



# Machine-Level Programming III: Procedures

These slides adapted from materials provided by the textbook

# Mechanisms in Procedures

## ■ Passing control

- To beginning of procedure code
- Back to return point

## ■ Passing data

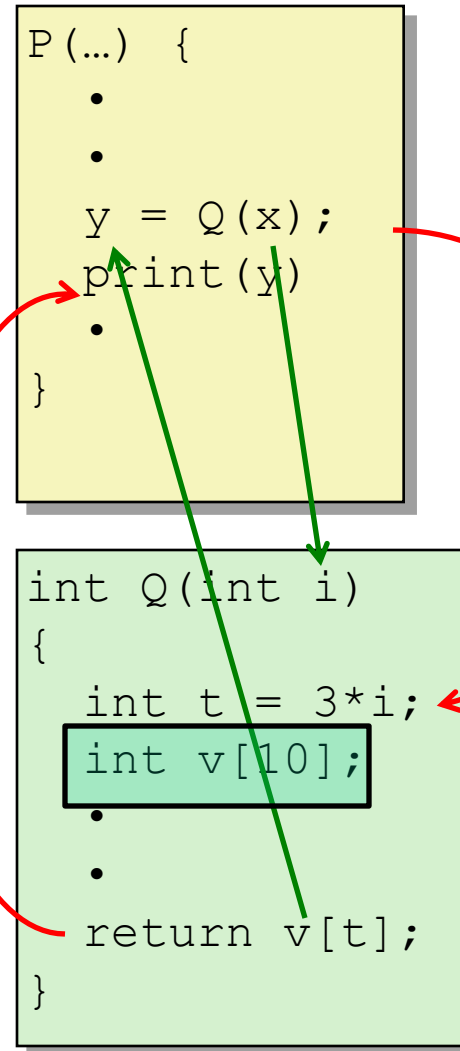
- Procedure arguments
- Return value

## ■ Memory management

- Allocate during procedure execution
