有符号定点数的编码表示——补码

计算机中所有的运算都是有模运算

例： $(1423)_{10} \bmod 10^2 = 23$

$(1101)_2 \bmod 2^3 = (101)_2$

定义：设二进制数 x 多于 n 位，**模运算**操作为：

x **模** $2^n \Leftrightarrow$ 丢弃数 x 高位而保留低 n 位

计算机中的运算部件有**字长限制** n ，所有的运算都是**有模运算**，只能保留低 n 位。

$(x*y) \bmod 2^n$

两种类型机器数的模

设机器字长为 n 位，则下列**机器数**的模，

定点整数的模：$M=2^n$

n-1	n-2	………	2	1	0

定点小数的模： $M=2$

0	-1	-2	………	-n+2	-n+1

补的概念

例：假定现在钟表时针指向 6 点，要将它拨向 3 点，则有两种拨法：

① 逆拨 3 格：$6-3=3 \pmod{12}$

② 顺拨 9 格：$6+9=3 \pmod{12}$

说明：在模 12 系统中，

$6-3 \equiv 6+9 \pmod{12}$

可见 -3 可用 $+9$ 代替减法 ⟶ 加法

称 9 是 -3 以 12 为模的补码。

记作 $-3 \equiv +9 \pmod{12}$

同理 $-5 \equiv +7 \pmod{12}$

补的概念

B 和 A **模 M 相等**： $A=B+kM$ (k 为整数)

记为： $A \equiv B \pmod{M}$

例： $-4 \equiv 8 \pmod{12}$

称 8 是 -4 以 12 为模的**补码**。

说明：这里只讨论**绝对值小于模**的数。

在有模运算中：

一个负数加上“**模**”即得该负数的**补码**。

负数及其补码的绝对值之和为“**模**”。

正数的补码即为其本身。

补码表示法

结论：在有模运算中，补码可以**用加法实现减法运算**。

$$10-4 \equiv 10+8 \pmod{12}$$

例 4位加法器（模 16）

$$1010-0011 \equiv 1010+(2^4-0011)$$

$$\equiv 1010+1101$$

$$\equiv 0111 \pmod{2^4}$$

$$\begin{array}{r} 1010 \\ +\ 1101 \\ \hline \boxed{1}0111 \end{array}$$

自然去掉

补码的定义

定义：[X]补 = M+X(mod M)；|X|≤M/2

定点小数（纯小数）：|X|≤M/2=1

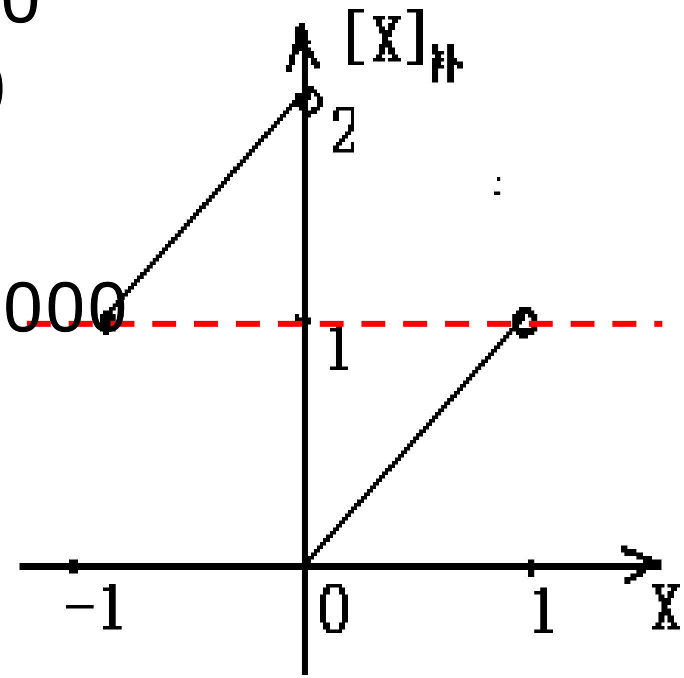
$$[x]_{补} = \begin{cases} x & 1 > x \quad 0 \\ 2 + x & 0 > x \quad -1 \end{cases}$$

[+0.1001]补 =0.1001000

[-0.1001]补 =1.0111000

[0]补 = 0.0000000

[-1.0000000]补 =1.0000000



补码的定义

定点整数（纯整数）：略.

