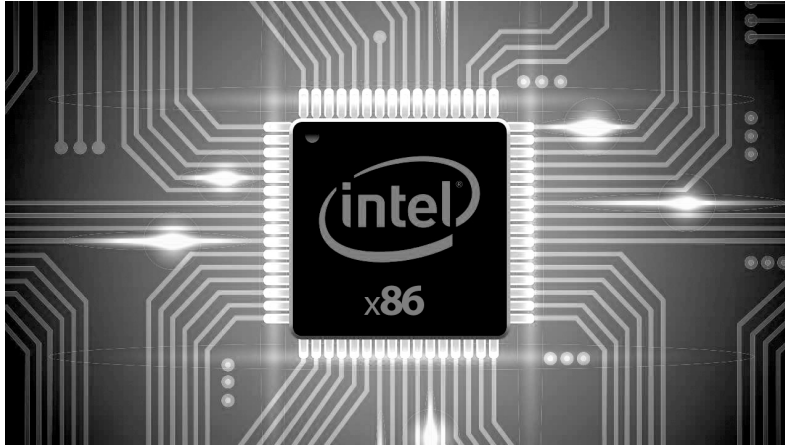
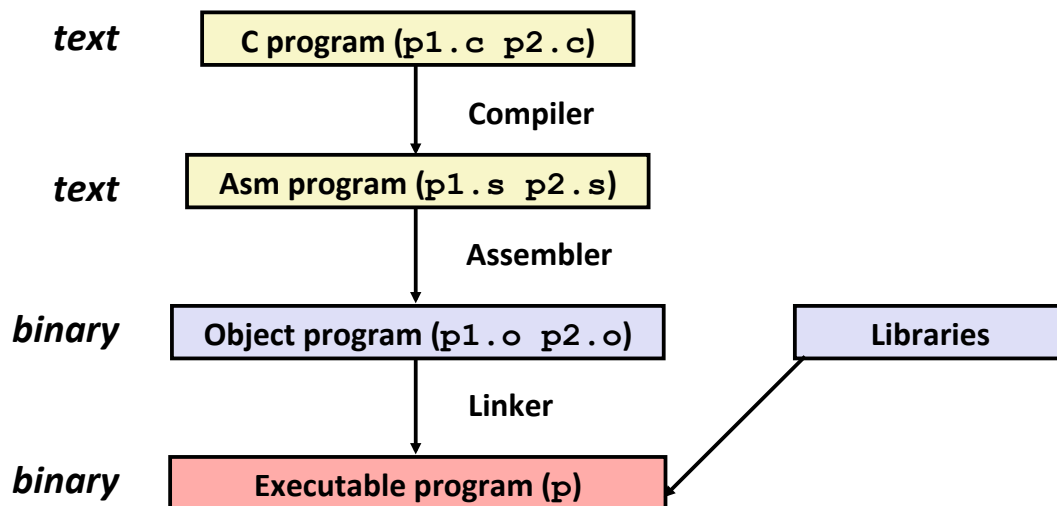


