



南京大學
NANJING UNIVERSITY



定点数的编码表示

南京大学

计算机科学与技术系

袁春风

email: cfyuan@nju.edu.cn

2015.6

数值数据的表示

- 数值数据表示的三要素
 - 进位计数制
 - 定、浮点表示
 - 如何用二进制编码
- 进位计数制
 - 十进制、二进制、十六进制、八进制数及其相互转换
- 定/浮点表示 (解决小数点问题)
 - 定点整数、定点小数
 - 浮点数 (可用一个定点小数和一个定点整数来表示)
- 定点数的编码 (解决正负号问题)
 - 原码、补码、移码、反码 (很少用)

原码 (Sign and Magnitude) 表示

Decimal	Binary	Decimal	Binary
0	0000	-0	1000
1	0001	-1	1001
2	0010	-2	1010
3	0011	-3	1011
4	0100	-4	1100
5	0101	-5	1101
6	0110	-6	1110
7	0111	-7	1111

“正” 号用0表示

“负” 号用1表示

数值部分不变!

◆ 容易理解, 但是:

- ✓ 0 的表示不唯一, 故不利于程序员编程
- ✓ 加、减运算方式不统一
- ✓ 需额外对符号位进行处理, 故不利于硬件设计
- ✓ 特别当 $a < b$ 时, 实现 $a - b$ 比较困难

从 50年代开始, 整数都采用补码来表示

但浮点数的尾数用原码定点小数表示

补码 - 模运算 (modular 运算)

重要概念：在一个模运算系统中，一个数与它除以“模”后的余数等价。

时钟是一种模12系统 现实世界中的模运算系统

假定钟表时针指向10点，要将它拨向6点， 则有两种拨法：

① 倒拨4格： $10 - 4 = 6$

② 顺拨8格： $10 + 8 = 18 \equiv 6 \pmod{12}$

模12系统中： $10 - 4 \equiv 10 + 8 \pmod{12}$

$-4 \equiv 8 \pmod{12}$

则，称8是-4对模12的补码 (即：-4的模12补码等于8)。

同样有 $-3 \equiv 9 \pmod{12}$

$-5 \equiv 7 \pmod{12}$ 等

结论1：一个负数的补码等于模减该负数的绝对值。

结论2：对于某一确定的模，某数减去小于模的另一数，总可以用该数加上另一数负数的补码来代替。

补码 (modular运算)：+ 和- 的统一

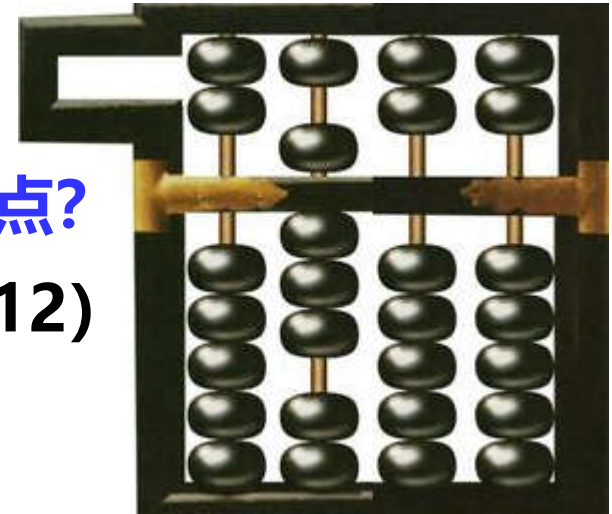
补码 (2's complement) 的表示

现实世界的模运算系统举例

例1: “钟表” 模运算系统

假定时针只能顺拨, 从10点倒拨4格后是几点?

$$10 - 4 = 10 + (12 - 4) = 10 + 8 = 6 \pmod{12}$$



例2: “4位十进制数” 模运算系统

假定算盘只有四档, 且只能做加法, 则在算盘上计算

9828-1928等于多少?

