X86-64 Registers

64-bit	32-bit	16-bit	8-bit	Use / Function's Duty
RAX	EAX	AX	AL	Return Value / Not preserved
RBX	EBX	BX	BL	Temporary / Preserve
RCX	ECX	CX	CL	4th Argument / Not preserved
RDX	EDX	DX	DL	3rd Argument / Not preserved
RSI	ESI	SI	SIL	2nd Argument / Not Preserved
RDI	EDI	DI	DIL	1st Argument / Not Preserved
RBP	EBP	BP	BPL	Current Stack-Frame Pointer / Preserve
RSP	ESP	SP	SPL	Stack Pointer / Preserve
R8	R8D	R8W	R8B	5th Argument / Not preserved
R9	R9D	R9W	R9B	6th Argument / Not preserved
R10	R10D	R10W	R10B	Temporary / Not preserved
R11	R11D	R11W	R11B	Temporary / Not preserved
R12	R12D	R12W	R12B	— / Preserve
R13	R13D	R13W	R13B	— / Preserve
R14	R14D	R14W	R14B	— / Preserve
R15	R15D	R15W	R15B	— / Preserve
RIP				Instruction Pointer

Code:

```
long add_two_longs(long a, long b) {
    long result = a + b;
    return result;
}

void add_two_longPtrs(long* a, long* b, long* r) {
    *r = *a + *b;
}
```

Generated Code (Optimized):

```
add_two_longs:
        leaq    (%rdi,%rsi), %rax
        ret

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

Generated Code (Not Optimized):

```
add_two_longs:
        pushq   %rbp
        movq    %rsp, %rbp
        movq    %rdi, -24(%rbp)
        movq    %rsi, -32(%rbp)
        movq    -24(%rbp), %rdx
        movq    -32(%rbp), %rax
        addq    %rdx, %rax
        movq    %rax, -8(%rbp)
        movq    -8(%rbp), %rax
        popq    %rbp
        ret

add_two_longPtrs:
        pushq   %rbp
        movq    %rsp, %rbp
        movq    %rdi, -8(%rbp)
        movq    %rsi, -16(%rbp)
        movq    %rdx, -24(%rbp)
        movq    -8(%rbp), %rax
        movq    (%rax), %rdx
        movq    -16(%rbp), %rax
        movq    (%rax), %rax
        addq    %rax, %rdx
        movq    -24(%rbp), %rax
        movq    %rdx, (%rax)
        nop
        popq    %rbp
```

Code:

```
long fact(long n) {
    if (n < 1)
        return 1;
    long result = fact(n-1);
    return result * n;
}
```

Generated Code (Optimized):

```
fact:
        movl    $1, %eax
        testq   %rdi, %rdi
        jle     .L5
        pushq   %rbp
        movq    %rsp, %rbp
        pushq   %rbx
        subq    $8, %rsp
        movq    %rdi, %rbx
        leaq    -1(%rdi), %rdi
        call    fact
        imulq   %rbx, %rax
        movq    -8(%rbp), %rbx
        leave
        ret
.L5:
        ret
```

Generated Code (Not Optimized):

```
fact:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $32, %rsp
        movq    %rdi, -24(%rbp)
        cmpq    $0, -24(%rbp)
        jg      .L2
        movl    $1, %eax
        jmp     .L3
.L2:
        movq    -24(%rbp), %rax
        subq    $1, %rax
        movq    %rax, %rdi
        call    fact
        movq    %rax, -8(%rbp)
        movq    -8(%rbp), %rax
        imulq   -24(%rbp), %rax
.L3:
        leave
        ret
```

Code:

```
long do_add1(long a, long b) {
    long x = a;
    long y = b;
    long z = add_two_longs(a, b);
    return z;
}

long do_add2(long a, long b) {
    long x = a;
    long y = b;
    long z;
    add_two_longPtrs(&x, &y, &z);
    return z;
}
```

Generated Code:

```
do_add1:
        pushq   %rbp
        movq    %rsp, %rbp
        call    add_two_longs
        popq    %rbp
        ret

do_add2:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $32, %rsp
        movq    %rdi, -8(%rbp)
        movq    %rsi, -16(%rbp)
        leaq    -24(%rbp), %rdx
        leaq    -16(%rbp), %rsi
        leaq    -8(%rbp), %rdi
        call    add_two_longPtrs
        movq    -24(%rbp), %rax
        leave
        ret
```