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

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

b = 1 byte, w = 2 bytes, l = 4 bytes, q = 8 bytes

Shape, rectangle

Description automatically generatedShape, rectangle

Description automatically generated

**Table

Description automatically generated with medium confidenceNormalized value** (exp ≠ 000…0 and exp ≠ 111…1)

E = Exp – Bias. Bias =

**Denormalized Value** (exp = 000…0)

Exponent value: E = 1 – Bias (**equispaced**)

**Infinity**: exp = 111…1, frac = 000…0

**NaN**: exp = 111…1, frac ≠ 000…0

Chart

Description automatically generated**x86-64 linux calling convention:**

Integer parameters:

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

Others are stored in stack, pushed in reversed (right-to-left) order

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

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

Implicitly set by 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**

unsigned arithmetic -> CF | signed arithmetic -> OF

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

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

Table

Description automatically generated

**movzbl**: zero-extend, byte -> long. **movslq**: sign-extend, long -> quad. Etc.

A picture containing diagram

Description automatically generated

**Buffer overflow attacks**

Stack Smashing Attacks: overwrite normal return address. Code Injection Attacks: overwrite normal return address and jump to exploit code

**Measures**

Avoid overflow vulnerabilities: strcpy -> strncpy. Employ system-level protections: randomized stack offsets, nonexecutable code segments. Have compiler use stack canaries

**Return-Oriented Programming Attacks**

Work around stack randomization and marking stack nonexecutable. Does not overcome stack canaries

Diagram

Description automatically generatedGraphical user interface, text, application

Description automatically generated