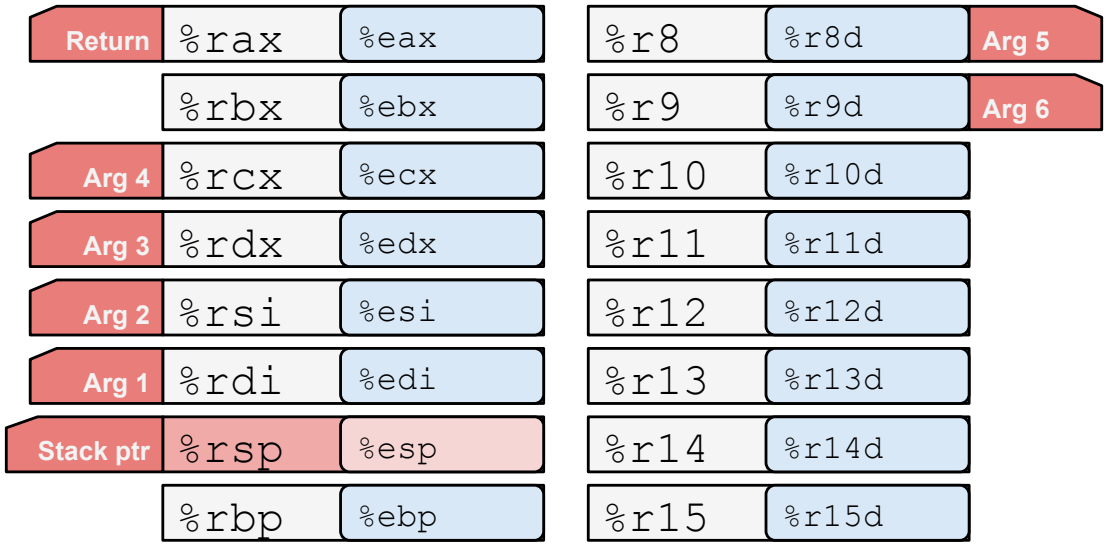
Procedure Call Registers



Switch Statements

```
long switch_eg
   (long x, long y, long z)
{
    long w = 1;
    switch(x) {
    case 1:
        w = y*z;
        break;
    case 2:
        w = y/z;
        /* Fall Through */
    case 3:
        w += z;
        break;
    case 5:
    case 6:
        w -= z;
        break;
    default:
        w = 2;
    }
    return w;
}
```

Jump Tables

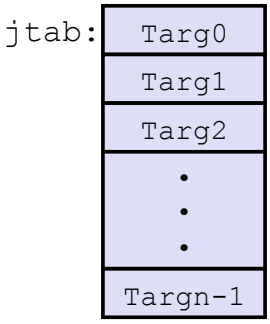
Switch Form

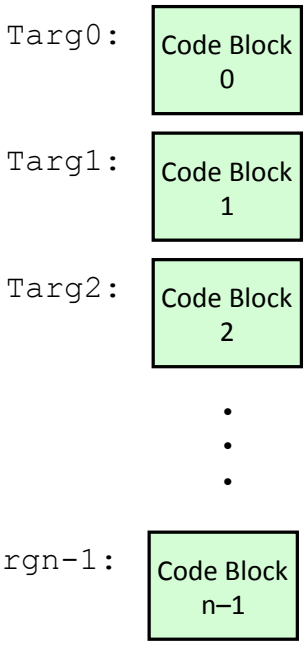
```
switch(x) {
  case val_0:
    Block 0
  case val_1:
    Block 1
    • • •
  case val_n-1:
    Block n–1
}
```

Translation (Extended C)

```
goto *jtab[x];
```

Jump Table





Switch Example

```
long switch_eg
   (long x, long y, long z)
{
    long w = 1;
    switch(x) {
    case 1:
        w = y*z;
        break;
    case 2:
        w = y/z;
        /* Fall Through */
    case 3:
        w += z;
        break;
    case 5:
    case 6:
        w -= z;
        break;
    default:
        w = 2;
    }
    return w;
}
```

