Lecture Note: Two's Complement

Yue Wu
December 03, 2025

---

1. Introduction

In standard mathematics, the set of integers $\mathbb{Z}$ is infinite. You can always add 1 to a number and get a larger number. However, computers operate with finite resources. A variable must only occupy a finite amount of RAM.

This means computers do not operate in $\mathbb{Z}$. Intuitively, we can choose to restrict on a subset of $\mathbb{Z}$, and prohibit all operations whose results are outside this subset. However, early hardware designers found that it's easier to let the result "overflow" and taking a subset of bits. This essentially results in a cyclic *cyclic ring* $\mathbb{Z}/n\mathbb{Z}$, where $n = 2^k$ and k is the number of bits in a primitive integer.

2. Cyclic rings

Two examples

In elementary math, you have experienced with some cyclic rings. For example:

- Even and odd numbers form such a ring. A even number plus a even number will always give a even number. The product of two odd numbers must also be odd.
- "The last (decimal) digit of numbers" form such a ring. Without calculating the entire thing, you know that $233 + 666$ will end in 9, because $3 + 6 = 9$; you know that 114514×19260817 will end in 8, because $4 \times 7 = 28$ ends in 8.

In both examples, we classified numbers into different classes: even numbers, or numbers that ends in 7. If you see a formula $a + b = c$, which class c belongs to will be fully determined by the class of a and b.

Naming them

In both examples, we say that two numbers belong to the same class, if they differ by a multiple of n. For even and odd numbers, $n = 2$, and we have two classes $\mathrm{even} = \{0, 2, 4, 6, \ldots\}$, and $\mathrm{odd} = \{1, 3, 5, 7, \ldots\}$.

For the last decimal digit of numbers, we don't have single-word names for them. Since $\mathbb{N} = \{0, 1, 2, \ldots\}$, we can write $10\mathbb{N} = \{0, 10, 20, \ldots\}$ to denote the class that ends in 0, and $7 + 10\mathbb{N} = \{7, 17, 27, \ldots\}$ to denote the class that ends in 7, etc.. (In this notation, if a is a number, S is a set, then $a + S$ is the set $\{a + s \mid s \in S\}$, and aS is the set $\{as \mid s \in S\}$. Then $10\mathbb{N}$ is all the natural number multiples of 10.)

In this way, we can say that even numbers are $2\mathbb{N}$ and odd numbers are $1+2\mathbb{N}$. In fact, in this way, for every positive integer n, we can classify $\mathbb{N}$ into $n\mathbb{N}, 1+n\mathbb{N}, ..., (n-1)+\mathbb{N}$.

With these names, we can rewrite "A even number plus a odd number will always give a odd number" into mathematical statements like $2\mathbb{N}+(1+2\mathbb{N})=1+2\mathbb{N}$. What does "$(2+10\mathbb{N})\cdot(3+10\mathbb{N})=6+10\mathbb{N}$" mean?

Extending to negatives

The fact "*which class $a+b$ belongs to will be fully determined by the class of a and b.*" extends naturally to negative numbers as well. If we simply replace $\mathbb{N}$ with $\mathbb{Z}$ in the above definitions, let's see what happens:

- In the $\mathbb{Z}/2\mathbb{Z}$ example, $2\mathbb{Z}=\{..., -4, -2, 0, 2, 4, ...\}$ is still all the even numbers in $\mathbb{Z}$, and $1+2\mathbb{Z}=\{..., -3, -1, 1, 3, ...\}$ is still all the odd numbers.
- In the $\mathbb{Z}/10\mathbb{Z}$ example, $10\mathbb{Z}=\{..., -20, -10, 0, 10, 20, ...\}$, but $3+10\mathbb{Z}=\{..., -27, -17, -7, 3, 13, 23, ...\}$ is no more the class of integers that end in 3. However, they can still calculate fine: $-7\in 3+10\mathbb{Z}, -18\in 2+10\mathbb{Z}$; please verify that $(-7)+(-18)\in 5+10\mathbb{Z}$ and $(-7)\times(-18)\in 6+10\mathbb{Z}$.

The collection $\{n\mathbb{Z}, 1+n\mathbb{Z}, ..., (n-1)+n\mathbb{Z}\}$ is called the quotient ring $\mathbb{Z}/n\mathbb{Z}$. (If you want to know what's a ring and what's a quotient precisely, feel free to read some basic abstract algebra!)

> **Quotient Ring**
>
> For a natural number n, we define:
> - $a+n\mathbb{Z}=\{..., a-2n, a-n, a, a+n, a+2n, ...\}$
> - and the quotient ring $\mathbb{Z}/n\mathbb{Z}=\{n\mathbb{Z}, 1+n\mathbb{Z}, ..., (n-1)+n\mathbb{Z}\}$.
> - For every integer x, there exists a number $a\in\{0, 1, ..., n-1\}$ such that $x\in a+n\mathbb{Z}$; in fact, this set is just $x+n\mathbb{Z}$.

