

Stack Animations

Sample Code

- Silly code (we wouldn't actually use these functions for anything other than using to study how the stack is used).
 - Example 1: recursive function that is called and passed 2 parameter values. No local variables!
 - Example 2: functions that have no parameters, but have some local variables.
 - Example 3: crash and burn.

Recursive Function add ()

```
// This function computes the sum of x and y
// using the property:  $x+y = (x+1) + (y-1)$ 
// and the special case when  $y=0$  ( $x+y=x$ )

unsigned int add(unsigned int x, unsigned int y)
{
    if (y==0)
        return(x) ;
    else
        return(add(x+1,y-1)) ;
}

int main() {
    add(4,3) ;
}
```

add () Assembly Code

_add:

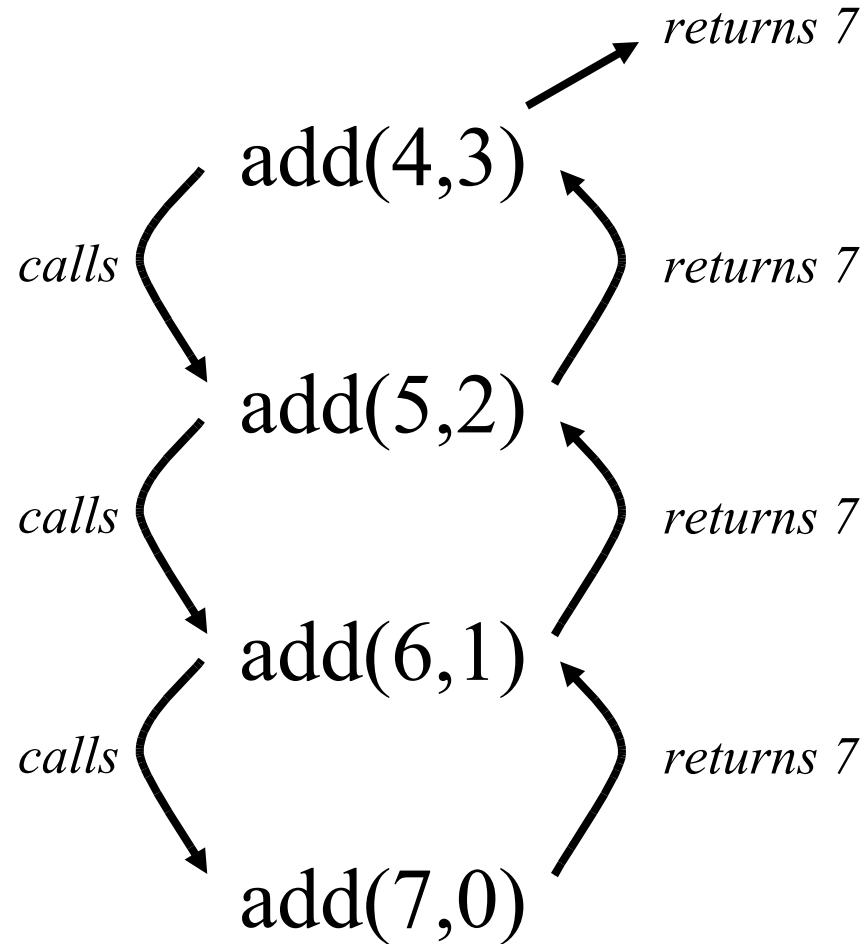
```
    pushl %ebp
    movl  %esp,%ebp
    movl  8(%ebp),%eax    # eax is x
    movl  12(%ebp),%edx   # edx is y

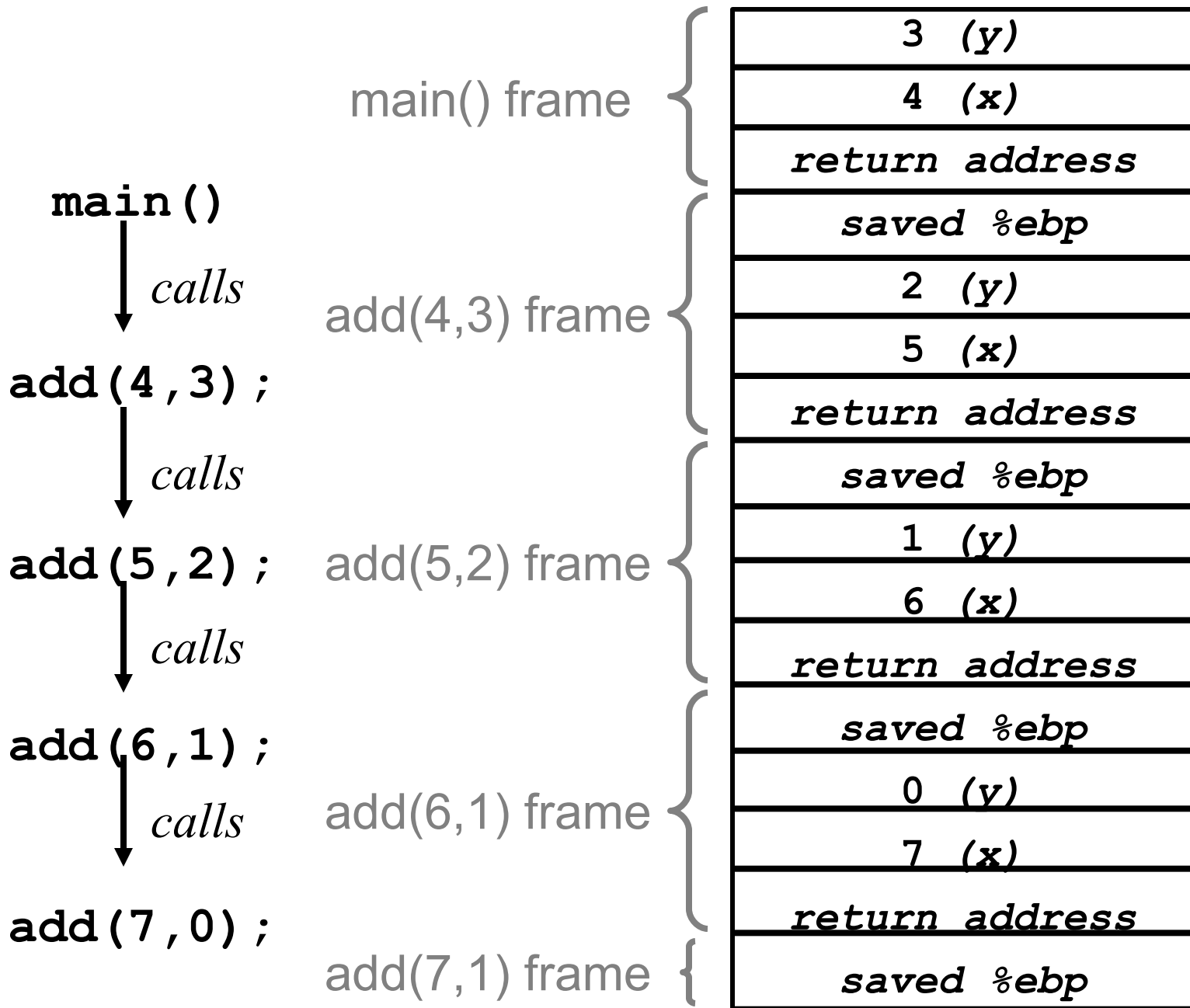
    testl %edx,%edx      # is y == 0 ?
    je    _add_ret       # yes - jump
    incl  %eax            # eax = x+1
    decl  %edx            # edx = y-1
    pushl %edx            # pass y-1 to add
    pushl %eax            # pass x+1 to add
    call  _add2           # call add
```