Jump table

```
.section    .rodata
  .align 8
.L4:
  .quad     .L8 # x = 0
  .quad     .L3 # x = 1
  .quad     .L5 # x = 2
  .quad     .L9 # x = 3
  .quad     .L8 # x = 4
  .quad     .L7 # x = 5
  .quad     .L7 # x = 6
```

Register	Use(s)
%rdi	Argument x
%rsi	Argument y
%rdx	Argument z
%rax	Return value

```
switch_eg:
    movq    %rdx, %rcx
    cmpq    $6, %rdi        # x:6
    ja      .L8             # Use default
    jmp     *.L4(,%rdi,8) # goto *JTab[x]
```

Example Jump Table

Jump table

```
.section    .rodata
  .align 8
.L4:
  .quad     .L8  # x = 0
  .quad     .L3  # x = 1
  .quad     .L5  # x = 2
  .quad     .L9  # x = 3
  .quad     .L8  # x = 4
  .quad     .L7  # x = 5
  .quad     .L7  # x = 6
```

```
    switch(x) {
    case 1:       // .L3
        w = y*z;
        break;
    case 2:       // .L5
        w = y/z;
        /* Fall Through */
    case 3:       // .L9
        w += z;
        break;
    case 5:
    case 6:       // .L7
        w -= z;
        break;
    default:      // .L8
        w = 2;
    }
```


Code Blocks

```
    long w = 1;
    switch(x) {
    case 1:       // .L3
        w = y*z;
        break;
    case 2:       // .L5
        w = y/z;
        /* Fall Through */
    case 3:       // .L9
        w += z;
        break;
    case 5:
    case 6:       // .L7
        w -= z;
        break;
    default:      // .L8
        w = 2;
    }
    return w;
```

Register	Use(s)
%rdi	Argument x
%rsi	Argument y
%rdx	Argument z
%rax	Return value

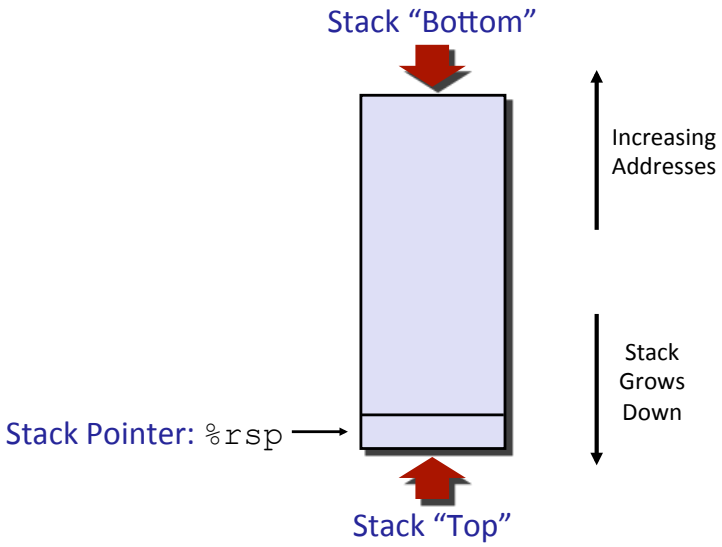
```
switch_eg:
    movq    %rdx, %rcx
    cmpq    $6, %rdi       # x:6
    ja      .L8            # Use default
    jmp     *.L4(,%rdi,8) # goto *JTab[x]
```

```
.L3:                       # Case 1
   movq     %rsi, %rax     # y
   imulq    %rdx, %rax     # y*z
   ret
.L5:                       # Case 2
   movq     %rsi, %rax
   cqto
   idivq    %rcx           #  y/z
   jmp      .L6            #  goto merge
.L9:                       # Case 3
   movl     $1, %eax       #  w = 1
.L6:                       # merge:
   addq     %rcx, %rax     #  w += z
   ret
.L7:                       # Case 5,6
  movl   $1, %eax          #  w = 1
  subq   %rdx, %rax        #  w -= z
  ret
.L8:                       # Default:
  movl   $2, %eax          #  2
  ret
```

