第1章　数制与编码

名词：

- 数字（数码）:用来记数并可用来表示数量大小的符号。
- 数制：多位数码中每一位的构成方法以及从低位到时高位的进位规则。

常用数制

数值	十进制	二进制	十六进制	八进制	数值	十进制	二进制	十六进制	八进制
零	0	0	0	0	十	10	1010	A	12
一	1	1	1	1	十一	11	1011	B	13
二	2	10	2	2	十二	12	1100	C	14
三	3	11	3	3	十三	13	1101	D	15
四	4	100	4	4	十四	14	1110	E	16
五	5	101	5	5	十五	15	1111	F	17
六	6	110	6	6	十六	16	10000	10	20
七	7	111	7	7	十七	17	10001	11	21
八	8	1000	8	10	十八	18	10010	12	22
九	9	1001	9	11	十九	19	10011	13	23

数的表示方法：

- 并列法：3210.123
- 多项式表示法：

$$3\times10^{3}+2\times10^{2}+1\times10^{1}+0\times10^{0}+1\times10^{-1}+2\times10^{-2}+3\times10^{-3}$$

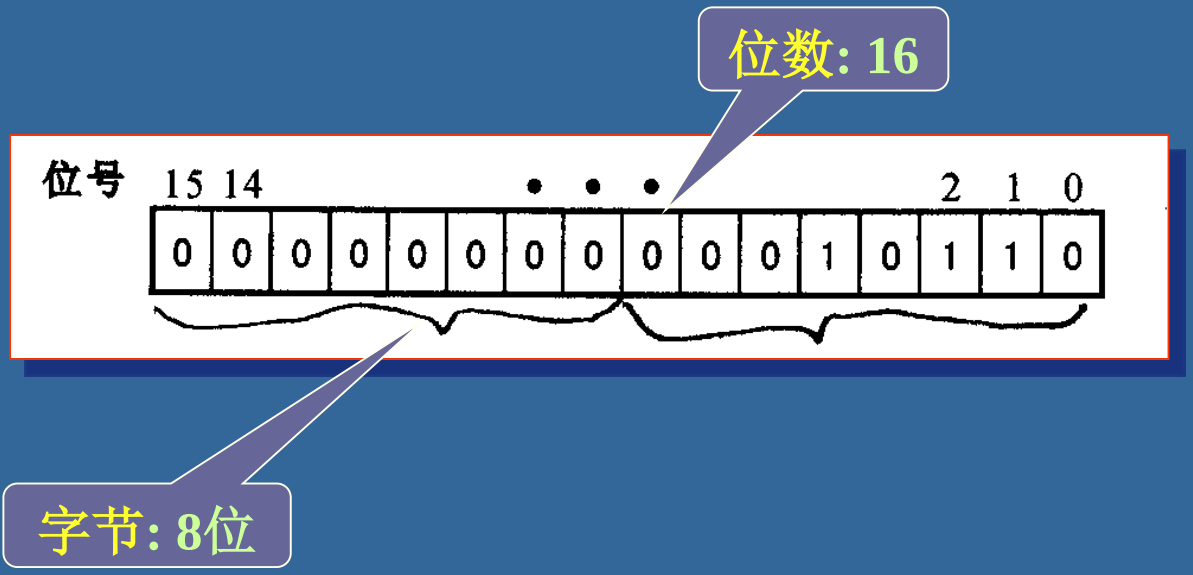
十进制数

位号	3	2	1	0		−1	−2	−3
十进制数	3	4	5	6	.	7	8	9
位权	10^{3}	10^{2}	10^{1}	10^{0}		10^{-1}	10^{-2}	10^{-3}

二进制数：

例： $10110 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

无符号二进制数的存放



字长: 二进制数包含的位数或字节数。

八进制数：

符号0、1、…7和小数点，且逢八进一，8为基，8^i 称为第 i 位上的权。

例： $(37.6)_8=(37.6)_O=3\times8^1+7\times8^0+6\times8^{-1}=(31.75)_{10}$

十六进制数：

符号0、1、…9、A、B、C、D、E、F和小数点，且逢十六进一，16为基，16^i称为第 i 位上的权。

例： $(B1F.8)_{16}=(B1F.8)_H=11\times16^2+1\times16^1+15\times16^0+8\times16^{-1}$

$=(2847.5)_{10}$

二—十进制间的转换

(1) 二进制数 → 十进制数

$$(101.11)_2 = 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2} = (5.75)_{10}$$