_add_ret:

```
    movl  %ebp,%esp
    popl  %ebp
    ret
```

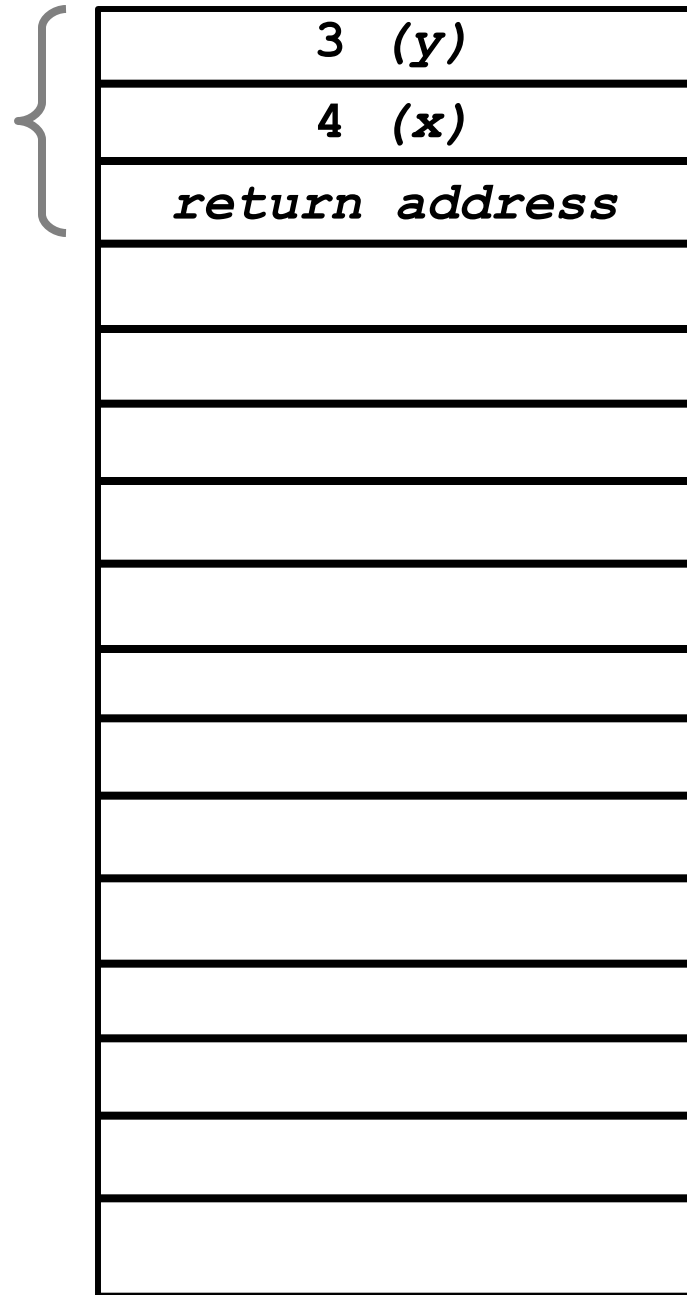
add (4 , 3) ; call sequence





`main()`
↓ *calls*
`add(4, 3);`

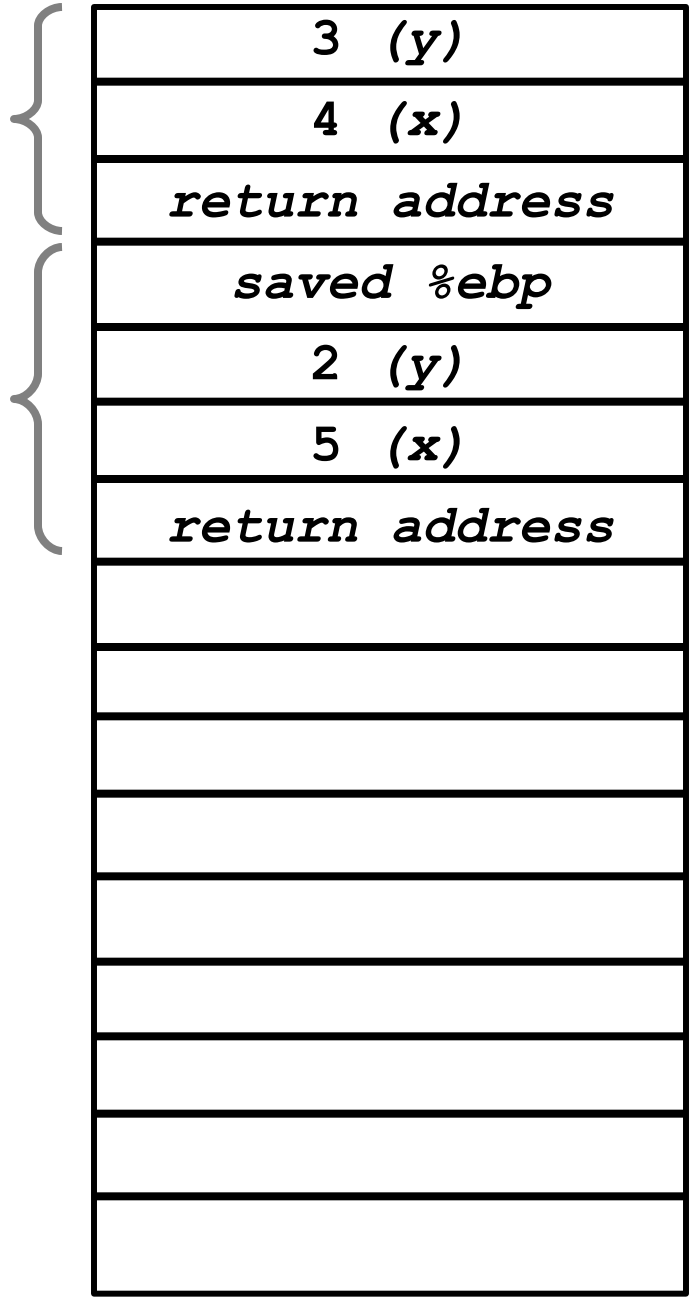
main() frame



`main()`
↓ *calls*
`add(4, 3);`
↓ *calls*
`add(5, 2);`

main() frame

add(4,3) frame



```

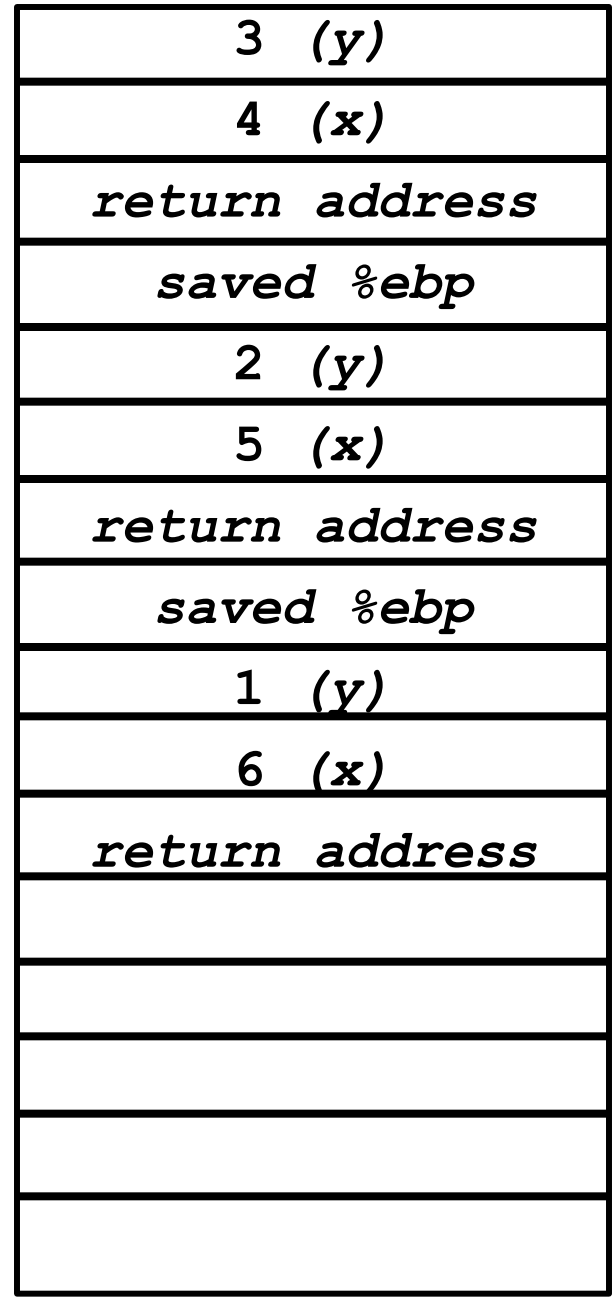
main ( )
  ↓ calls
add ( 4 , 3 ) ;
  ↓ calls
add ( 5 , 2 ) ;
  ↓ calls
add ( 6 , 1 ) ;