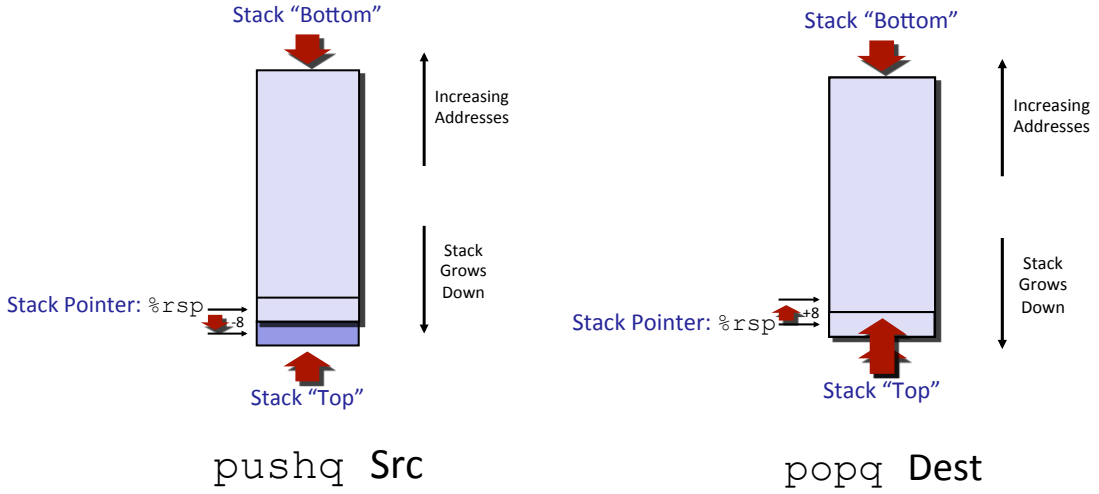
Procedure Call Registers

Return	%rax	%eax	%r8	%r8d	Arg 5
	%rbx	%ebx	%r9	%r9d	Arg 6
Arg 4	%rcx	%ecx	%r10	%r10d	
Arg 3	%rdx	%edx	%r11	%r11d	
Arg 2	%rsi	%esi	%r12	%r12d	
Arg 1	%rdi	%edi	%r13	%r13d	
Stack ptr	%rsp	%esp	%r14	%r14d	
	%rbp	%ebp	%r15	%r15d	