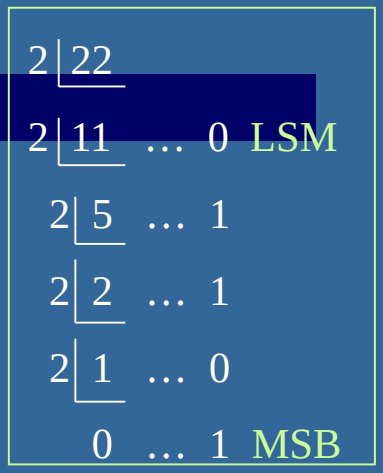
(2) 十进制数 → 二进制数

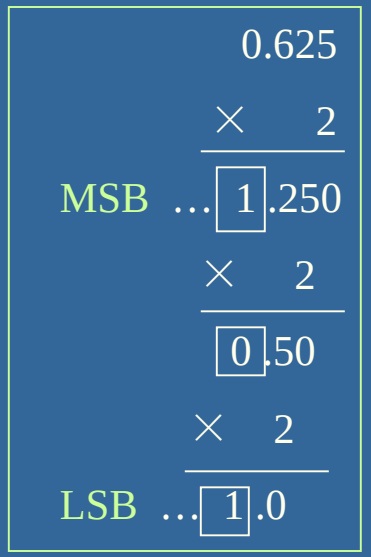
整数部分用基数除法，小数部分用基数乘法，小数部分算到r位误差小于2^{-r}。

例1：$(22.625)_{10}=(10110.101)_2$

例2：$(0.71)_{10}=(0.101101)_2$
(六位二进制小数)

例3：$(0.71)_{10}=(0.10110101)_2$
(误差小于5‰)





二—八—十六进制之间的转换

例1： $(1100101.11)_2=(\underline{0110}\ \underline{0101}.\underline{1100})_2=(65.C)_{16}$

$=(\underline{001}\ \underline{100}\ \underline{101}.\underline{110})_2=(145.6)_8$