- Deallocate upon return

## ■ Mechanisms all implemented with machine instructions

## ■ x86-64 implementation of a procedure uses only those mechanisms required

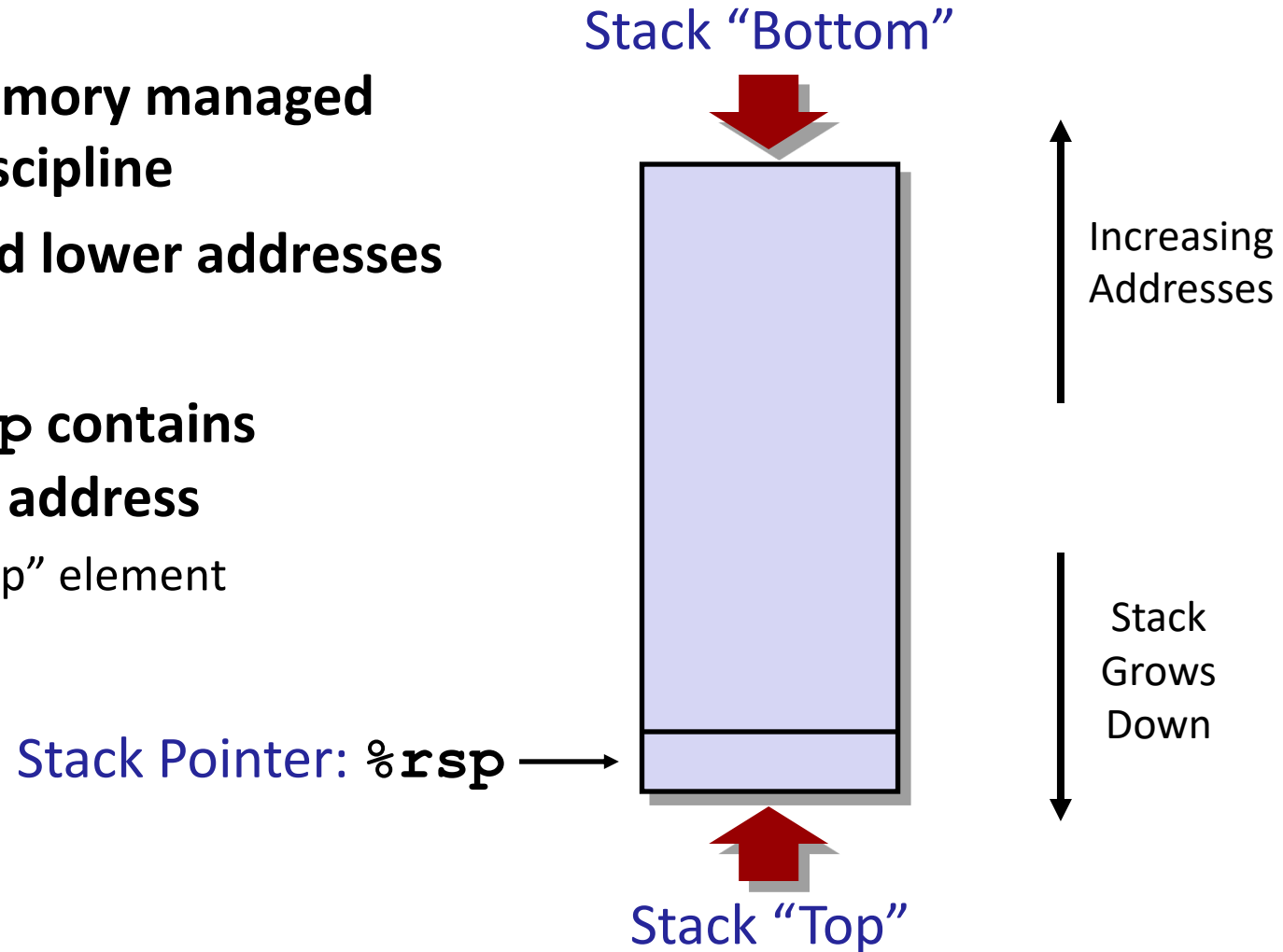


# Machine-Level Programming II: Procedures

- **Procedures**
  - **Stack Structure**
  - **Calling Conventions**
    - Passing control
    - Passing data
    - Managing local data