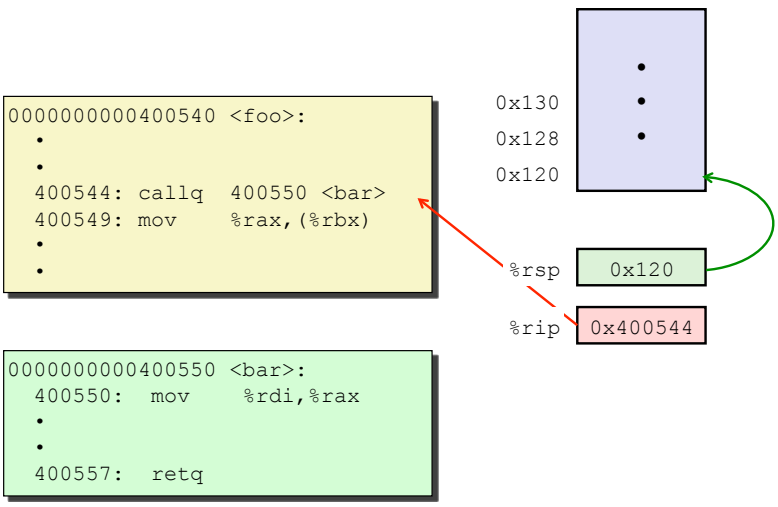
Call Stack



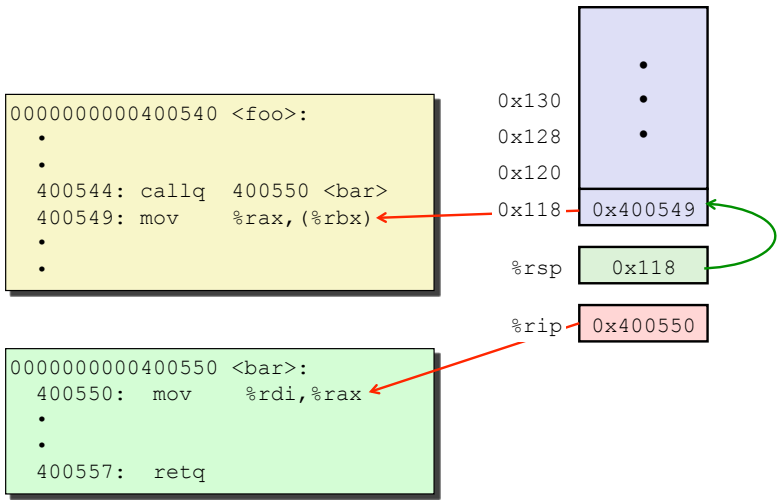
Stack Operations



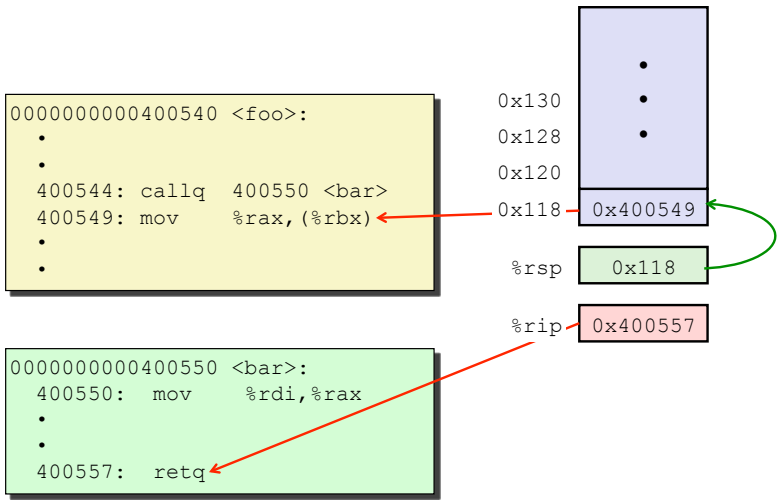
Control Flow Example (1)



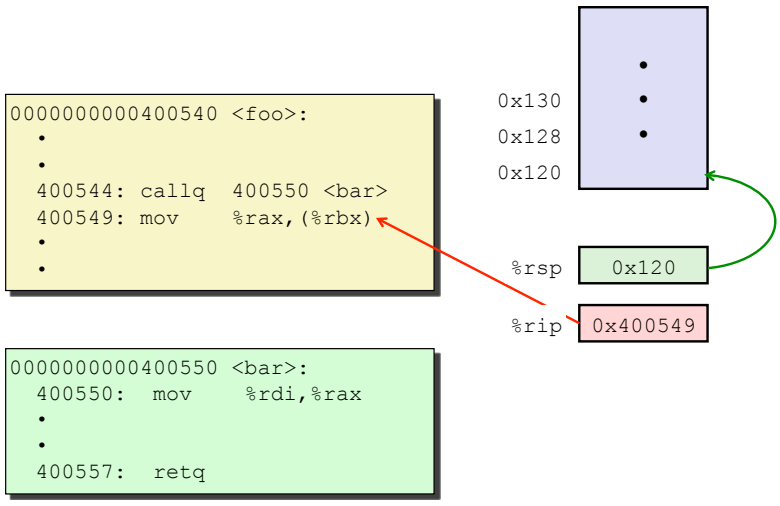
Control Flow Example (2)