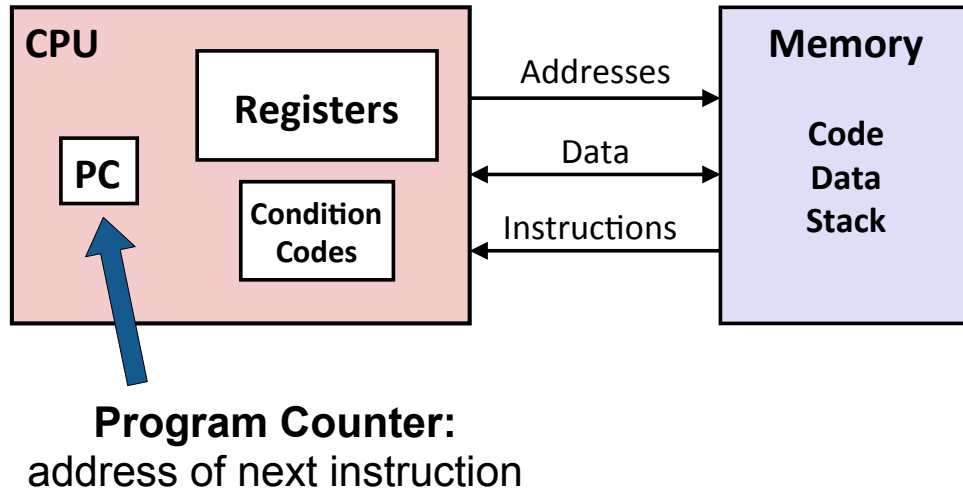
Machine Code



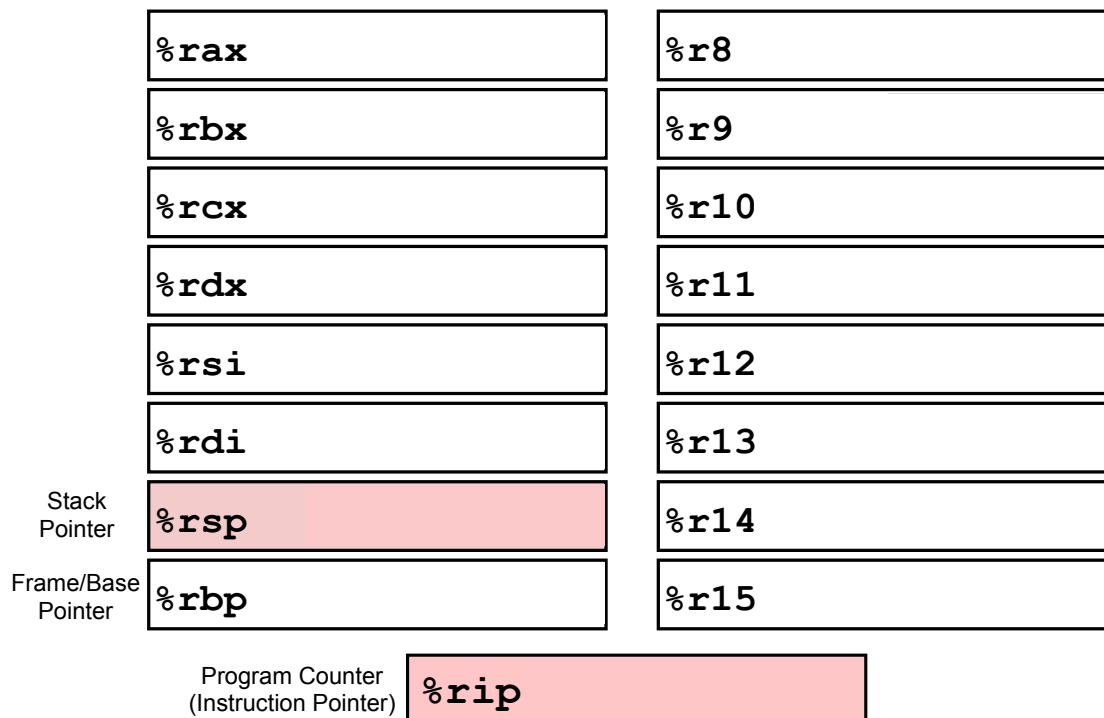
From C to Executable Code



Assembly View of the Machine



x86-64 Integer Registers



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;

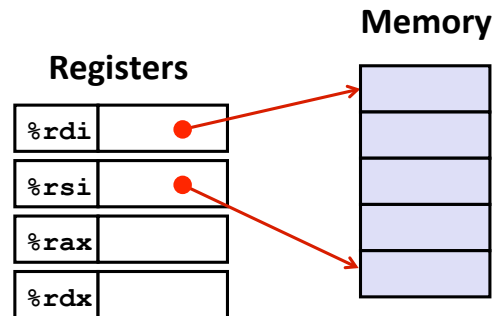
Assembly Translation Example

```
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;
}
```