# x86-64 Stack

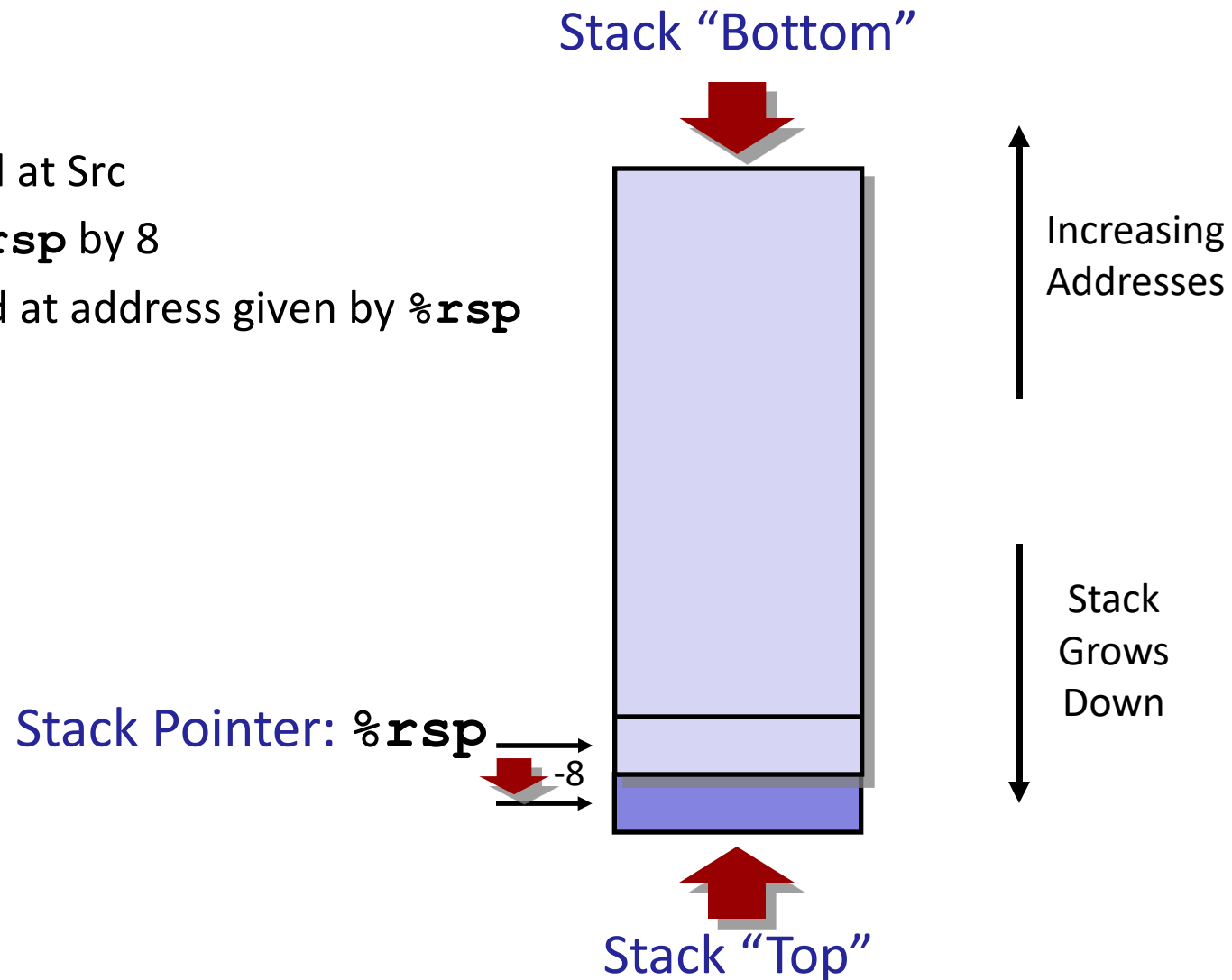
- Region of memory managed with stack discipline
- Grows toward lower addresses
- Register `%rsp` contains lowest stack address
  - address of “top” element