Control Flow Example (3)



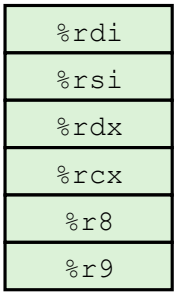
Control Flow Example (4)



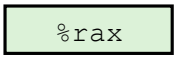
Passing Data

Registers

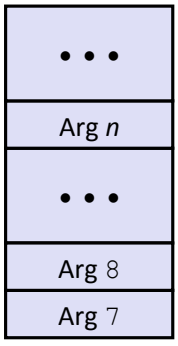
- **First 6 arguments**



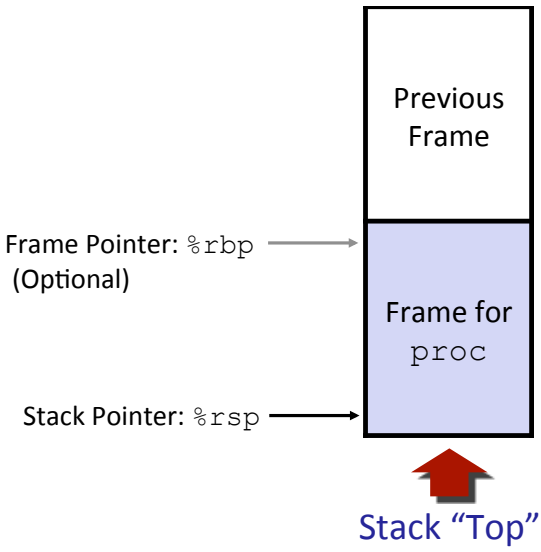
- **Return value**



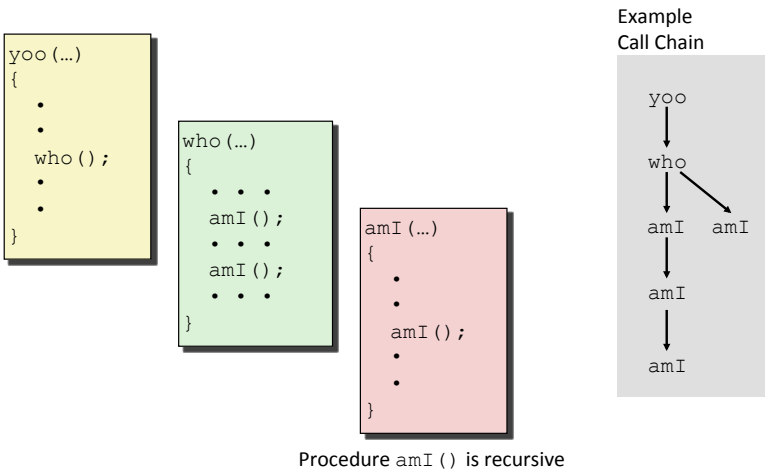
