



# Machine-Level Programming I: Basics

These slides adapted from materials provided by the textbook authors.

# Machine Programming I: Basics

- History of Intel processors and architectures
- C, assembly, machine code
- **Assembly Basics: Registers, operands, move**
- Arithmetic & logical operations

# x86-64 Integer Registers

<b>%rax</b>	<b>%eax</b>
<b>%rbx</b>	<b>%ebx</b>
<b>%rcx</b>	<b>%ecx</b>
<b>%rdx</b>	<b>%edx</b>
<b>%rsi</b>	<b>%esi</b>
<b>%rdi</b>	<b>%edi</b>
<b>%rsp</b>	<b>%esp</b>
<b>%rbp</b>	<b>%ebp</b>

<b>%r8</b>	<b>%r8d</b>
<b>%r9</b>	<b>%r9d</b>
<b>%r10</b>	<b>%r10d</b>
<b>%r11</b>	<b>%r11d</b>
<b>%r12</b>	<b>%r12d</b>
<b>%r13</b>	<b>%r13d</b>
<b>%r14</b>	<b>%r14d</b>
<b>%r15</b>	<b>%r15d</b>

- Can reference low-order 4 bytes (also low-order 1 & 2 bytes)

# Some History: IA32 Registers

general purpose	<b>%eax</b>	<b>%ax</b>	<b>%ah</b>	<b>%al</b>	Origin (mostly obsolete)  <i>accumulate</i>
	<b>%ecx</b>	<b>%cx</b>	<b>%ch</b>	<b>%cl</b>	<i>counter</i>
	<b>%edx</b>	<b>%dx</b>	<b>%dh</b>	<b>%dl</b>	<i>data</i>
	<b>%ebx</b>	<b>%bx</b>	<b>%bh</b>	<b>%bl</b>	<i>base</i>
	<b>%esi</b>	<b>%si</b>			<i>source index</i>
	<b>%edi</b>	<b>%di</b>			<i>destination index</i>
	<b>%esp</b>	<b>%sp</b>			<i>stack pointer</i>
	<b>%ebp</b>	<b>%bp</b>			<i>base pointer</i>

16-bit virtual registers  
(backwards compatibility)

# Moving Data

## ■ Moving Data

`movq Source, Dest:`

## ■ Operand Types

- **Immediate:** Constant integer data
  - Example: `$0x400`, `$-533`
  - Like C constant, but prefixed with ``$'`
  - Encoded with 1, 2, or 4 bytes
- **Register:** One of 16 integer registers
  - Example: `%rax`, `%r13`
  - But `%rsp` reserved for special use
  - Others have special uses for particular instructions
- **Memory:** 8 consecutive bytes of memory at address given by register
  - Simplest example: `(%rax)`
  - Various other “address modes”

`%rax`

`%rcx`

`%rdx`

`%rbx`

`%rsi`

`%rdi`

`%rsp`

`%rbp`

`%rN`

# movq Operand Combinations

	Source	Dest	Src, Dest	C Analog
movq	Imm	Reg	movq \$0x4, %rax	temp = 0x4;
		Mem	movq \$-147, (%rax)	*p = -147;
	Reg	Reg	movq %rax, %rdx	temp2 = temp1;
		Mem	movq %rax, (%rdx)	*p = temp;
	Mem	Reg	movq (%rax), %rdx	temp = *p;

*Cannot do memory-memory transfer with a single instruction*

# Simple Memory Addressing Modes

## ■ Normal (R) Mem[Reg[R]]

- Register R specifies memory address
- Aha! Pointer dereferencing in C

```
movq (%rcx) , %rax
```

## ■ Displacement D(R) Mem[Reg[R]+D]

- Register R specifies start of memory region
- Constant displacement D specifies offset

```
movq 8(%rbp) , %rdx
```

# Example of Simple Addressing Modes

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

```
swap:
    movq    (%rdi), %rax
    movq    (%rsi), %rdx
    movq    %rdx, (%rdi)
    movq    %rax, (%rsi)
    ret
```



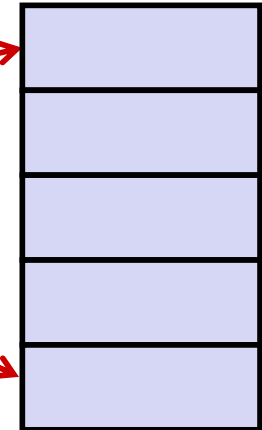
# Understanding Swap()