Just as the $n=2$ and $n=10$ examples above, the basic operations (except division) are preserved in them:

> **Laws of a Quotient Ring**
>
> For a quotient ring $\mathbb{Z}/n\mathbb{Z}$, we have:
> - $(a+n\mathbb{Z})+(b+n\mathbb{Z})=(a+b)+n\mathbb{Z}$, and the latter is equal to some $p+n\mathbb{Z}$;
> - $(-a\mathbb{Z})=q+n\mathbb{Z}$ for some q;
> - $(a+n\mathbb{Z})(b+n\mathbb{Z})=(ab)+n\mathbb{Z}$, and the latter is equal to some $q+n\mathbb{Z}$.

> In the above, p, q, r are numbers in $\{0, 1, ..., n-1\}$.

In the video, the "2-digit calculator" does calculations on the $\mathbb{Z}/100\mathbb{Z}$ ring. We have shown that addition and multiplication works fine. In fact, when you see a k-bit computer, it does compute in $\mathbb{Z}/2^k\mathbb{Z}$. The "overflow" and "underflow" corresponds to the behavior of finding a representative element in $\{0, 1, ..., n-1\}$.

3. Naming the classes in a quotient ring

In a quotient ring, you don't have to choose $\{0, 1, ..., n-1\}$ as your representative. In fact, you can say that the two elements of $\mathbb{Z}/2\mathbb{Z}$ are $42 + 2\mathbb{Z}$ and $73 + 2\mathbb{Z}$, and it's just as valid.

In the video, this is called "naming the elements". Here, we say we are picking representative elements in the ring. To be exact, we consider the following representatives:

> **The "signed" representatives**
>
> For a quotient ring $\mathbb{Z}/n\mathbb{Z}$ where n is even:
> - $\{-\frac{n}{2}, -\frac{n}{2}+1, ..., -1\}$ are the negative representatives;
> - $\{0, 1, ..., \frac{n}{2}-1\}$ are the positive representatives.

This gives the $\{-50, -49, ..., 48, 49\}$ representative shown in the 2-digit example.

4. Two's Complement (Binary)

Modern computers use binary ($n = 2^k$). In CS terms, a k-bit integer has 2^k unique states; in math terms, the ring $\mathbb{Z}/2^k\mathbb{Z}$ has 2^k elements. Two of the integer types represents two ways of picking representatives in the elements of the ring.

- **Unsigned** Integers: Range $\{0, 1, ..., 2^k - 1\}$.
- **Signed** Integers: Range $\{0, 1, ..., 2^{k-1} - 1, -2^{k-1}, ..., -1\}$.

Finding negative

To find the negation of an element $-x + 2^k\mathbb{Z}$, we just want to find another number y that is a representative and that $-x + \mathbb{Z} = y + 2^k\mathbb{Z}$; this is equivalently $y \in -x + \mathbb{Z}$, and if x is in the range of $[1, 2^k - 1]$, we can just pick $y = -x + 2^k$.

This is then equal to $\left(2^k - 1\right) - x + 1$. However, when calculating $\left(2^k - 1\right) - x$, we can realize that $2^k - 1$ is the all-1 string, and subtracting from it is just flipping every bit. Therefore, $-x \equiv 2^k - x = \left(2^k - 1\right) - x + 1 = \text{bitwise\_negate}(x) + 1$.

The sign bit

The reason we set 2^{k-1} to be negative is that testing negativity is easy; we can look at the highest bit in the binary representation. For example, if we are dealing with $n = 2^4$, then the negative representatives are $\{1000 = -8, 1001 = -7, ..., 1111 = -1\}$, and they all have 1 as their highest bit; the nonnegative representatives are $\{0000 = 0, ..., 0111 = 7\}$.

5. Extra notes

5.1. The Asymmetry

Because 0 is technically "positive" (it has a 0 MSB), there is one more negative number than positive number.

- Max Positive: $011...1 = 2^{k-1} - 1$
- Max Negative: $100...0 = -2^{k-1}$

Note: You cannot take the negative of the most negative number (e.g., -128 in 8-bit). Mathematically, it is because $-128 + 256\mathbb{Z} = 128 + 256\mathbb{Z}$. So, negation maps it to itself.

5.2. Overflow & underflow

Since we have a very simple way of detecting negative-ness (the sign bit), it's easy to detect overflow and underflow. If two numbers has 0 at their highest bit and the addition result has 1 as their highest bit, there is an overflow. Underflow can be defined analogously.

5.3. Historical note

You might have heard of "sign-and-magnitude" and "one's complement" as alternative ways to encode negative integers. In the very early days of computing, they used to exist in some very old architectures. However, people quickly realized that two's complement makes signed addition and multiplication smooth, because they don't even need to exist separately. It is also mathematically the cleanest (for the same reason). Sign-magnitude exists in floating point numbers (IEEE754) and that's it.

All modern CPUs that you can buy (x86, ARM, RISC-V, or even MIPS, POWER) use two's complement. Modern programming languages like C20 and C++23 enforces two's complement. Therefore, I believe *only* two's complement should be taught in a computer architecture class; sign-magnitude and one's complement only belongs to computer history, along with segmented memory and branch delay slots.