Stack



Stack Frames

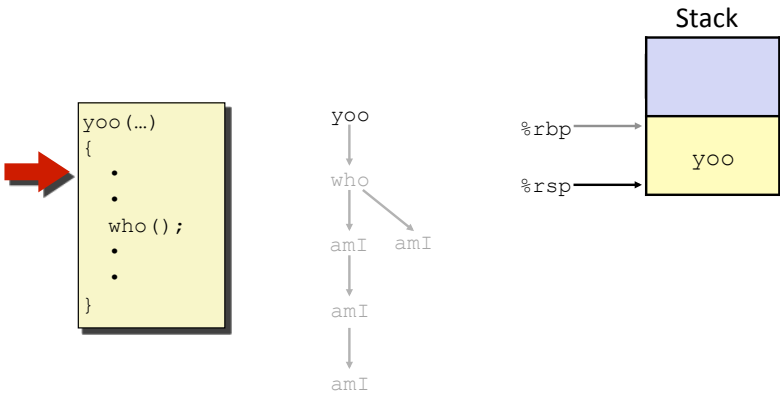


Call Chain Example

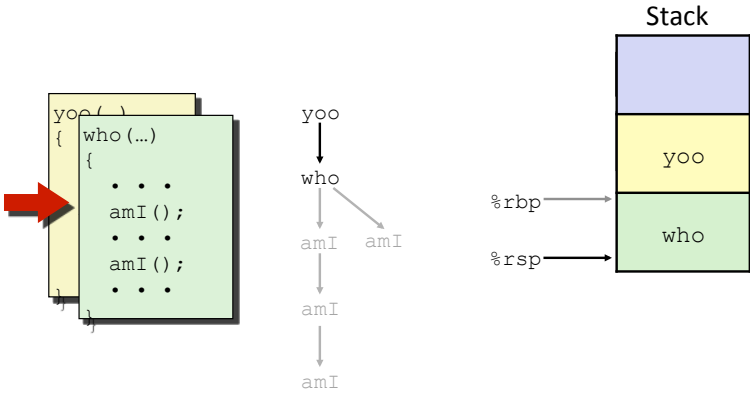


Procedure `amI()` is recursive

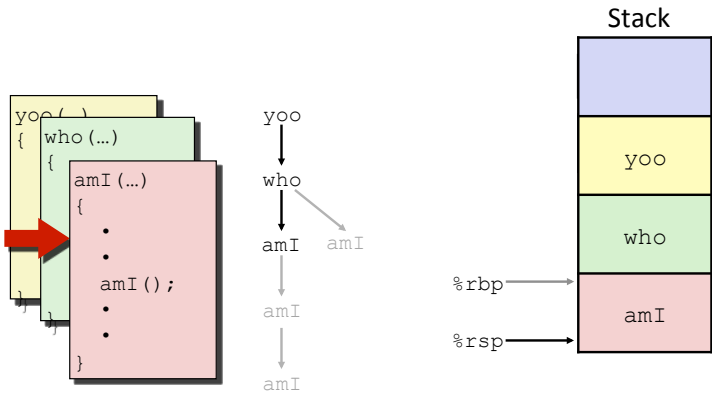
Stack Frame Allocation (1)



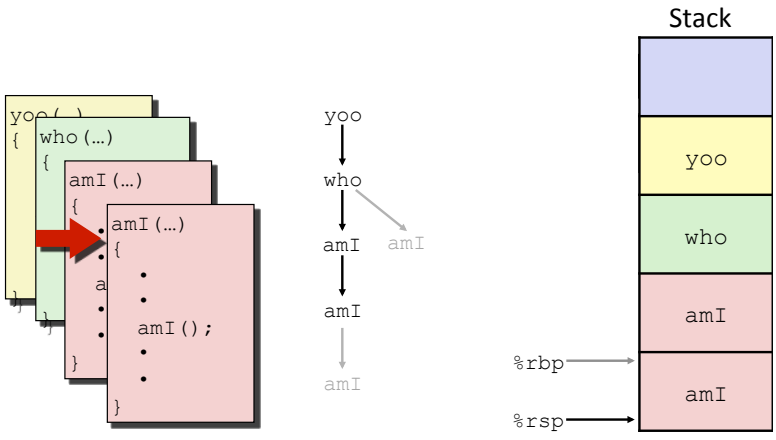
Stack Frame Allocation (2)



