整数的编码

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

- 标准的C语言中，各整数类型所占用的存储空间为:

int	4字节：-2^{31} ~ $(2^{31}-1)$
short	2字节：-2^{31} ~ $(2^{31}-1)$
long	4个字节（32位机器）-2^{31} ~ $(2^{31}-1)$， 8个字节（64位机器）-2^{63} ~ $(2^{63}-1)$，

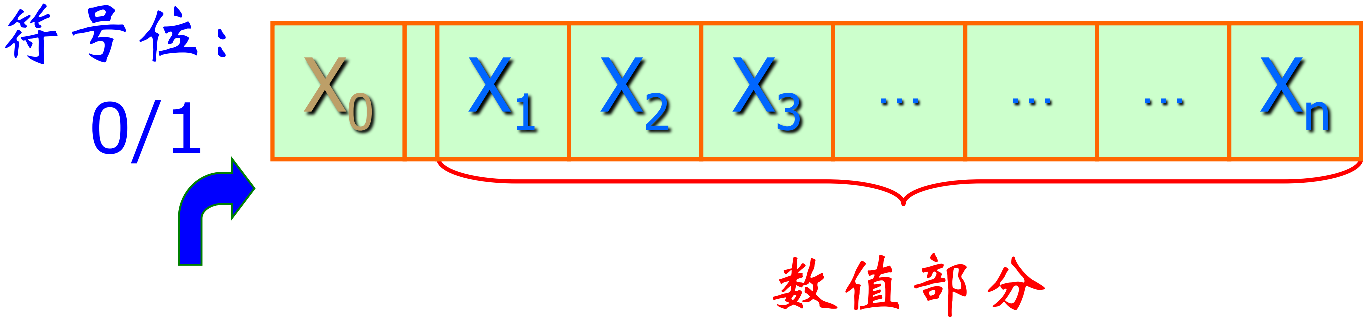
整数在机器内部的编码

原码 Signed magnitude

反码 One’s complement

补码 Two’s complement

原码表示法



- 最高位：符号位，0为正，1为负
- 剩余的位数：数值部分对应的编码

$[14]_{10} = [1110]_2$

$[1110]_{原} = 0\ 1110$

$[-14]_{10} = [-1110]_2$

$[-1110]_{原} = 1\ 1110$

0有两种编码: 正0和负0

$[+0]_{原}=0\ 000\cdots0$

$[-0]_{原}=1\ \ 000\cdots0$

原码缺点：不方便计算

$$
\begin{array}{rcr}
01011001_2 & = & 89_{10} \\
+\ \underline{11001101_2} & = & \underline{-77_{10}} \\
00100110_2 & = & 32_{10}
\end{array}
$$

反码（one's complement）

正数符号位：0

负数符号位：1

X_0	X_1	X_2	X_3	…	…	…	X_n

正数数值部分不变

负数数值部分取反

- 符号位: 正数为0，负数为1.
- 数值位: 正数维持编码不变，负数的各位数码取反
- 例: $[7_{10}]_{反} = 00111_2$ ； $[-7_{10}]_{反} = 11000_2$

反码的缺点

- 两个零：正零和负零
  - 0x00000000 = $+0_{ten}$
  - 0xFFFFFFFF = -0_{ten}

- 反码不能直接运算得到结果的反码

补码（Two's complement)

正数符号位：0

负数符号位：1

X_0		X_1	X_2	X_3	…	…	…	X_n

正数数值部分不变

负数数值部分取反+1

- 符号位：正数为0，负数为1
- 数值位：正数维持编码不变，负数的各位数码取反再加1
- 例: $[7_{10}]_{反}= 0\ 0111_2$ ； $[-7_{10}]_{反}=1\ 1000_2$； $[-7_{10}]_{补}=1\ 1001_2$

int在机器内的编码：补码

$0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$

$0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = +\ 1_{ten}$

$0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = +\ 2_{ten}$

...

$1\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -\ 2_{ten}$

$1\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -\ 1_{ten}$

...