例2： $(22)_{10}=(10110)_2=(16)_{16}=(26)_8$

二进制的运算

1101
+1011
11000

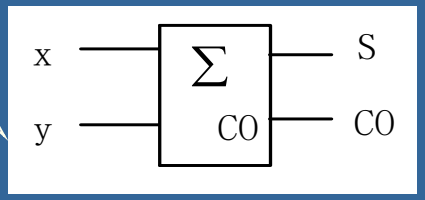
加法

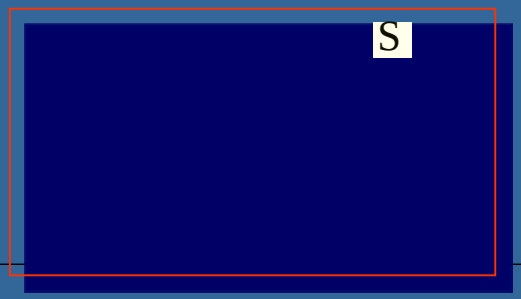
最低位：本位相加（无进位加）。

其他位：除本位相加外，再加低位的进位（带进位加）。

二进制加法规则

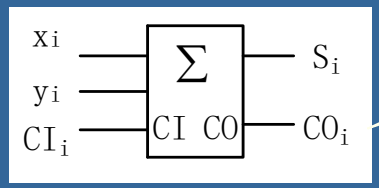
半加器





（输入信号X为1位被加数、Y为1位加数，输出信号S、CO分别是和数及向高位的进位 ）

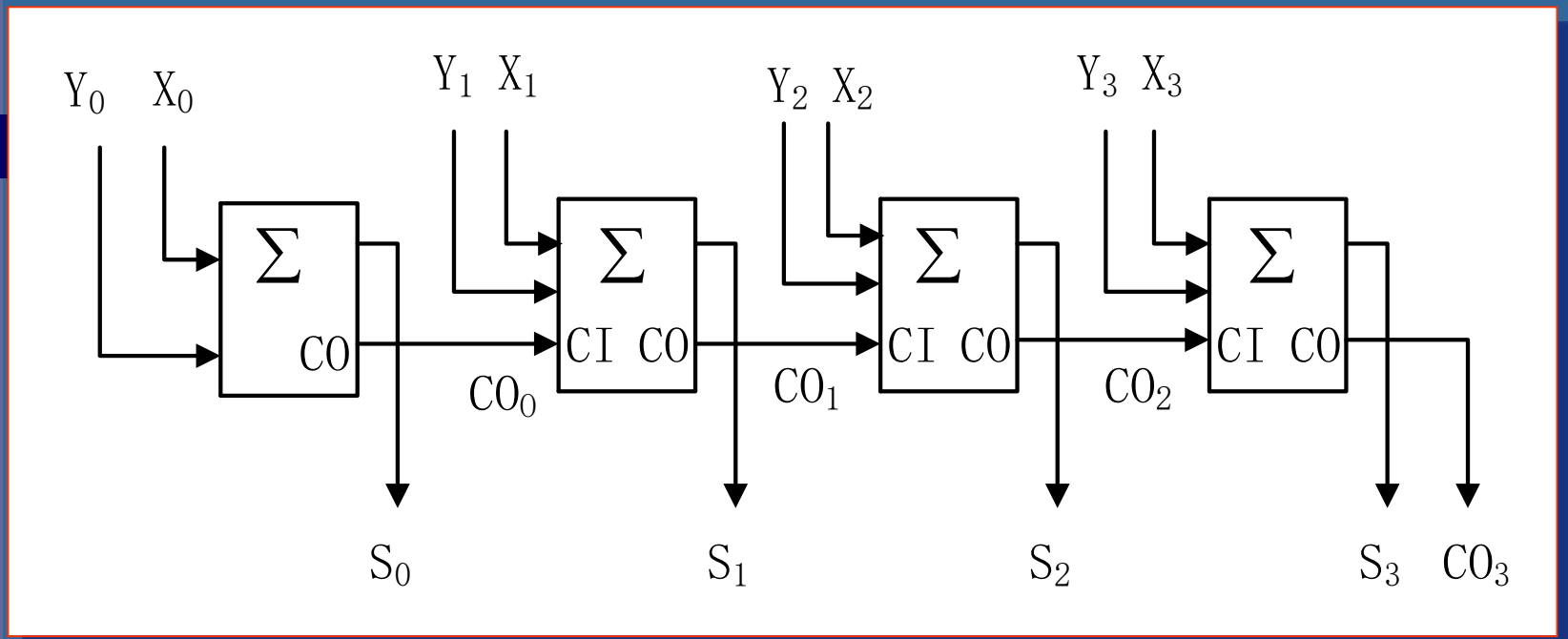
全加器



CI_i	x_i	y_i	CO_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

（被加数X_i、加数Y_i和低位来的进位CI_i，输出信号是本位和S_i及向高位的进位CO_i ）

四位全加器



■ 减法

最低位： 本位相减。

其他位： 除本位相减外，再减低位的借位 （带借位减）。

二进制减法规则

x	y	b	d
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

```
  1101
- 1011
------
  0010
```

说明： 通常利用所谓二进制的**补码或反码**相加来完成减法运算。

乘法、除法

乘法： 当乘数为2^r时，积为被乘数左移r位。

除法： 当除数为2^r时，商为被除数右移r位。

```
      1 0 1 0
  ×       1 0 0
-------------------
      0 0 0 0
    0 0 0 0
+ 1 0 1 0
-------------------
  1 0 1 0 0 0
```

$(1010)_2 \times (100)_2 = (101000)_2$

```
        1 0.1
     ------------
100 | 1 0 1 0
      1 0 0
    ------------
        1 0 0
        1 0 0
    ------------
        0 0 0
```

$(1010)_2 \div (100)_2 = (10.1)_2$

可见：乘法运算可以由加法运算和左移操作来完成，除法运算可以由减法运算和右移操作来完成。

编 码

定义：

- 广义上：用文字、符号或者数码来表示某种信息的过程叫编码。

（由编码得到的表示给定的信息的符号串称为代码，符号串中的各符号称为码元，符号的位数称为码长。

在数字系统中，任何信息都是由若干位“0”和“1”组成的，这种编码称为二值编码。码长为n的二值编码，它的n位码元可组成2^n种不同的代码，代表2^n种不同的信息。）