```

main() frame

add(4,3) frame

add(5,2) frame



```

main()
  ↓ calls
add(4,3);
  ↓ calls
add(5,2);
  ↓ calls
add(6,1);
  ↓ calls
add(7,0);

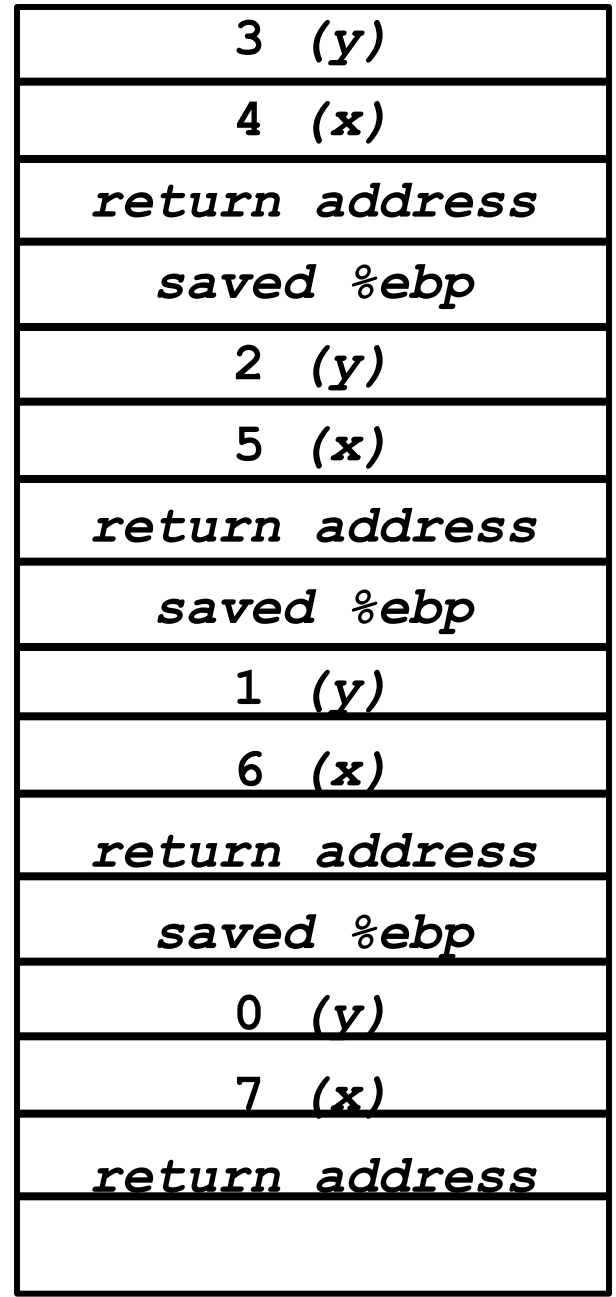
```