$0\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = +\ 2{,}147{,}483{,}647_{ten}$

$0\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = +\ 2{,}147{,}483{,}646_{ten}$

$1\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -\ 2{,}147{,}483{,}648_{ten}$

$1\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -\ 2{,}147{,}483{,}647_{ten}$

补码：正数和负数相加结果为0

- -8 = 0-8，　8的二进制是00001000，求 -8：

```
  0 0 0 0 0 0 0 0
- 0 0 0 0 1 0 0 0
- - - - - - - - -

1 0 0 0 0 0 0 0 0
- 0 0 0 0 1 0 0 0
- - - - - - - - -
  1 1 1 1 1 0 0 0     (-8)补
```

```
  1 1 1 1 1 1 1 1
- 0 0 0 0 1 0 0 0
- - - - - - - - -
  1 1 1 1 0 1 1 1       (取反)
+ 0 0 0 0 0 0 0 1       (加 1)
- - - - - - - - -
  1 1 1 1 1 0 0 0       (-8)补
```

一个负数的补码的两个转换步骤：取反，加1

补码：2的补码

$$[X]_{补}=\begin{cases} X & 0\le X<2^n \\ 2^{n+1}+X & -2^n\le X<0 \end{cases}$$

补码编码长度为n+1位

例如：编码长度为4，表达的数据范围为-8 ~ +7

5的补码：0101

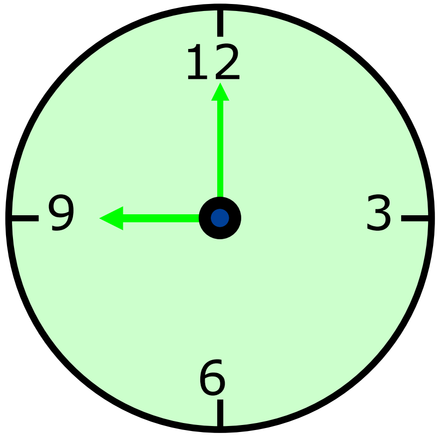
-5的补码：16 ＋（-5）＝ $[11]_{10}=[1011]_2$

$$\begin{array}{rrr} & 5 & 0101 \\ & -5 & 1011 \\ \hline = & 0 & 1\,0000 \end{array}$$

长度为4的补码，是一个模为16的计量系统

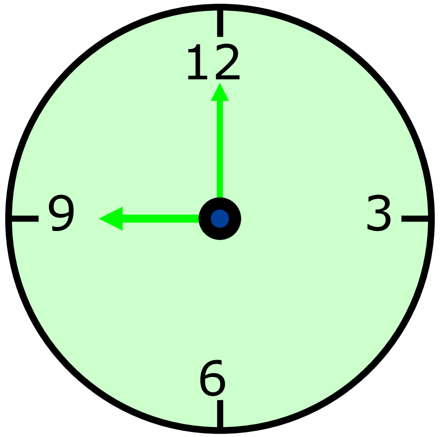
时钟的计数范围是0~11，模是12

模：产生溢出的量。
当运算结果超出计数范围时，则会执行求模运算

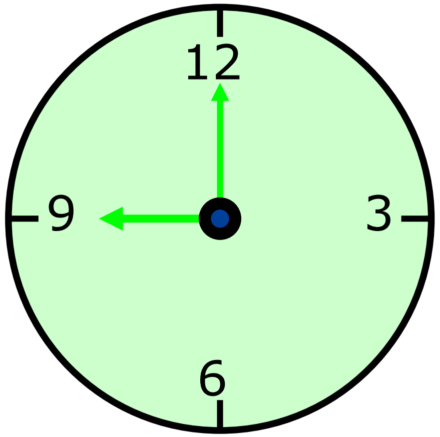


9+24= 9

9-24= 9



9+9 = 6



9-3=6

在一个有模的计量系统里，减法运算可以转变为加法运算

例如，在模为12的系统里 计算 7-4

可转变为：

=7+(12-4)

=7+8

=15

=12+3 （舍弃模， 12 ）

= 3 （结果为3）

int：以2^{32}为模的计量系统

$$
\begin{array}{rl}
 & 0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = +\ 2_{补} \\
+ & 1\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -\ 2_{补} \\
\hline
 & 1,\ 0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 2^{32} = 0
\end{array}
$$

$$
\begin{array}{rl}
 & 0\ 000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = +\ 2_{补} \\
+ & 1\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -\ 3_{补} \\
\hline
 & 1\ 111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -\ 1_{补}
\end{array}
$$

补码的特点

- 零是唯一的，没有正零和负零的区别
- 负数比正数多一个
- 补码是以2^{n+1} 为模的计量系统
- $[X + Y]_{补} = [X]_{补} + [Y]_{补}$
- 减法可以转换为加法
  - $[X-Y]_{补} = [X]_{补} + [-Y]_{补}$
- 符号位可以直接参与运算
- 计算机中广泛采用补码表达有符号整数。

小结

- 整数类型： int，long , short
  - 存储长度和表示范围
- 原码、反码、补码的编码规则的特点

谢谢！