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

## Registers

%rdi	
%rsi	
%rax	
%rdx	

## Memory



Register	Value
%rdi	xp
%rsi	yp
%rax	t0
%rdx	t1

swap:

```
movq    (%rdi), %rax    # t0 = *xp
movq    (%rsi), %rdx    # t1 = *yp
movq    %rdx, (%rdi)    # *xp = t1
movq    %rax, (%rsi)    # *yp = t0
ret
```

# Understanding Swap()

## Registers

%rdi	0x120
%rsi	0x100
%rax	
%rdx	

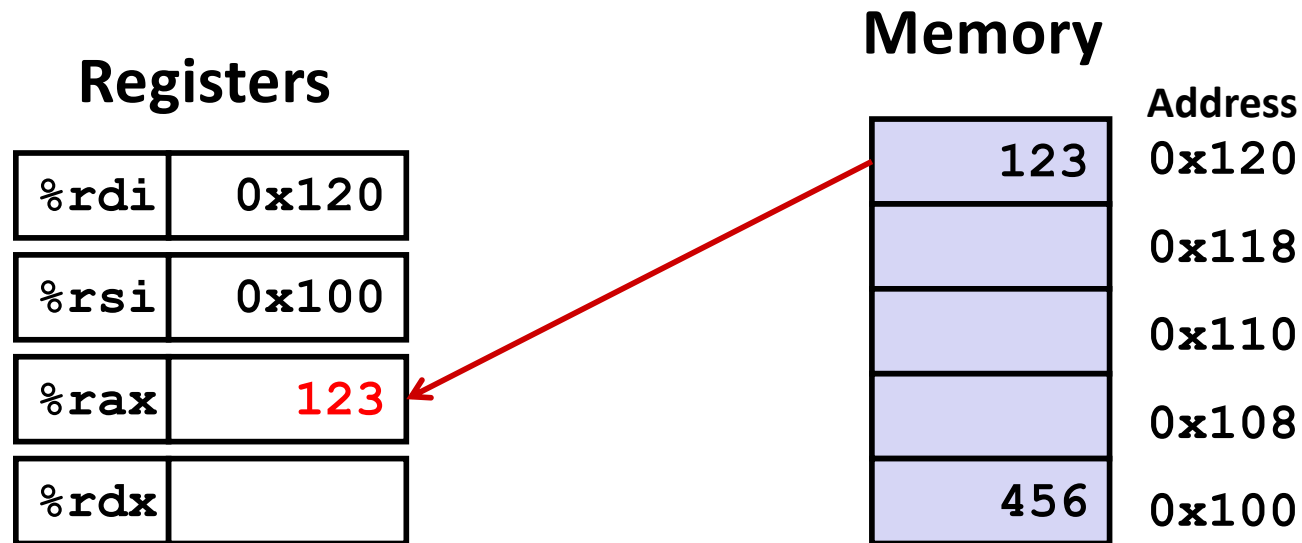
## Memory

	Address
123	0x120
	0x118
	0x110
	0x108
456	0x100

swap:

```
    movq    (%rdi), %rax    # t0 = *xp
    movq    (%rsi), %rdx    # t1 = *yp
    movq    %rdx, (%rdi)    # *xp = t1
    movq    %rax, (%rsi)    # *yp = t0
    ret
```