两种类型定点数表示范围：

机器数（n 位）	定点整数	定点小数
00…0~01…1	$0\sim2^{n-1}-1$	$0\sim1-2^{-(n-1)}$
10…0~11…1	$-2^{n-1}\sim-1$	$-1\sim-2^{-(n-1)}$

$M=2^n$

n-1	n-2	……	2	1	0

$M=2$

0	-1	-2	……	-n+2	-n+1

补码的特点

1. 最高位**可看作**符号位，0 正 1 负；
2. 数 0 的补码是唯一的：00…0 ；
3. 负数补码的表示范围比原码多一种组合：-1 ，-2^{n-1} ，10…0

求负数补码的简单方法

方法：由真值、原码求补码

过程：真值 → 原码 → 补码

1) 正数的补码与原码相同

例：$X_{原}=0.1010000$，$X_{补}=0.1010000$

2) 负数原码 → 补码：

① 符号位不变，其余各位变反，末位加 1；

② 符号位不变，尾数自低位向高位，第一个 1 及以前的 0 保持不变，其余各位变反。

例：设机器数长度为 8，X=-0.1011010，求 $[X]_{补}$。 $[X]_{原}=1.1011010$

$[X]_{补}=1.0100101+0.0000001$

$=1.0100110$

由补码求真值

由补码求真值、原码仍采用上两种方法做转换：

1) 正数的原码与补码相同；

2) 负数补码 → 原码：

① 符号位不变，其余各位变反，末位加 1 ；

② 符号位不变，尾数自低位向高位，第一个 1 及以前的 0 保持不变，其余各位变反。

例．设机器数长度为 8 ，$[X]_{补}$=10110100，求 X 。

$[X]_{原}$=11001100

x= -1001100

变补

变补（**求补**）：由 $[Y]_{补}$ 求 $[-Y]_{补}$ 称作对 $[Y]_{补}$ 求补或变补。

方法：

1) 将 $[Y]_{补}$ 连同符号位一起各位取反，末尾加 1 ，不论 $[Y]_{补}$ 本身为正或负。
2) 略。

例：设机器数长 8 位，最高位是符号位。已知 $[x]_{补}$=0000 0011 ，$[y]_{补}$=1111 1011 ，求，

$[-x]_{补}$ = 1111 1101

$[-y]_{补}$ = 0000 0101

补码表示法

补码表示的实质：将负数映射到正数域，利用有模运算，实现化减为加、简化运算的目的；

计算机采用补码表示作为运算基础。

反码表示法

1. 正数的反码与原码、补码相同；
2. 负数的反码：
   1) 原码符号位不变(1)，其余各位按位变反；

   例：求反码。

   $[+0.1011]_{反} = 0101\ 1000$

   $[-0.1001]_{反} = 1011\ 0111$

   $[-1011]_{反} = 1111\ 0100$

   2) 0 的反码有 +0 和 -0 两种形式。

   $[+0]_{反} = 0\ 0…0$　$[-0]_{反} = 1\ 1…1$

   3) 反码表示范围同原码，表示2^n-1个数。

三种编码的比较

1. 三种编码都是为了解决负数在机器中的表示而提出的。

正数：原码＝补码＝反码＝真值，符号位（最高位）都是０；

负数：符号位均为“1”，数值位则各有不同的表示：

三种编码的比较

2. 原码和反码都有 +0 和 -0 两种零的表示，而补码可唯一表示 0 。
3. 补码和反码的符号位作为数值的一部分，和数值位一起参加运算。而原码的符号位必须和代表绝对值的数值位分开处理。
4. 原码和反码能表示的正数和负数的范围相对零来说是对称的。补码的表示范围不对称，负数表示的范围较正数宽，能多表示一个最小负数：-2^{n-1} 或 -1 。