$$9828 - 1928 = 9828 + (10^4 - 1928)$$

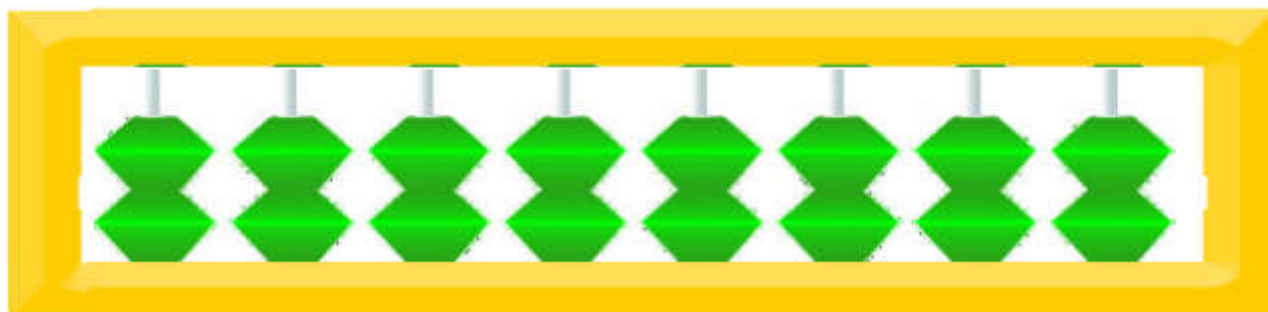
$$= 9828 + 8072$$

$$= \boxed{1}7900$$

$$= 7900 \pmod{10^4}$$

取模即只留余数, 高位“1”被丢弃!
相当于只有低4位留在算盘上。

计算机中的运算器是模运算系统



8位二进制加法器模运算系统

$[-0100\ 0000]_{\text{补}} = ?$

计算 $0111\ 1111 - 0100\ 0000 = ?$

$$\begin{aligned} 0111\ 1111 - \boxed{0100\ 0000} &= 0111\ 1111 + \boxed{(2^8 - 0100\ 0000)} \\ &= 0111\ 1111 + \boxed{1100\ 0000} = \boxed{1}0011\ 1111 \pmod{2^8} \\ &= 0011\ 1111 \end{aligned}$$

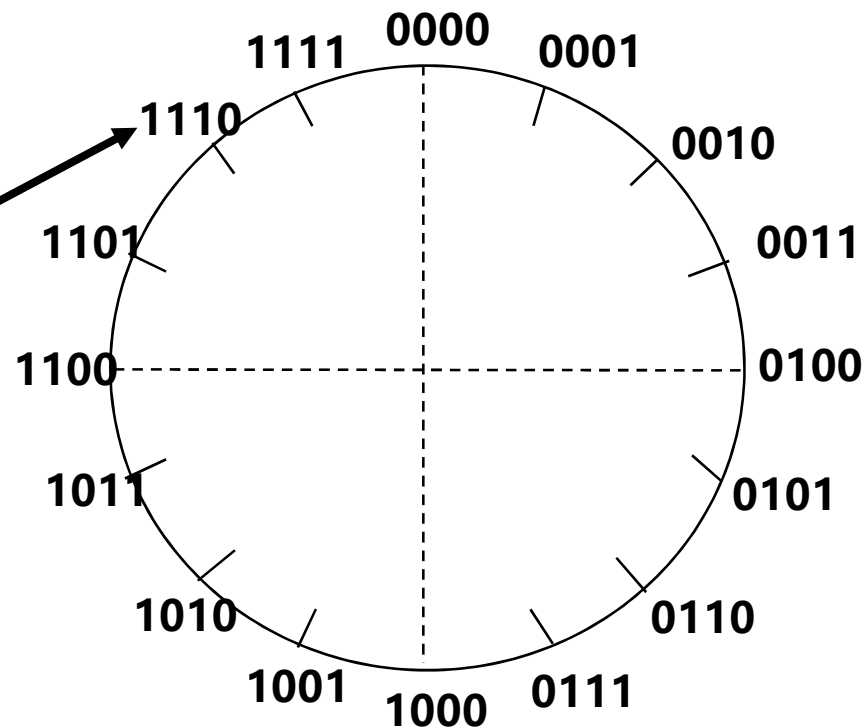
只留余数，“1”被丢弃

结论1： 一个负数的补码等于将对应正数补码
各位取反、末位加一

运算器适合用补码表示和运算

运算器只有有限位，假设为 n 位，则运算结果只能保留低 n 位，故可看成是个只有 n 档的二进制算盘，因此，其模为 2^n 。

当 $n=4$ 时，共有16个机器数：
0000 ~ 1111，可看成是模为
 2^4 的钟表系统。真值的范围为
 $-8 \sim +7$



补码的定义 假定补码有 n 位，则：

$$[X]_{\text{补}} = 2^n + X \quad (-2^{n-1} \leq X < 2^{n-1}, \text{ mod } 2^n)$$

X 是真值， $[x]_{\text{补}}$ 是机器数

真值和机器数的含义是什么？

求特殊数的补码

假定机器数有n位

$$\textcircled{1} [-2^{n-1}]_{\text{补}} = 2^n - 2^{n-1} = 10\dots0 \text{ (n-1个0)} \pmod{2^n}$$