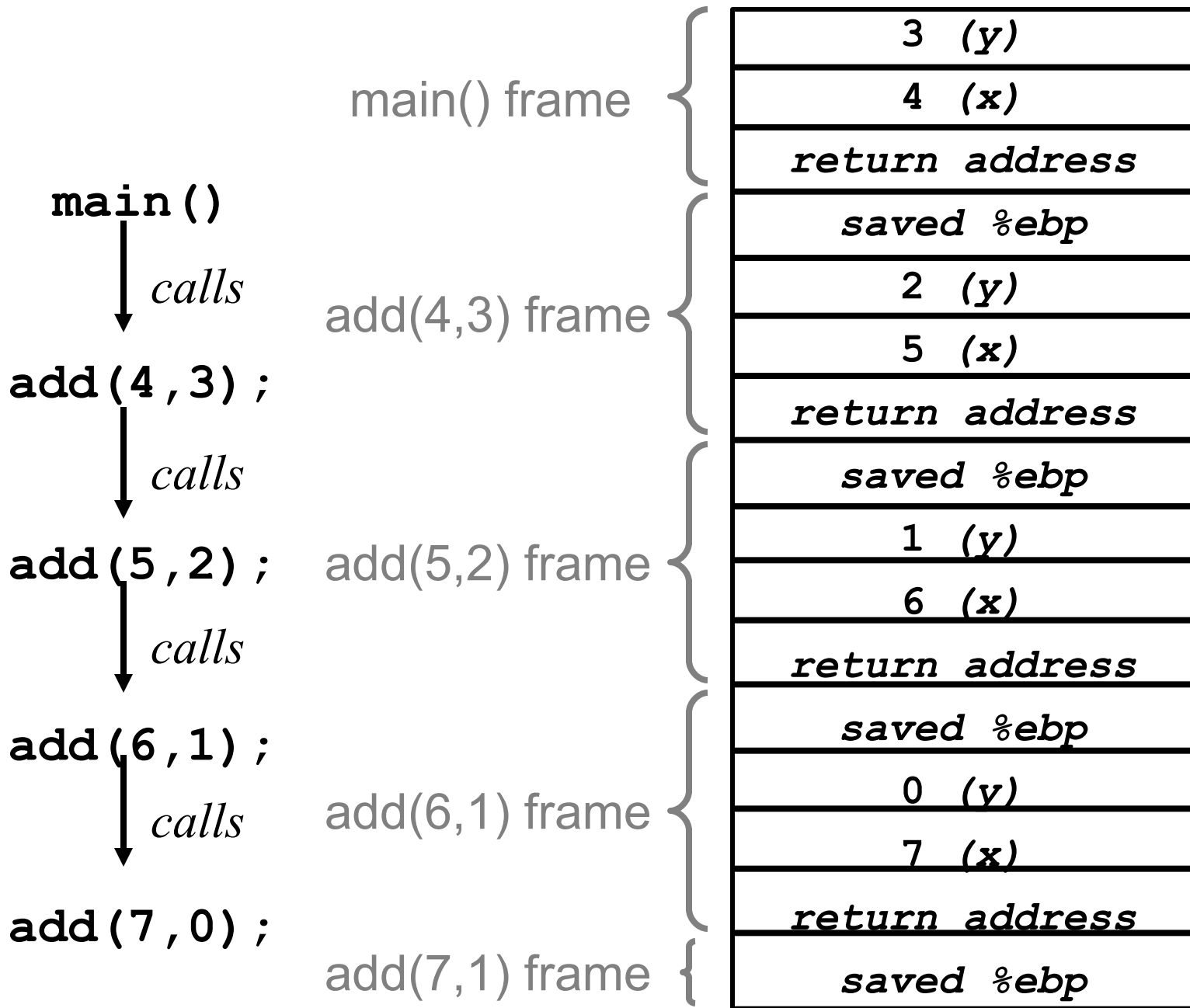
main() frame

add(4,3) frame

add(5,2) frame

add(6,1) frame





Local Variables

- Functions that have no parameters.
- Two local variables.

blah() and foo()

```
int blah() {
    int blah_a=1;
    int blah_b=2;
    return(blah_a + blah_b);
}

int foo() {
    int foo_a=10;
    int foo_b=20;
    return(blah()+foo_a+foo_b);
}

int main() {
    foo();
}
```

blah () Assembly Code

```
_blah:  
  pushl %ebp  
  movl %esp,%ebp  
  subl $8,%esp           # allocate space for locals  
  movl $1,-4(%ebp)       # blah_a = 1  
  movl $2,-8(%ebp)       # blah_b = 2  
  movl -4(%ebp),%eax     # eax = blah_a  
  movl -8(%ebp),%ecx     # ecx = blah_b  
  leal (%ecx,%eax),%edx # edx = blah_a + blah_b  
  movl %edx,%eax  
  movl %ebp,%esp  
  popl %ebp  
  ret
```

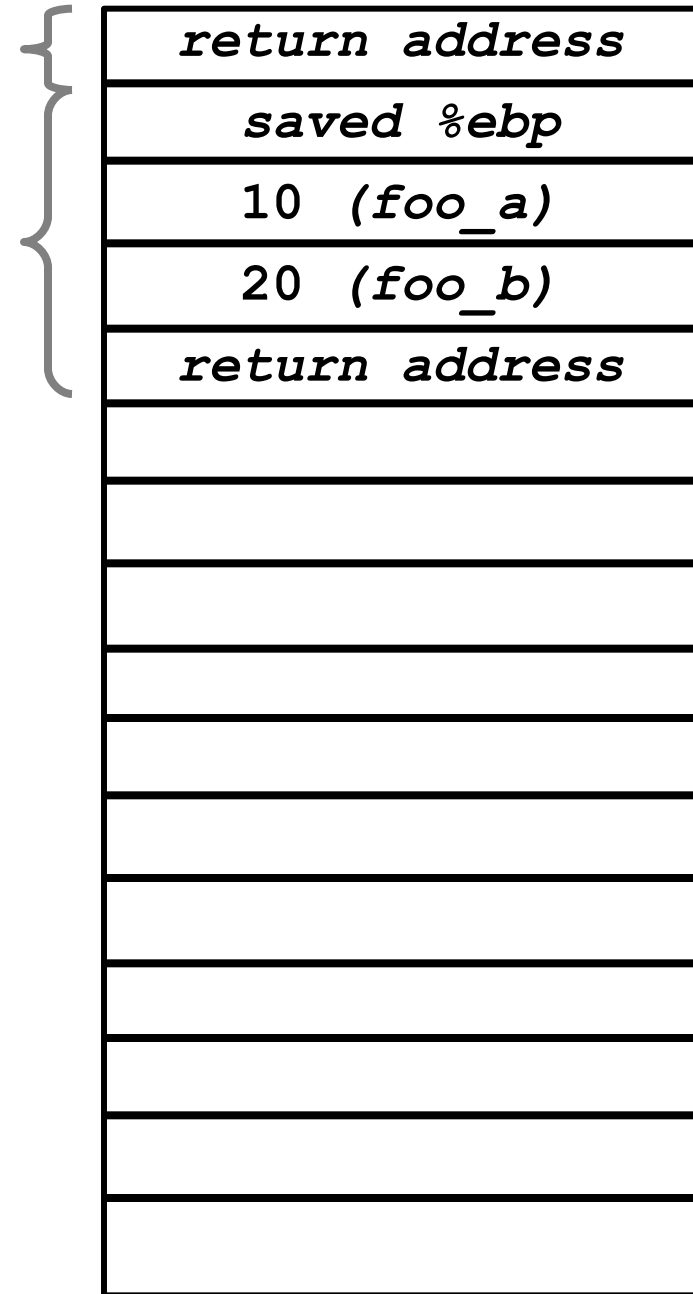
foo () Assembly Code

```
_foo:  
  pushl %ebp  
  movl %esp,%ebp  
  subl $8,%esp           # allocate space for locals  
  movl $10,-4(%ebp)      # foo_a = 10  
  movl $20,-8(%ebp)      # foo_b = 20  
  call _blah             # eax = blah()  
  addl -4(%ebp),%eax      # eax = blah()+foo_a  
  addl -8(%ebp),%eax      # eax = blah()+foo_a+foo_b  
  movl %ebp,%esp  
  popl %ebp  
  ret
```


