a | b = ~(~a & ~b)

a ^ b = (a & ~b) | (~a & b)

x86-64 linux calling convention:

Integer parameters:

%rdi, %rsi, %rdx, %rcx, %r8 and %r9

Others are stored in stack

**CF** Carry Flag (for unsigned) **SF** Sign Flag (for signed)

**ZF** Zero Flag **OF** Overflow Flag (for signed)

Implicitly set (as side effect) of arithmetic operations (**but not set by leaq instruction)**

addq Src DestDest (t = a + b)

**CF** set if carry out from most significant bit (unsigned overflow)

**ZF** set if t == 0

**SF** set if t < 0 (as signed)

**OF** set if two’s complement (signed) overflow

**Rules for turning on the carry flag**

1. The carry flag is set if the addition of two numbers causes a carry out of the most significant bits added.

1111 + 0001 = 0000 (carry flag is turned on)

2. The carry (borrow) flag is also set if the subtraction of two numbers requires a borrow into the most significant (leftmost) bits subtracted

0000 - 0001 = 1111 (carry flag is turned on)

**Rules for turning on the overflow flag**

1. If the sum of two numbers with the sign bits off yields a result number with the sign bit on

0100 + 0100 = 1000 (overflow flag is turned on)

2. If the sum of two numbers with the sign bits on yields a result number with the sign bit off

1000 + 1000 = 0000 (overflow flag is turned on)

**Note that different from above (1111 + 0001 = 0000), the result is correct even though CF is set**

In unsigned arithmetic, use the carry flag

In signed arithmetic, use the overflow flag

**cmp Instruction**

cmp b, a

Computes *b - a* (just like sub). Sets condition codes based on result, but **does not change *b***

**test instruction**

test a, b

Computes 𝑏 ∧ 𝑎 just like and. Sets condition codes (only SF and ZF) based on result, but **does not change 𝒃**

Most common use: test x, x

to compare x to zero