# x86-64 Stack: Push

## ■ `pushq Src`

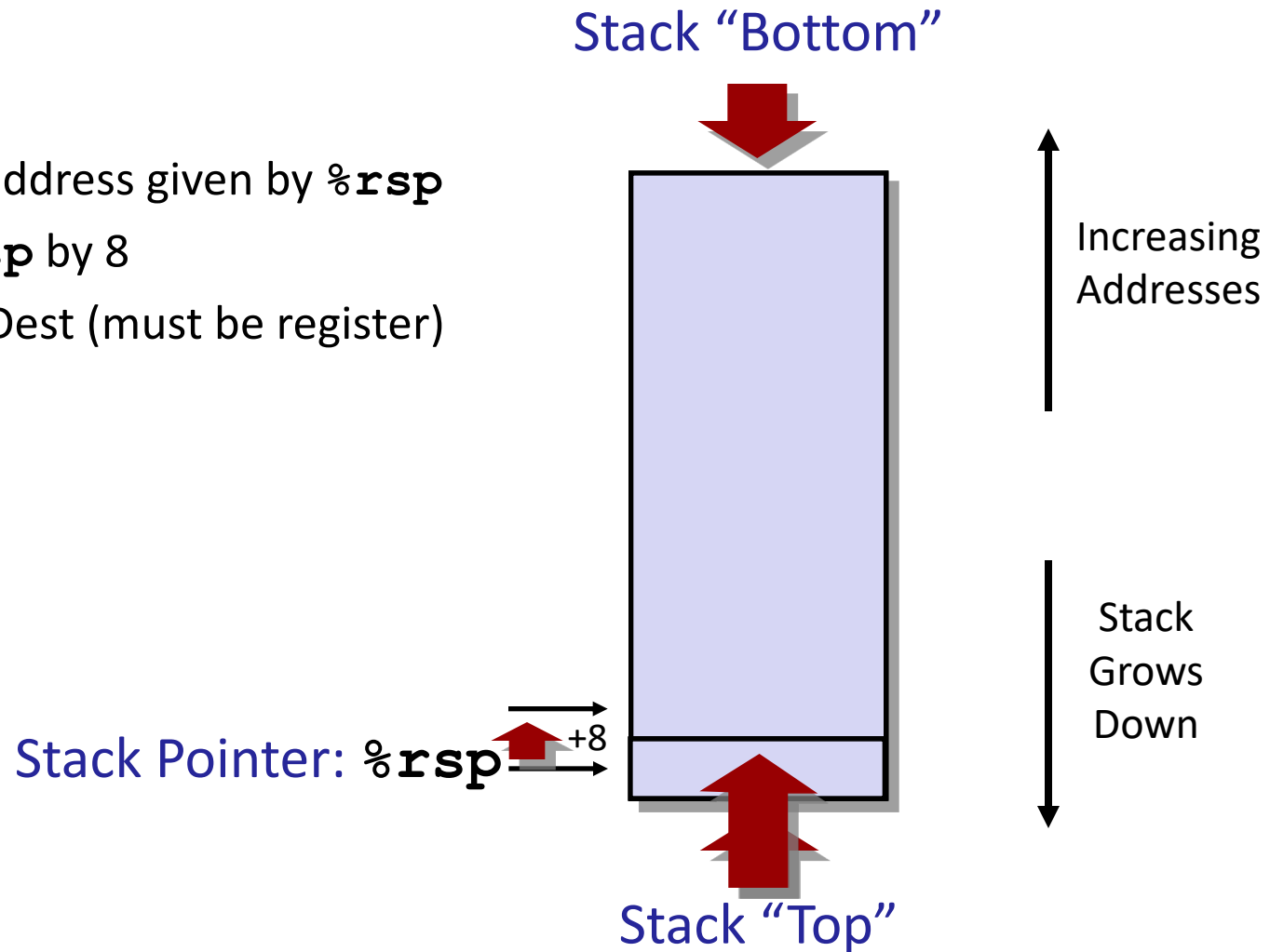
- Fetch operand at Src
- Decrement `%rsp` by 8
- Write operand at address given by `%rsp`



# x86-64 Stack: Pop

## ■ `popq Dest`

- Read value at address given by `%rsp`
- Increment `%rsp` by 8
- Store value at `Dest` (must be register)



# Machine-Level Programming II: Procedures

## ■ Procedures

- Stack Structure
- Calling Conventions
  - Passing control
  - Passing data
  - Managing local data

# Code Examples

```
void multstore
(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
```

```
00000000000400540 <multstore>:
  400540: push    %rbx                # Save %rbx
  400541: mov     %rdx,%rbx           # Save dest
  400544: callq   400550 <mult2>      # mult2(x,y)
  400549: mov     %rax, (%rbx)         # Save at dest
  40054c: pop     %rbx                # Restore %rbx
  40054d: retq                                # Return
```

```
long mult2
(long a, long b)
{
    long s = a * b;
    return s;
}
```

```
00000000000400550 <mult2>:
  400550: mov     %rdi,%rax           # a
  400553: imul    %rsi,%rax           # a * b
  400557: retq                                # Return
```



# Procedure Control Flow

- Use stack to support procedure call and return
- **Procedure call:** `call label`
  - Push return address on stack
  - Jump to label
- **Return address:**
  - Address of the next instruction right after call
  - Example from disassembly
- **Procedure return:** `ret`
  - Pop address from stack
  - Jump to address

