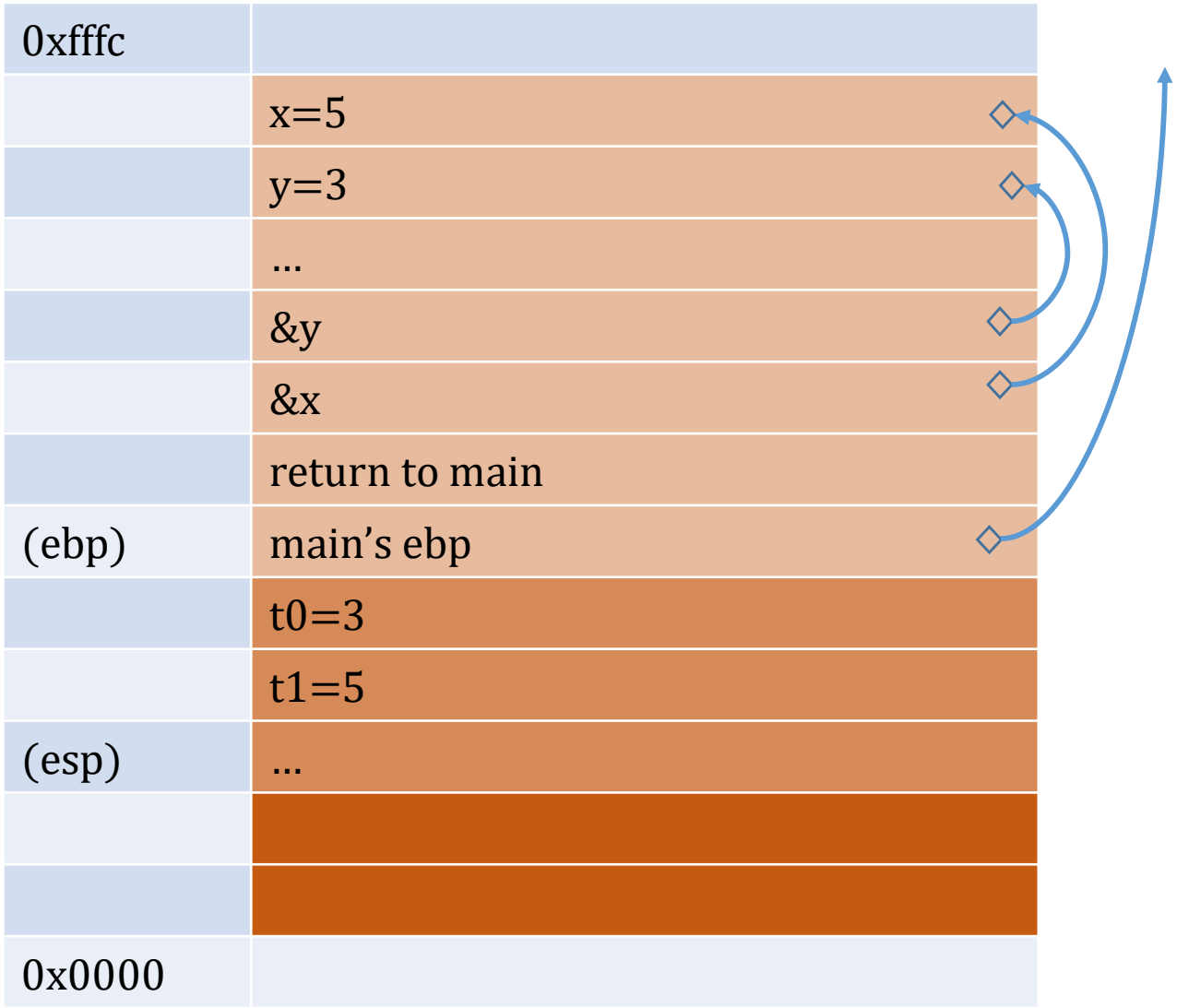
Stack after swap prologue



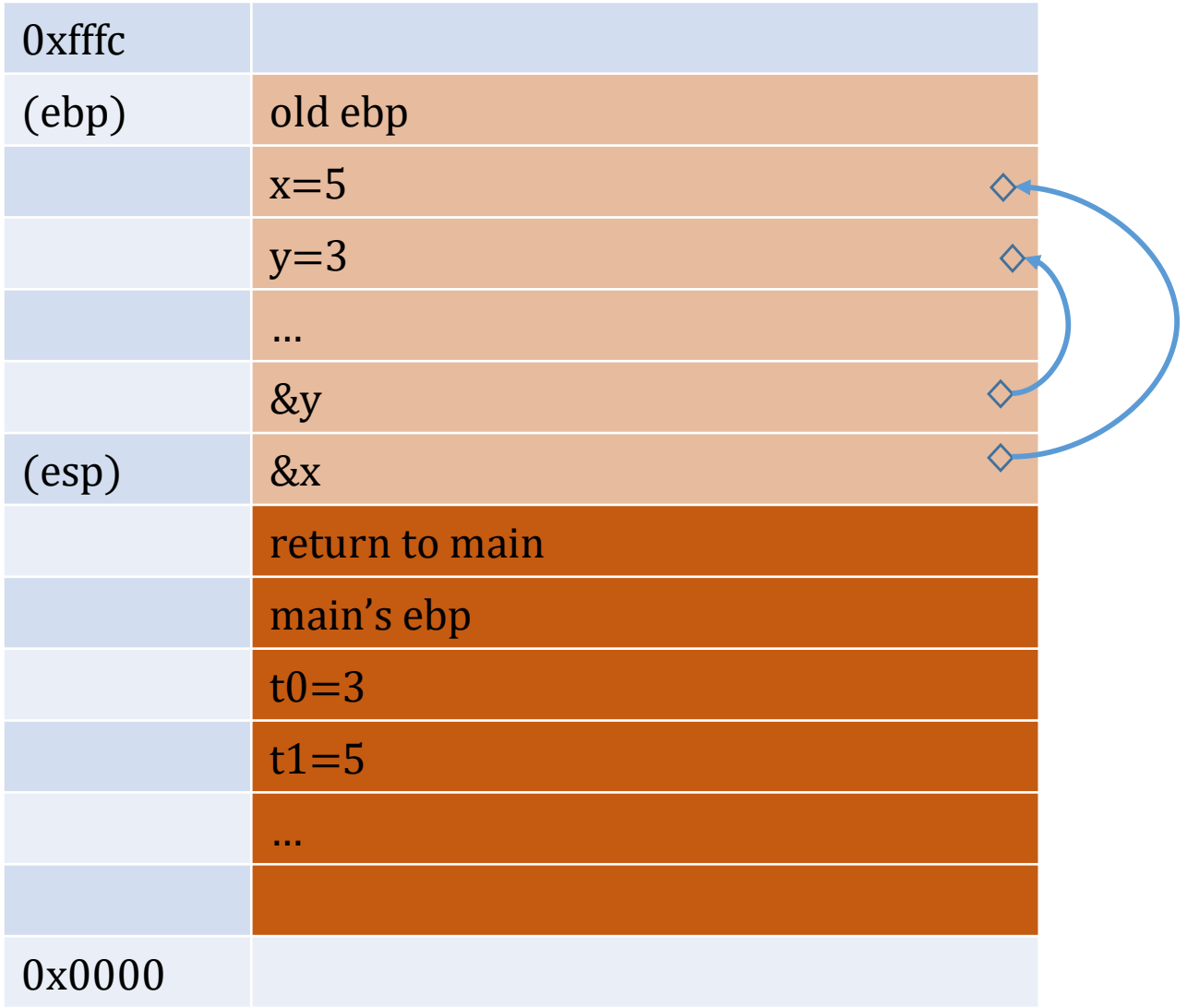
swap guts

```
movl    8(%ebp), %eax       ; eax=&x
movl    (%eax), %eax        ; eax=*(&x)=x=3
movl    %eax, -4(%ebp)      ; t0=eax=3
movl    12(%ebp), %eax      ; eax=&y
movl    (%eax), %eax        ; eax=*(&y)= y=5
movl    %eax, -8(%ebp)      ; t1=eax=5
movl    8(%ebp), %eax       ; eax=&x
movl    -8(%ebp), %edx      ; edx=t1=5
movl    %edx, (%eax)        ; *eax (or x)=edx=5
movl    12(%ebp), %eax      ; eax=&y
movl    -4(%ebp), %edx      ; edx=t0=3
movl    %edx, (%eax)        ; *eax (or y)=3
```

Stack after swap guts



Stack after swap return



IA64 Register Conventions

- First four parameters are contained in registers:
  %RDI, %RSI, %RCX, %RDX
- Local Variables kept in registers:
  e.g. %RBP, %RBX
- Return value: %RAX
- Bottom of Stack: %RSP

IA64 Stack Usage

- No longer use %RBP to keep track of “top of stack frame”
- Require %RSP is the same before/after a call (among others)
- Use stack to keep saved state
- Use stack for parameters if there are more than 4
- Use stack for local variables if you run out of registers
- Use stack for return address
- Only use stack if necessary

64-bit code for swap

```
swap:

   movl   (%rdi), %edx
   movl   (%rsi), %eax
   movl   %eax, (%rdi)
   movl   %edx, (%rsi)

   ret
```

Set Up

Body

Finish

- Operands passed in registers
  - 64-bit pointers
  - First (xp) in %rdi,
  - second (yp) in %rsi
- All local variables in registers
  - t0 in %eax, t1 in %edx
- No stack operations required
- 32-bit data
  - Data held in registers %eax and %edx
  - movl operation