例：A　1000001

- 用一组二进制码按一定规则排列起来以表示数字、符号等特定信息。

- **代码**：利用数码来作为某一特定信息的代号
- **编码**：代码的编制过程称之编码。
- **码制**：在编码时所遵循的规则。

BCD码编码

十进制数	8421	2421	631－1	余3码	格雷码	5中取2码	左移码
0	0000	0000	0011	0011	0010	00011	00000
1	0001	0001	0010	0100	0110	00101	10000
2	0010	0010	0101	0101	0111	00110	11000
3	0011	0011	0111	0110	0101	01001	11100
4	0100	0100	0110	0111	0100	01010	11110
5	0101	1011	1001	1000	1100	01100	11111
6	0110	1100	1000	1001	1101	10001	01111
7	0111	1101	1010	1010	1111	10010	00111
8	1000	1110	1101	1011	1110	10100	00011
9	1001	1111	1100	1100	1010	11000	00001

部分字符的ASCII码

字符	ASCII码	字符	ASCII码	字符	ASCII码
空	0100000	4	0110100	K	1001011
.	0101110	5	0110101	L	1001100
(	0101000	6	0110110	M	1001101
+	0101011	7	0110111	N	1001110
$	0100100	8	0111000	O	1001111
*	0101010	9	0111001	P	1010000
)	0101001	A	1000001	Q	1010001
–	0101101	B	1000010	R	1010010
/	0101111	C	1000011	S	1010011
,	0101100	D	1000100	T	1010100
'	0100111	E	1000101	U	1010101
=	0111101	F	1000110	V	1010110
0	0110000	G	1000111	W	1010111
1	0110001	H	1001000	X	1011000
2	0110010	I	1001001	Y	1011001
3	0110011	J	1001010	Z	1011010

数制：表示数值的符号与规则

编码：表示信息的符号与规则

(PC机键盘)

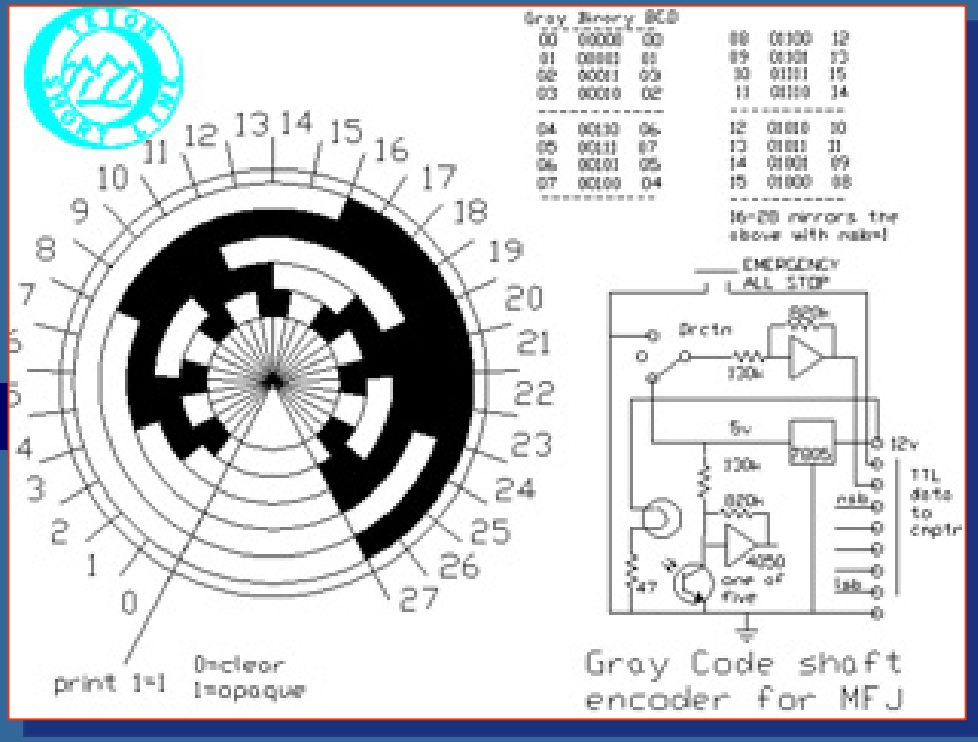
自然二进制码　按自然数顺序排列的二进制码

常用四位自然二进制码，表示十进制数0--15，各位的权值依次为2^3、2^2、2^1、2^0。

十进制数	(自然)二进制码
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

自然二进制码的缺点： 当一个代码变为相邻代码时，如欲由**1001**变为**1010**，由于实际电路中各个码元的变化总有先有后，难以做到绝对地“同时”变化，若**1001**的最低位**1**先变成**0**，然后次低位**0**再变成**1**，则**1001**变成**1010**的变化过程是：**1001→1000→1010**，出现了误码**1000**。