# Control Flow Example #1

```
00000000000400540 <multstore>:  
.  
.  
400544: callq 400550 <mult2>  
400549: mov    %rax, (%rbx)  
.  
.
```

```
00000000000400550 <mult2>:  
400550: mov    %rdi,%rax  
.  
.  
400557: retq
```

0x130

0x128

0x120

%rsp

%rip

0x120

0x400544

# Control Flow Example #2

```
00000000000400540 <multstore>:  
.  
.  
400544: callq    400550 <mult2>  
400549: mov     %rax, (%rbx) ←  
.  
.
```

```
00000000000400550 <mult2>:  
400550: mov     %rdi,%rax ←  
.  
.  
400557: retq
```

0x130

0x128

0x120

0x118

%rsp

%rip

0x400549

0x118

0x400550

# Control Flow Example #3

```
00000000000400540 <multstore>:  
.  
.  
400544: callq 400550 <mult2>  
400549: mov    %rax, (%rbx) ←  
.  
.
```

```
00000000000400550 <mult2>:  
400550: mov    %rdi,%rax  
.  
.  
400557: retq ←
```

0x130

0x128

0x120

0x118

%rsp

%rip

0x400549

0x118

0x400557

# Control Flow Example #4

```
00000000000400540 <multstore>:  
.  
.  
400544: callq    400550 <mult2>  
400549: mov     %rax, (%rbx)  
.  
.
```

```
00000000000400550 <mult2>:  
400550: mov     %rdi,%rax  
.  
.  
400557: retq
```

0x130

0x128

0x120

%rsp

0x120

%rip

0x400549

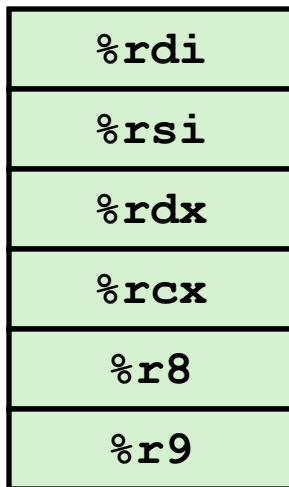
# Machine-Level Programming II: Procedures

- **Procedures**
  - Stack Structure
  - Calling Conventions
    - Passing control
    - **Passing data**
    - Managing local data

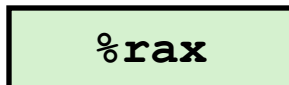
# Procedure Data Flow

## Registers

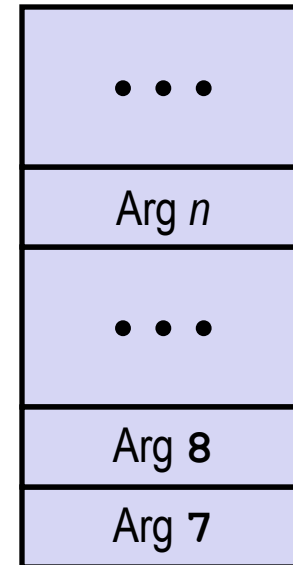
### ■ First 6 arguments



### ■ Return value



## Stack



### ■ Only allocate stack space when needed

# Data Flow Examples

```
void multstore
(long x, long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
```

```
0000000000400540 <multstore>:
    # x in %rdi, y in %rsi, dest in %rdx
    . . .
400541: mov     %rdx,%rbx        # Save dest
400544: callq   400550 <mult2>    # mult2(x,y)
    # t in %rax
400549: mov     %rax, (%rbx)      # Save at dest
    . . .
```

```
long mult2
(long a, long b)
{
    long s = a * b;
    return s;
}
```

```
0000000000400550 <mult2>:
    # a in %rdi, b in %rsi
400550: mov     %rdi,%rax        # a
400553: imul    %rsi,%rax        # a * b
    # s in %rax
400557: retq                      # Return
```