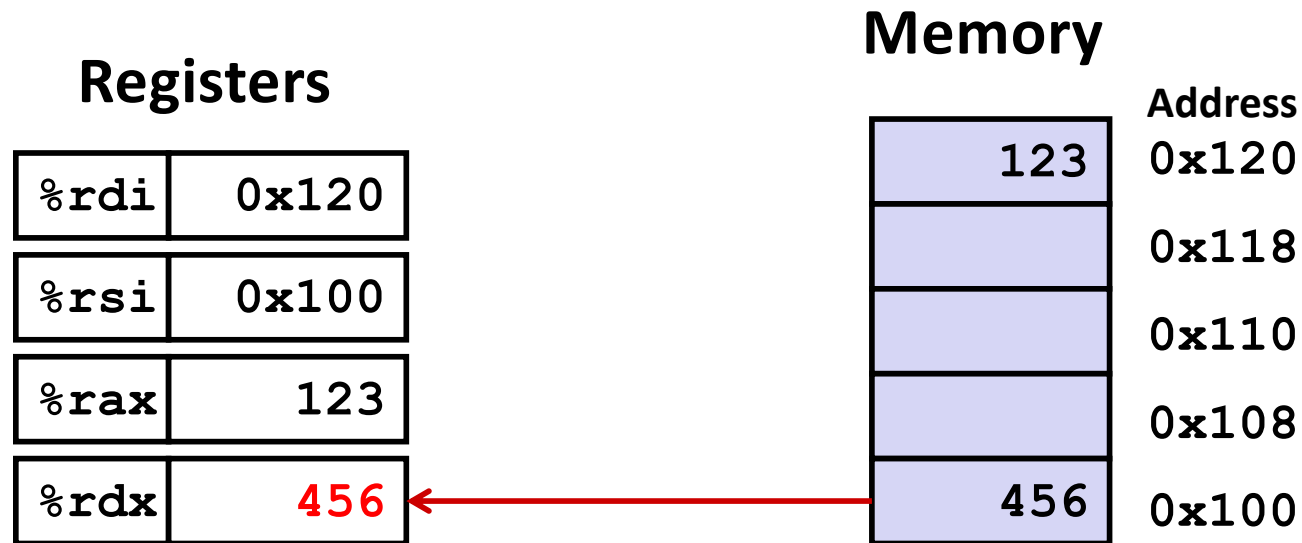
# Understanding Swap()



swap:

```
movq    (%rdi), %rax    # t0 = *xp
movq    (%rsi), %rdx    # t1 = *yp
movq    %rdx, (%rdi)    # *xp = t1
movq    %rax, (%rsi)    # *yp = t0
ret
```

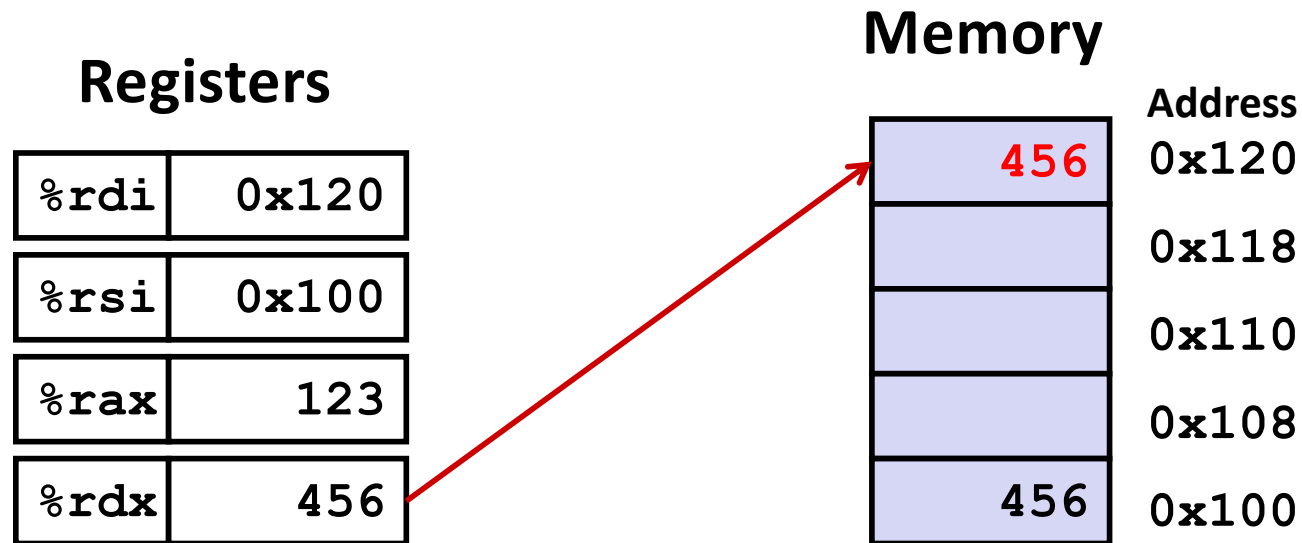
# Understanding Swap()



swap:

```
movq    (%rdi), %rax    # t0 = *xp
movq    (%rsi), %rdx    # t1 = *yp
movq    %rdx, (%rdi)    # *xp = t1
movq    %rax, (%rsi)    # *yp = t0
ret
```