main()
↓ *calls*
foo();
↓ *calls*
blah();

main() frame

foo() frame



```

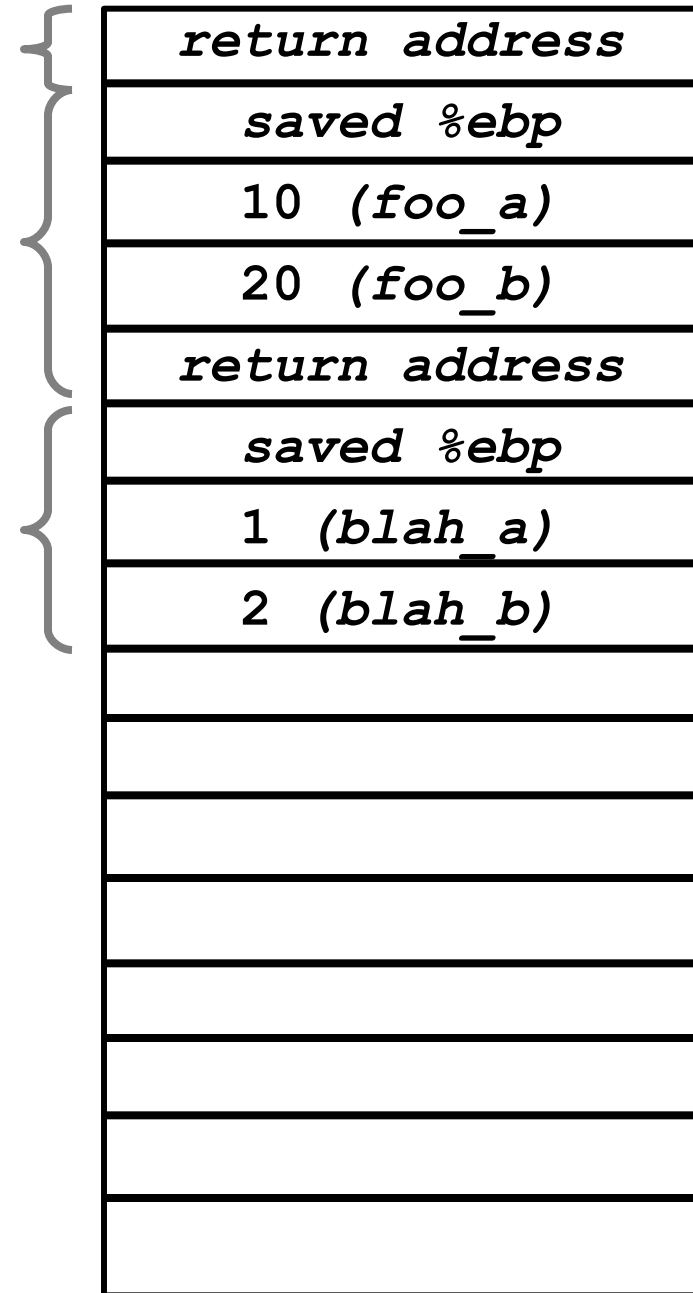
main ( )
  ↓ calls
foo ( ) ;
  ↓ calls
blah ( ) ;

```

main() frame

foo() frame

blah() frame



Crash ()

```
void crash() {  
    int x;  
    int *ptr=&x;  
  
    *ptr++=0;  
    *ptr++=0;  
    *ptr++=0;  
}
```

_crash:

```
    pushl %ebp
    movl %esp,%ebp
    subl $8,%esp           # allocate space for x,ptr
    leal -4(%ebp),%eax     # %eax = &x
    movl %eax,-8(%ebp)     # ptr = &x
    movl -8(%ebp),%eax     # %eax = ptr
    movl $0,(%eax)        # *ptr = 0
    addl $4,-8(%ebp)      # ptr = ptr + 4
    movl -8(%ebp),%eax     # %eax = ptr
    movl $0,(%eax)        # *ptr = 0
    addl $4,-8(%ebp)      # ptr = ptr + 4
    movl -8(%ebp),%eax     # %eax = ptr
    movl $0,(%eax)        # *ptr = 0
    addl $4,-8(%ebp)      # ptr = ptr +4
```