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
```

x86-64 Virtual Registers

64-Bit Register	Lowest 32 Bits	Lowest 16 Bits	Lowest 8 Bits
%rax	%eax	%ax	%al
%rbx	%ebx	%bx	%bl
%rcx	%ecx	%cx	%cl
%rdx	%edx	%dx	%dl
%rsi	%esi	%si	%sil
%rdi	%edi	%di	%dil
%rbp	%ebp	%bp	%bpl
%rsp	%esp	%sp	%spl
%r8	%r8d	%r8w	%r8b
%r9	%r9d	%r9w	%r9b
%r10	%r10d	%r10w	%r10b
%r11	%r11d	%r11w	%r11b
%r12	%r12d	%r12w	%r12b
%r13	%r13d	%r13w	%r13b
%r14	%r14d	%r14w	%r14b
%r15	%r15d	%r15w	%r15b

Data Size Suffixes

Suffix	Size	Description
b	8 bits	byte
w	16 bits	word (historical)
l	32 bits	long word
q	64 bits	quad word

Setting 1 or 2-byte virtual register (e.g. `%ax`, `%al`): top 6/7 bytes **unchanged**

Setting 4-byte virtual register (e.g. `%eax`): top 4 bytes **zeroed**

Data Movement Examples

“Copy K bytes from [val N/addr N/reg N] to [addr M/reg M]”

1. `movq %rax, %rbx`
2. `movw %ax, %bx`
3. `movb $5, %cl`
4. `movq $-12, (%rcx)`
5. `movl $0xFF, %eax`
6. `movl 5, %eax`
7. `movw %ax, 30`
8. `movq (%rax), %rbx`
9. `movl (%rax), %ebx`
10. `movb %al, (%rbx)`

General Memory Addressing

General Form:

$D(Rb, Ri, S) \quad Mem[D + Reg[Rb] + S * Reg[Ri]]$

D Constant "displacement"
Rb Base register
Ri Index register
s Scale constant: 1, 2, 4, or 8

$5(\%rax, \%rbx, 8) \quad Mem[5 + \%rax + 8 * \%rbx]$

Special Cases:

(Rb)	(%rax)	Mem[Reg[Rb]]
D(Rb)	5(%rax)	Mem[D + Reg[Rb]]
(Rb, Ri)	(%rax, %rbx)	Mem[Reg[Rb] + Reg[Ri]]
D(Rb, Ri)	5(%rax, %rbx)	Mem[D + Reg[Rb] + Reg[Ri]]
(Rb, Ri, S)	(%rax, %rbx, 8)	Mem[Reg[Rb] + S * Reg[Ri]]
(, Ri, S)	(, %rbx, 8)	Mem[S * Reg[Ri]]
D(, Ri, S)	5(, %rbx, 8)	Mem[D + S * Reg[Ri]]

Arithmetic Instructions

addq	Src, Dest	Dest = Dest + Src	
subq	Src, Dest	Dest = Dest - Src	
imulq	Src, Dest	Dest = Dest * Src	
sarq	Src, Dest	Dest = Dest >> Src	<i>Arithmetic RShift</i>
shrq	Src, Dest	Dest = Dest >> Src	<i>Logical RShift</i>
salq	Src, Dest	Dest = Dest << Src	<i>Also called shlq</i>
xorq	Src, Dest	Dest = Dest ^ Src	
andq	Src, Dest	Dest = Dest & Src	
orq	Src, Dest	Dest = Dest Src	
incq	Dest	Dest = Dest + 1	
decq	Dest	Dest = Dest - 1	
negq	Dest	Dest = -Dest	
notq	Dest	Dest = ~Dest	

leaq Src, Dest Dest = Src (as expr) *No memory access!*

Arithmetic Example

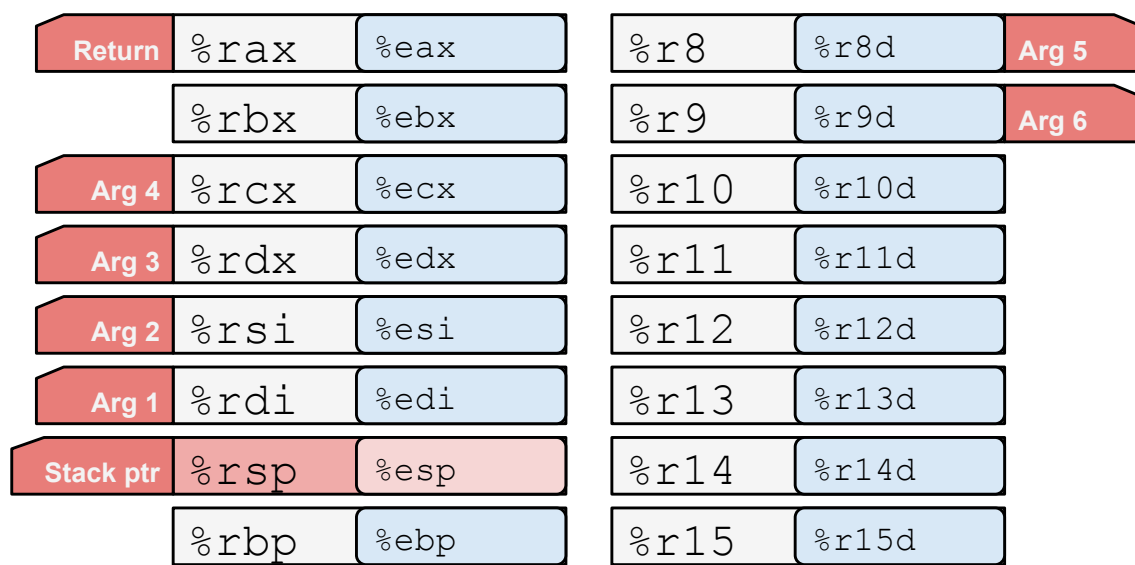
(x,y,z) -> (%rdi,%rsi,%rdx)

```
long arith
(long x, long y, long z)
{
    long t1 = x+y;
    long t2 = z+t1;
    long t3 = x+4;
    long t4 = y * 48;
    long t5 = t3 + t4;
    long rval = t2 * t5;
    return rval;
}
```

arith:

```
1. leaq    (%rdi,%rsi), %rax
2. addq    %rdx, %rax
3. leaq    (%rsi,%rsi,2), %rdx
4. salq    $4, %rdx
5. leaq    4(%rdi,%rdx), %rcx
6. imulq    %rcx, %rax
ret
```

Procedure Call Registers



Procedure Call Example

```
long x = add(3, 5);  
doSomething(x);
```

```
long add(long x, long y) {  
    return x + y;  
}
```

```
movq $3, %rdi  
movq $5, %rsi  
callq add  
movq %rax, %rdi  
callq doSomething
```

```
add:  
    movq %rdi, %rax  
    addq %rsi, %rax  
    retq
```