格雷码



格雷码（又称循环码或反射码），是美国贝尔实验室的数学家弗兰克·格雷（Frank Gray）在二战期间为解决采用脉码调制方式PCM的无线电通信中，由于线路中的脉冲干扰而造成误码率太多这一严重问题而提出的，据此发明的格雷编码管（Grag cooler Tube）于1953年3月17日获得了美国专利。因此，格雷是对现代通信技术做出了特殊贡献的数学家。

格雷码的特点：在计数过程中，任何相邻的两个码，只有一个数位（又称码元）不同。

循环码

格雷码之所以在通信中获得广泛应用，一是因为相邻码只有一位发生变化，也就是只有一根线上有脉冲变化（在格雷编码管中），干扰减少，使通信的出错概率大大降低，二是码的生成和变换比较简单。

格雷码的特性：

（1）循环性

（2）反射性

十进制数	循环码
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

从自然二进制变换为二进制格雷码的规则：

（1）两种代码的最高位（即最左边一位）相同；

（2）从高位至低位依次读取二进制码的各位码元。若某位码元与其前一位不同,则该位对应的格雷码的码元为**1**，否则为**0**。

某二进制数为 $B_{n-1}B_{n-2}\cdots B_2B_1B_0$

其对应的格雷码为 $G_{n-1}G_{n-2}\cdots G_2G_1G_0$

异或运算：

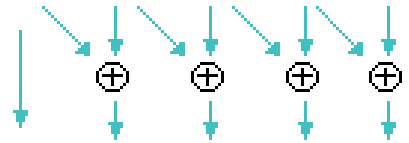
相同为0

相异为1

其中：最高位保留—— $G_{n-1}=B_{n-1}$

其他各位—— $G_i=B_{i+1}\oplus B_i$ i=0，1，2，…，n-2

例：二进制数为 1 0 1 1 0



格雷码为 1 1 1 0 1

从二进制格雷码变换为自然二进制的规则如下：

（1）两种代码的最高位相同；

（2）从高位至低位依次读取格雷码的各位码元。若某位码元为**0**，则表示与该位对应的二进制码的码元与其前一位相同；否则，表示与该位对应的二进制码的码元与其前一位不同。

某二进制格雷码为 $G_{n-1}G_{n-2}\cdots G_2G_1G_0$

其对应的自然二进制码为 $B_{n-1}B_{n-2}\cdots B_2B_1B_0$

异或运算：

相同为0

相异为1

其中：最高位保留—— $B_{n-1}=G_{n-1}$

其他各位—— $B_{i-1}=G_{i-1}\oplus B_i$ $i=1，2，\ldots，n-1$

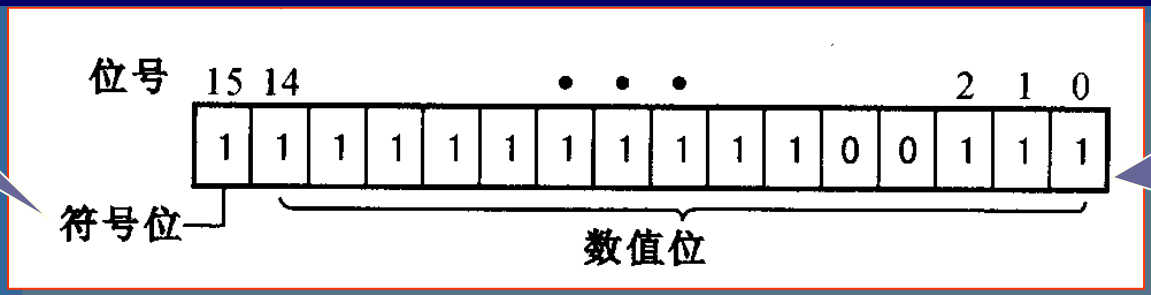
例：二进制格雷码为 1 0 1 1 0

自然二进制码为 1 1 0 1 1

带符号二进制数：

在对数进行算术运算时，必然涉及到数的符号问题。人们通常在一个数的前面用“+”号表示正数，用“－”号表示负数。而在数字系统中，符号和数值一样是用0和1来表示的，一般将数的最高位作为符号位，用0表示正，用1表示负。