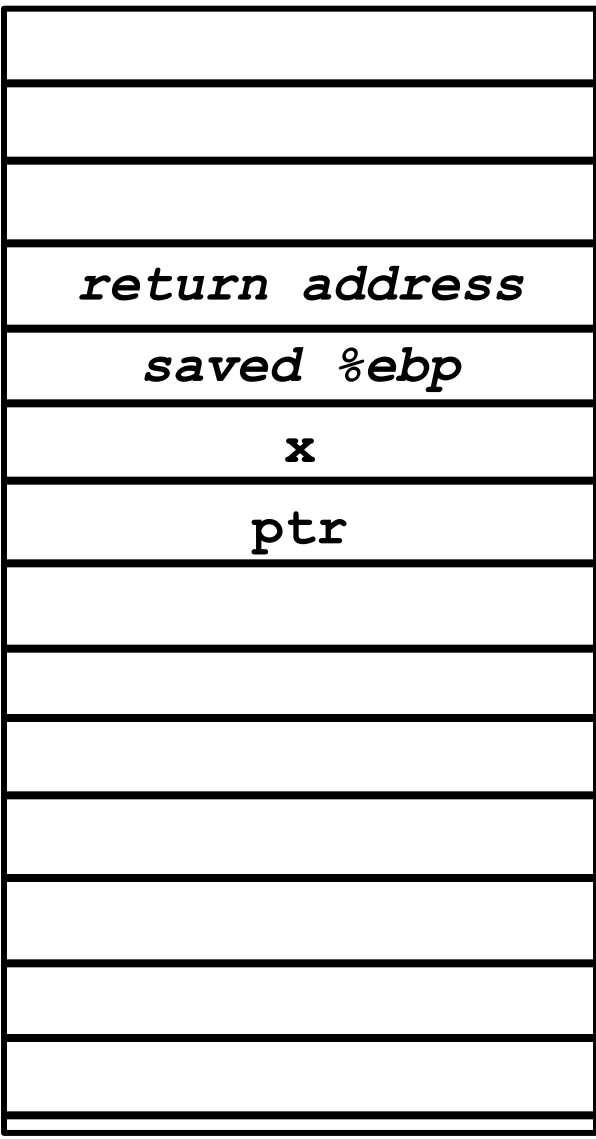
L10:

```
    movl %ebp,%esp
    popl %ebp
    ret
```

