Bits, Bytes, and Integers

In computers, everything consists of bits.

By encoding sets of bits in various ways, they made meanings:

- instructions
- and data(numbers, sets, strings, etc..)

Boolean Algebra

- and `A & B`
- or `A | B`
- not `~A`
- xor `A ^ B`

in C

- Shift (`<<` , `>>`)
  - Left Shift(`<<`)
    Zero fill on right
  - Right Shift(`>>`)
    Logical Shift: zero fill with 0's on left
    Arithmetic shift Relicate most significant bit on left

```c
#include <stdio.h>

int main() {
  int a = 0x7fffffff;
  int as = a << 1;
  printf("shl of %d: %d(%08x)\n", a, as, as);


  unsigned b = 0x7fffffff;
  unsigned bs = b << 1;
  printf("shl of %u: %u(%08x)\n", b, bs, bs);
}
```

```
shl of 2147483647: -2(fffffffe)
shl of 2147483647: 4294967294(fffffffe)
```

Integers

Representation & Encoding

- for w bits data x

$$B2U(X) = \sum_{i=0}^{w-1} x_i 2^i \quad B2T(X) = -x_{w-1} * 2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

Conversion

Mapping Between `signed` and `unsigned`

Maintain bit pattern

so the conversion of negative value of the signed or value over the T_MAX results different interpreted value.

Expanding, Truncating

expanding of w-bit signed integer

$X = x_{w-1}x_{w-2}\cdots x_0$

where $s = x_{w-1}$

expanding it to $w+k$-bit signed int:

$X = s_k s_{k-1}\cdots s_1 x_{w-1}\cdots x_0, \quad \text{where } s_i = s$

truncation of w-bit signed integer

first k bit change

C Puzzle

```
int x = foo();
int y = bar();

unsigned ux = x;
unsigned uy = y;
```

- `x < 0` -> `((x*2) < 0)`
  - `false` : underflow
- `ux >= 0`
  - it's converted to `ux >= 0u` so `true` for all ux
- `x & 7 == 7` -> `(x << 30) < 0`
  - `true`
- `ux > -1`
  - `always false`, `-1 = uint_max`
- `x > y` -> `-x < -y`
  - `false`
- `x * x >= 0`
  - `false`
- `x > 0 && y > 0` -> `x + y > 0`
- `x >= 0` -> `-x <= 0`
- `x <= 0` -> `-x >= 0`
- `(x|-x) >> 31 == -1`
- `ux >> 3 == ux/8`
- `x >> 3 == x/8`
- `x & (x-1) != 0`
  - `true`