0表示正，1表示负



符号和数值一起编码表示的二进制数称为**机器数**或**机器码**。

例：

$[+106]_{原}$ = **01101010**

$[-106]_{原}$ = **11101010**

原码（表示法）

原码表示方法**优点**：简单易懂；

缺点：实现加、减运算不方便。

- 当进行两数加、减运算时，要根据运算及参加运算的两个数的符号来确定是加还是减。
- 如果是做减法，则还需根据两数的大小确定被减数和减数，以及运算结果的符号。

显然，这将增加运算的复杂性，为此，引入了反码和补码的概念。

反码定义：

- 对于无符号数，反码是一种用对数值按位取反表示的二进制编码。
- 对于有符号数，反码是一种用符号位和对数值按位取反表示的二进制编码。

（有符号数的反码编码原则是：用最高位表示符号，正数用0表示，负数用1表示。正数的反码是其原码本身，负数反码的数值部分是原码的数值部分按位取反。）

补码定义：

- 对于无符号数，补码是一种用对数值按位取反并加1表示的二进制编码。
- 对于有符号数，补码是一种用符号和对数值按位取反并加1表示的二进制编码。

（有符号数的补码编码原则是：用最高位表示符号，正数用0表示，负数用1表示。正数的补码是其原码本身，负数补码的数值部分是原码的数值部分按位取反**并加1**。）

例：

十进制数	二进制原码	二进制反码	二进制补码
+26	00011010	00011010	00011010
-26	10011010	11100101	11100110

带符号二进制数的表示方法

		数值位		
	符号位	原码表示法	反码表示法	补码表示法
正数	0	绝对值的原码	绝对值的原码	绝对值的原码
负数	1	绝对值的原码	绝对值的反码	绝对值的补码

4位带符号位的原反补码

十进制	二进制码		
	原码	反码	补码
+8	—	—	—
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	1000	1111	—
−1	1001	1110	1111
−2	1010	1101	1110
−3	1011	1100	1101
−4	1100	1011	1100
−5	1101	1010	1011
−6	1110	1001	1010
−7	1111	1000	1001
−8	—	—	1000

用反码进行加法/减运算

用反码实现加/减运算的步骤：（1）将A与B均表示成反码形式；

（2）两个反码相加，且符号位也参与运算；

（3）若符号位有进位，符号位的进位需加到和数的最低位上（称为循环进位）。

例 试用反码运算求 36+22 和 −22−36，设字长为 8 位。

$[+36]_{反}=00100100$　　$[+22]_{反}=00010110$

$[-36]_{原}=10100100$　　$[-22]_{原}=10010110$

$[-36]_{反}=11011011$　　$[-22]_{反}=11101001$

$[36]_{反}+[22]_{反}$为

$$\begin{array}{r} 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0 \\ +\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0 \end{array}$$

即

$$[+36+22]_{反}=[+36]_{反}+[+22]_{反}=00111010$$

因此

而$[-36]_{反}+[-22]_{反}$为

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1 \\ \hline \boxed{1}\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0 \\ +\ \qquad\qquad\qquad\rightarrow\ 1 \\ \hline 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1 \end{array}$$

循环进位

即

$$[-22-36]_{反}=[-36]_{反}+[-22]_{反}=11000101$$

$$[-36-22]_{原}=10111010$$

故

$$-22-36=-58$$

用反码表示有符号数特点：

- **优点：**符号位直接参与运算。
- **缺点：**如果在运算时符号位有进位，需要将该进位加到结果的最低位上去，也就是说要做两次加法运算才能得到正确结果，增加了运算时间。并且，反码表示法仍然没有解决0的表示不惟一的问题。