$$\textcircled{2} [-1]_{\text{补}} = 2^n - 0\dots01 = 11\dots1 \text{ (n个1)} \pmod{2^n}$$

$$\textcircled{3} [+0]_{\text{补}} = [-0]_{\text{补}} = 00\dots0 \text{ (n个0)}$$

32位机器中，int、short、char型数据的机器数各占几位？

32位、16位、8位

变形补码 (4's complement) 的表示

补码定义: $[X]_{\text{补}} = 2^n + X \quad (-2^n \leq X < 2^n, \text{ mod } 2^n)$

- 正数: 符号位 (sign bit) 为0, 数值部分不变
- 负数: 符号位为1, 数值部分 “各位取反, 末位加1”

变形 (4's) 补码: 双符号, 用于存放可能溢出的中间结果。

Decimal	补码	变形补码	Decimal	Bitwise Inverse	补码	变形补码
0	0000	00000	-0	1111	0000	00000
1	0001	00001	-1	1110	1111	11111
2	0010	00010	-2	1101	1110	11110
3	0011	00011	-3	1100	1101	11101
4	0100	00100	-4	1011	1100	11100
5	0101	00101	-5	1010	1011	11011
6	0110	00110	-6	1001	1010	11010
7	0111	00111	-7	1000	1001	11001
8	1000	01000	-8	0111	1000	11000

值太大, 用4位补码无法表示, 故 “溢出”
但用变形补码可保留符号位和最高数值位

+0和-0表示唯一

求真值的补码


例: 设机器数有8位, 求123和-123的补码表示。

如何快速得到123的二进制表示?

解: $123 = 127 - 4 = 01111111\text{B} - 100\text{B} = 01111011\text{B}$

$-123 = -01111011\text{B}$

$[01111011]_{\text{补}} = 2^8 + 01111011 = 100000000 + 01111011$
 $= 01111011 \pmod{2^8}$, 即 7BH。

$[-01111011]_{\text{补}} = 2^8 - 01111011 = 10000\ 0000 - 01111011$

 $= 1111\ 1111 - 0111\ 1011 + 1$
 $= 1000\ 0100 + 1$ ← 各位取反, 末位加1
 $= 1000\ 0101$, 即 85H。

简便方法: 从右向左遇到第一个1的前面各位取反

当机器数为16位时, 结果怎样? $\text{mod} = 2^{16}$

求补码的真值

令： $[A]_{\text{补}} = a_{n-1}a_{n-2}\cdots a_1a_0$

则： $A = -a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0$

例如：补码 “11010110” 的真值为

$$-2^7 + 2^6 + 2^4 + 2^2 + 2 = -128 + 64 + 16 + 4 + 2 = -42$$

补码 “01010110” 的真值为

$$-0 \cdot 2^7 + 2^6 + 2^4 + 2^2 + 2 = 64 + 16 + 4 + 2 = 86$$

理论上的求法

简便求法：

符号为0，则为正数，数值部分相同

符号为1，则为负数，数值各位取反，末位加1

例如：补码 “01010110” 的真值为

$$+1010110 = 64 + 16 + 4 + 2 = 86$$

例如：补码 “11010110” 的真值为

$$\begin{array}{ccccccc} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ -0 & 1 & 0 & 1 & 0 & 1 & 0 \\ & & & & & & \uparrow \end{array} \quad -0101010 = -(32 + 8 + 2) = -42$$

移码表示 Excess (biased) notion

- 什么是移码表示？

将每一个数值加上一个偏置常数 (Excess / bias)

- 通常，当编码位数为 n 时，bias取 2^{n-1} 或 $2^{n-1}-1$ (如 IEEE 754)

Ex. $n=4$: $E_{\text{biased}} = E + 2^3$ (bias = $2^3 = 1000\text{B}$)

-8 (+8) ~ 0000B

-7 (+8) ~ 0001B

...

0 (+8) ~ 1000B

...

+7 (+8) ~ 1111B

0的移码表示唯一

当bias为 2^{n-1} 时，移码和补码仅第一位不同

移码用来表示浮点数的阶

- 为什么要用移码来表示指数（阶码）？

便于浮点数加减运算时的对阶操作 (比较大小)

例: $1.01 \times 2^{-1} + 1.11 \times 2^3$

$1.01 \times 2^{-1+4} + 1.11 \times 2^{3+4}$

补码: 111 < 011 ?
(-1) (3)

简化比较

移码: 011 < 111
(3) (7)