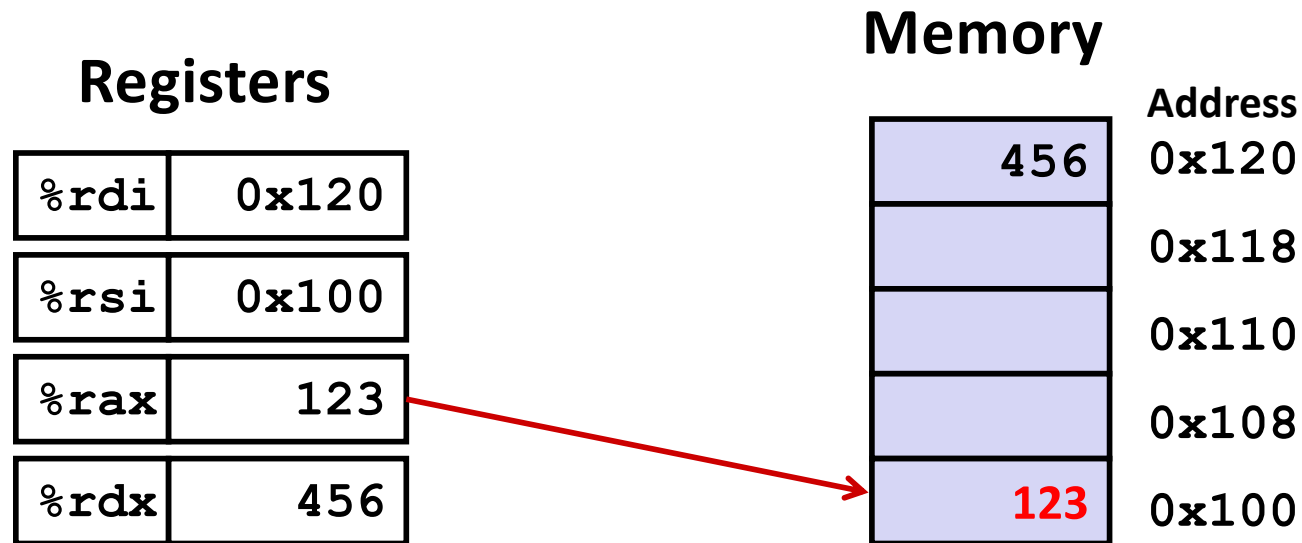
# Understanding Swap()



swap:

```
movq    (%rdi), %rax    # t0 = *xp
movq    (%rsi), %rdx    # t1 = *yp
movq    %rdx, (%rdi)    # *xp = t1
movq    %rax, (%rsi)    # *yp = t0
ret
```

# Understanding Swap()



swap:

```
movq    (%rdi), %rax    # t0 = *xp
movq    (%rsi), %rdx    # t1 = *yp
movq    %rdx, (%rdi)    # *xp = t1
movq    %rax, (%rsi)    # *yp = t0
ret
```

# Simple Memory Addressing Modes

## ■ Normal (R) Mem[Reg[R]]

- Register R specifies memory address
- Aha! Pointer dereferencing in C

```
movq (%rcx) , %rax
```

## ■ Displacement D(R) Mem[Reg[R]+D]

- Register R specifies start of memory region
- Constant displacement D specifies offset

```
movq 8(%rbp) , %rdx
```

# Complete Memory Addressing Modes

## ■ Most General Form

**$D(Rb, Ri, S)$**

**$Mem[Reg[Rb] + S * Reg[Ri] + D]$**

- D: Constant “displacement” 1, 2, or 4 bytes
- Rb: Base register: Any of 16 integer registers
- Ri: Index register: Any, except for `%rsp`
- S: Scale: 1, 2, 4, or 8 (*why these numbers?*)

## ■ Special Cases

**$(Rb, Ri)$**

**$Mem[Reg[Rb] + Reg[Ri]]$**

**$D(Rb, Ri)$**

**$Mem[Reg[Rb] + Reg[Ri] + D]$**

**$(Rb, Ri, S)$**

**$Mem[Reg[Rb] + S * Reg[Ri]]$**



# Address Computation Examples

<code>%rdx</code>	<code>0xf000</code>
<code>%rcx</code>	<code>0x0100</code>

Expression	Address Computation	Address
<code>0x8(%rdx)</code>	<code>0xf000 + 0x8</code>	<code>0xf008</code>
<code>(%rdx,%rcx)</code>	<code>0xf000 + 0x100</code>	<code>0xf100</code>
<code>(%rdx,%rcx,4)</code>	<code>0xf000 + 4*0x100</code>	<code>0xf400</code>
<code>0x80(,%rdx,2)</code>	<code>2*0xf000 + 0x80</code>	<code>0x1e080</code>