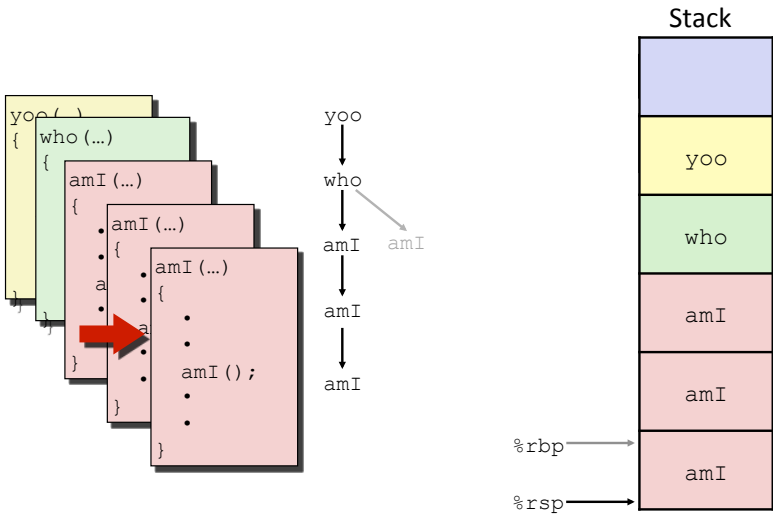
Stack Frame Allocation (3)



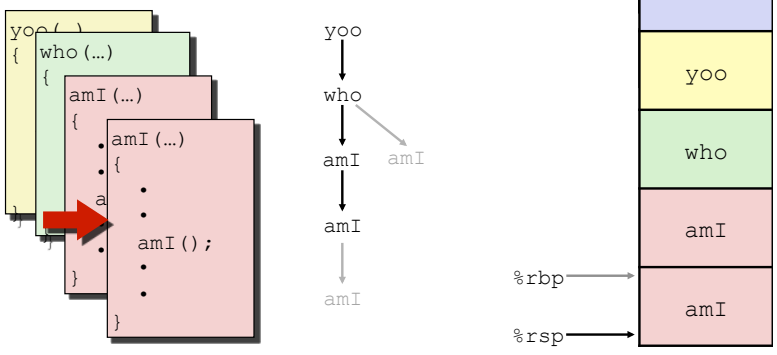
Stack Frame Allocation (4)



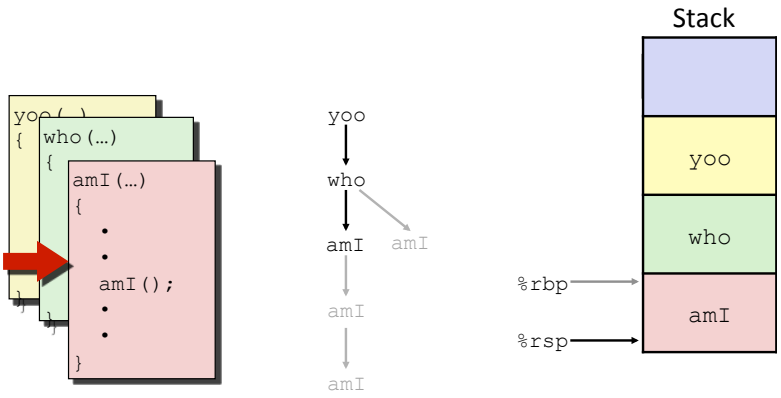
Stack Frame Allocation (5)



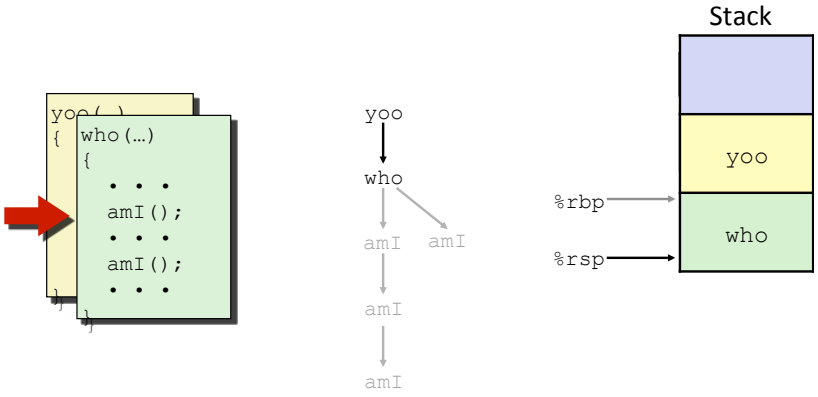
Stack Frame Allocation (6)