caller's frame

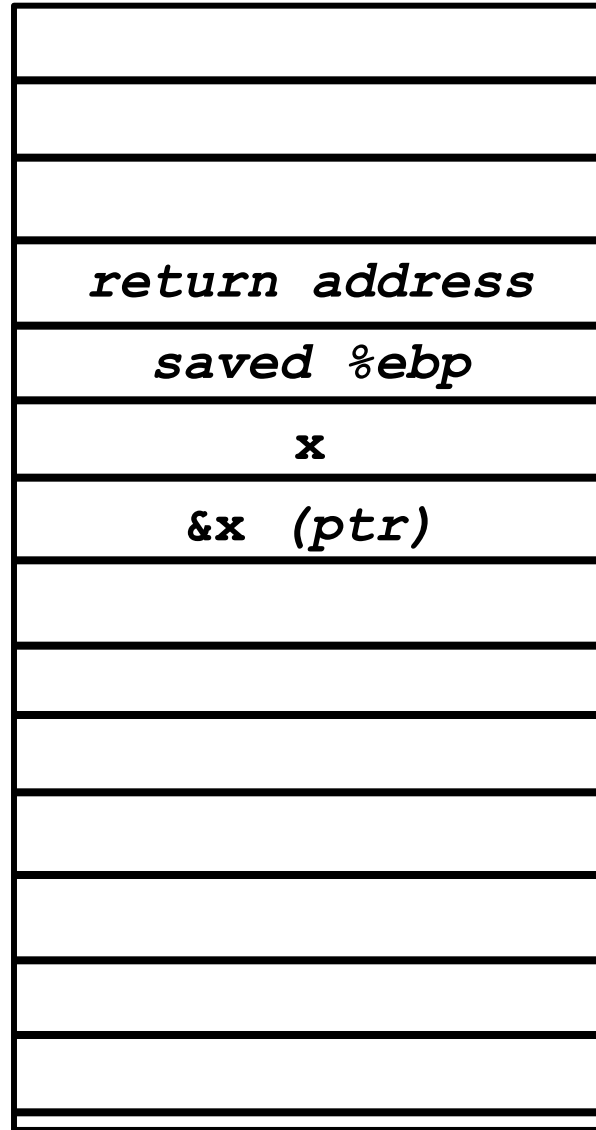


crash() frame



caller's frame

crash() frame

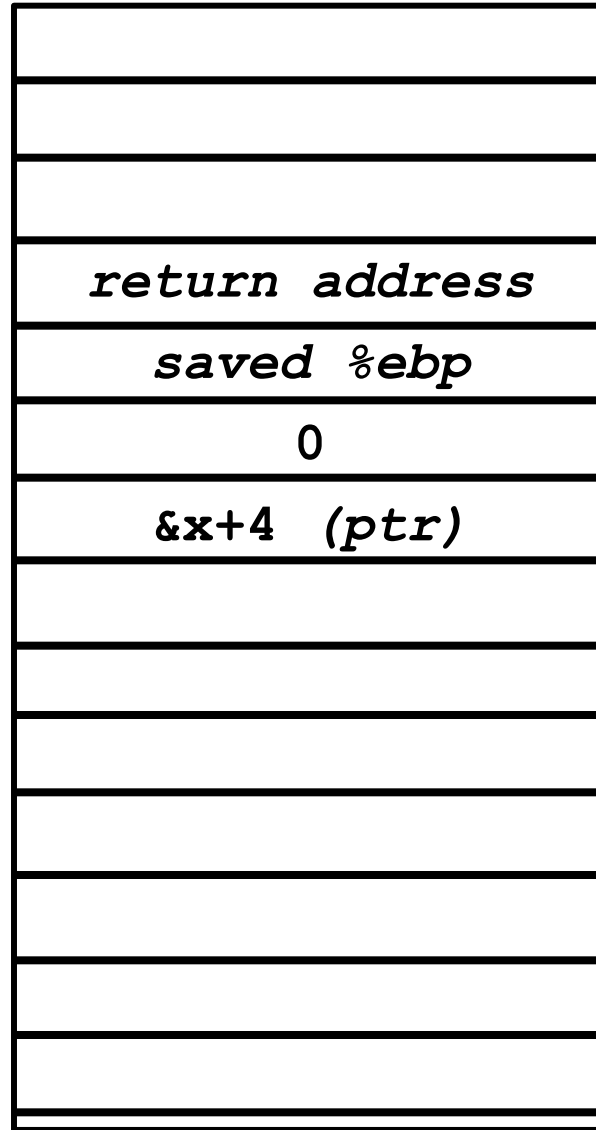


ptr

```
int *ptr=&x;
```

caller's frame

crash() frame