定点与浮点表示

如何表示实数小数点的位置？

定点表示

定点表示：约定**所有数据**的小数点位置固定不变。

小数点位置在计算机设计时被**隐含地规定**，不需要用任何硬件设备明显表示小数点。

±	1	0	1	0	1	1	0	1

小数点位置（隐含约定）

±	1	0	0	1	0	0	1	0

小数点位置（隐含约定）

定点表示

利用定点表示进行计算：如定点机。

1. 表示范围有限（**较小**），运算很容易产生**溢出**；
2. 须将所有数值按一定**比例**予以缩小（或放大），**规范**为约定的定点数格式后才能送入计算机运算；
3. 同时须将**计算结果**以同一比例增大（或缩小）后才能得正确结果值。

例，1.2*3.45=12/10 * 345/100 = 12*345/1000

4. 选择适当的**比例因子**有时也很困难。

   如，定点小数做加法，3.45+8.23。

浮点表示

浮点数：小数点位置不固定（浮动）。**比例因子**包含在数中。

任意一个二进制数 x ，可以表示成如下形式：

$$x = M \times R^{E}$$

M ：纯小数—尾数

E ：纯整数—阶码

R ：底，**约定**为 2,4,8,16 等(2^q)

$0.101 \times 2^{-8} = 0.101 \times 4^{-4}$

0.0000 0000 101 =

$0.11 \times 2^{32} = 0.11 \times 16^{8}$

$1.1 \times 2^{31} =$

浮点数格式

E_f	E_1	E_2	…… E_e	M_f	M_1	M_2	……	M_m	

阶码 E

尾数 M

阶符

数符

真值： $x = M \times R^E$

M ：**尾数**，用**定点小数**表示，位数决定了浮点数的**表示精度**——有效数字的位数 m;

E ：**阶码**，用**定点整数**表示，指明小数点在数据中的位置，位数决定了浮点数的**表示范围**；

R ：阶码的**底**，**隐含**约定。一般与尾数的基数相同 (2) ，也可以为 2^q。

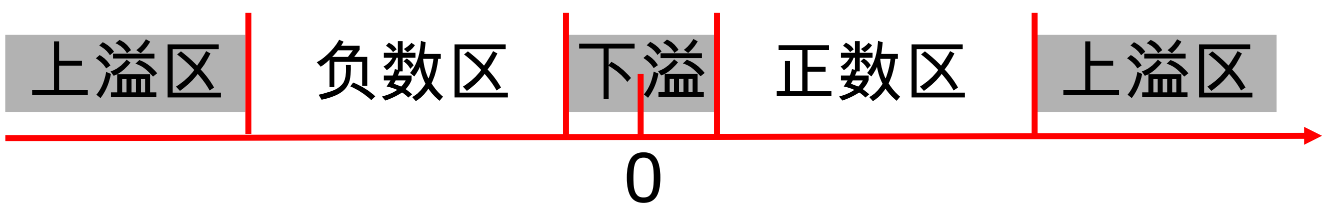
R^E ：尾数的**比例因子**。

浮点表示

E_f	E_1	E_2	……	E_e	M_f	M_1	M_2	……	M_m	

阶码 E　　　　　　　尾数 M

$$\left.\begin{array}{l} |X|_{max} = (1-2^{-m}) \times 2^{(2^e-1)} \\ |X|_{min} = 2^{-m} \times 2^{-(2^e-1)} \end{array}\right\} \text{原码表示}$$



特点：可以表示很大的数据范围以及较高的数据精度。

浮点数的编码表示

浮点数实际上是用**两个定点数**表示一个数值数据：

尾数：定点小数，补码或原码表示；

阶码：定点整数，一般用**移码**表示，便于比较大小——通过机器数比较真值。

问题：补码、原码为什么不行？它们的大小是怎么样的？