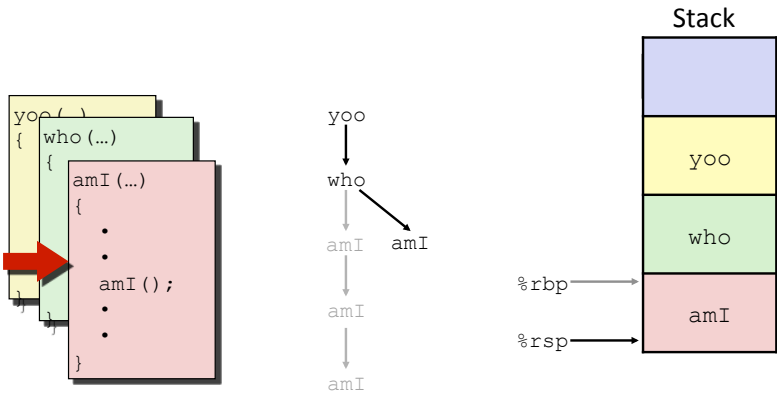
Stack Frame Allocation (7)



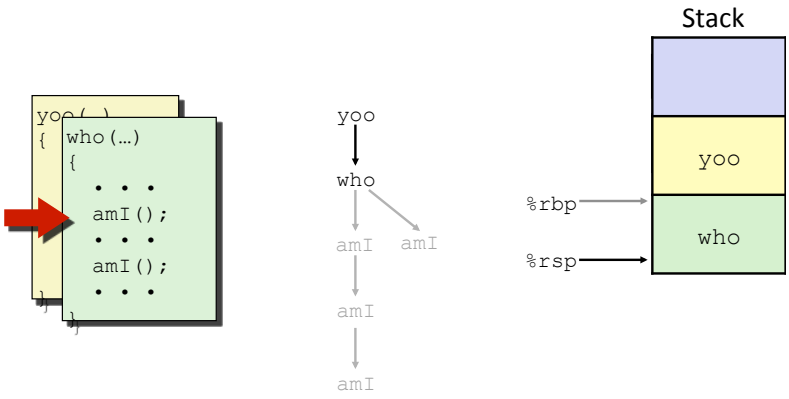
Stack Frame Allocation (8)



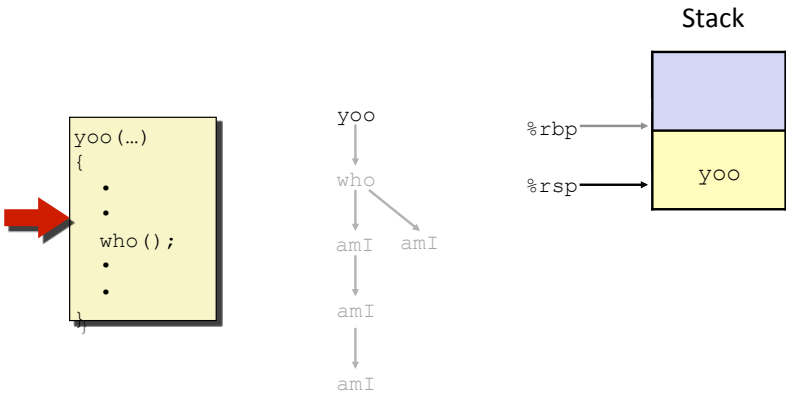
Stack Frame Allocation (9)



Stack Frame Allocation (10)



Stack Frame Allocation (11)



Stack Frame Components