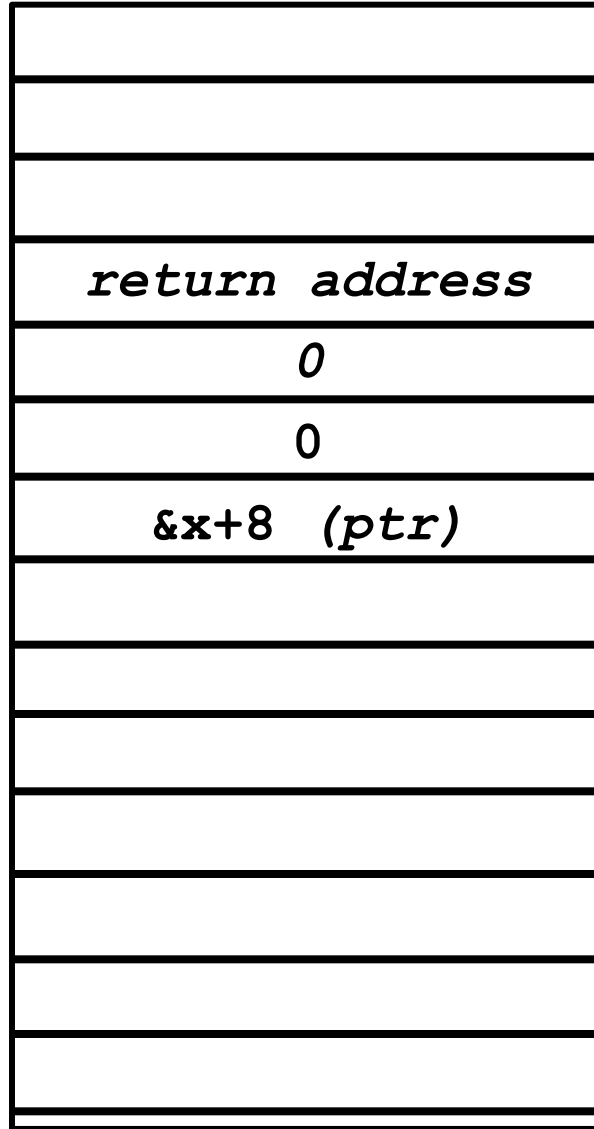


ptr

```
*ptr++=0;
```

caller's frame

crash() frame

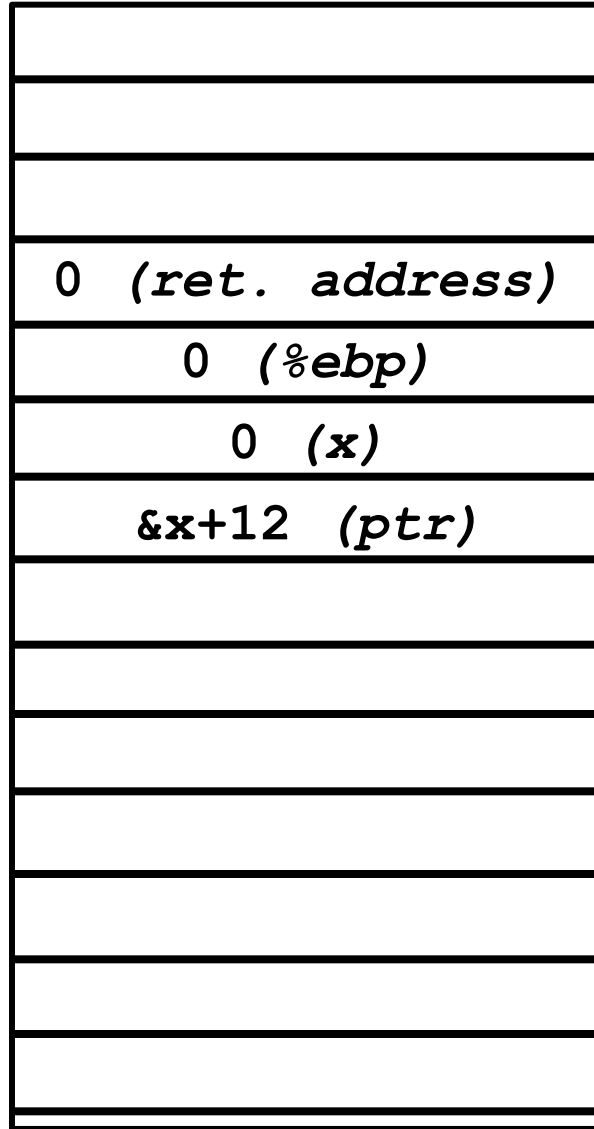


← *ptr*

```
*ptr++=0;
```

caller's frame

crash() frame



ptr

```
*ptr++=0;
```