用补码进行加法/减运算

用补码实现加法运算的步骤为：

（1） 将X与Y均表示成补码形式；

（2） 两个补码相加，且符号位也参与运算；

（3） 若符号位有进位，则自动丢失，所得结果为X+Y的补码。

例　试用补码运算求 36－22 和 22－36，设字长为 8 位。

$[+36]_{补}=00100100$　　$[+22]_{补}=00010110$

$[-36]_{原}=10100100$　　$[-22]_{原}=10010110$

$[-36]_{补}=11011100$　　$[-22]_{补}=11101010$

$[36]_{补}+[-22]_{补}$为

$$\begin{array}{r} 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0 \\ +\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ \hline \boxed{1}\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \end{array}$$

（自动丢失）

即

$$[36-22]_{补}=[36]_{补}+[-22]_{补}=00001110$$

因此

$$36-22=14$$

而$[22]_{补}+[-36]_{补}$为

$$\begin{array}{r} 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\ +\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\ \hline 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \end{array}$$

即

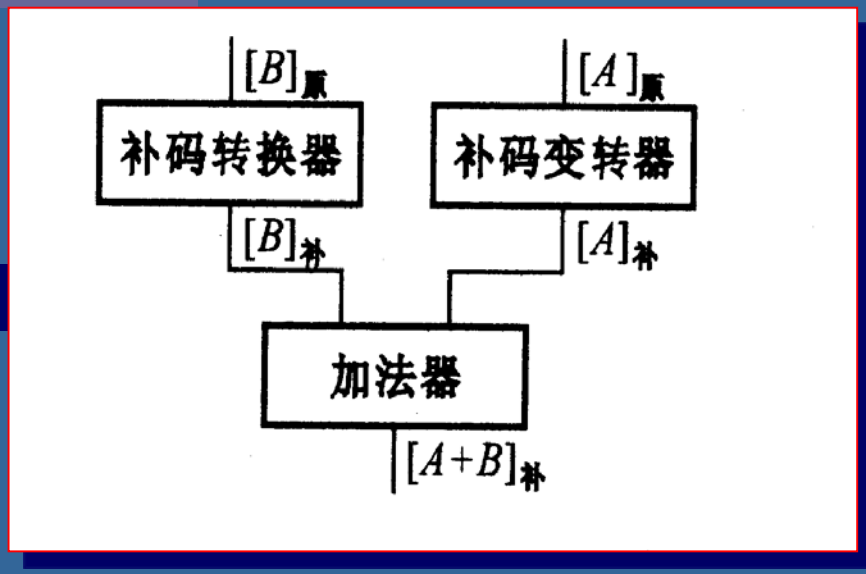
$$[22-36]_{补}=[36]_{补}+[-22]_{补}=11110010$$

$$[22-36]_{原}=10001110$$

故

$$22-36=-14$$

补码加法器框图



用补码来表示有符号数的优点：

- 符号位直接参与运算，两数补码的和直接等于两数和的补码，即$(A)_{补}$十$(B)_{补}=(A+B)_{补}$。
- 补码表示法对0的表示是唯一的，有利于简化系统。

四位带符号数的原码、反码和补码

十进制	二进制码		
	原码	反码	补码
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	—
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	—	—	1000

n位二进制原码、反码、补码表示的数的范围是：

- 原码： $-(2^{n-1}-1) \sim +(2^{n-1}-1)$
- 反码： $-(2^{n-1}-1) \sim +(2^{n-1}-1)$
- 补码： $-2^{n-1} \sim +(2^{n-1}-1)$ （不含－0）

特殊！该值仅是人为定义，可参与实际运算。但不能通过数值位减去1，然后求反而得到正确的原码。原因是已经超过了四位原码可表示的范围。

用补码进行同符号数加法运算时的溢出问题

（1）结果正确吗？
（2）如何判断？

+5	0 1 0 1
+ +2	0 0 1 0
+7	0 0 1 1 1

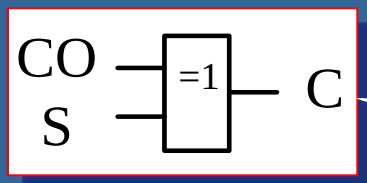
n位二进制原码、反码、补码表示的数的范围是：

- 原码：　$-(2^{n-1}-1) \sim +(2^{n-1}-1)$
- 反码：　$-(2^{n-1}-1) \sim +(2^{n-1}-1)$
- 补码：　$-2^{n-1} \sim +(2^{n-1}-1)$　（不含－0）

?

−7	1 0 0 1
+ −1	1 1 1 1
−8	1 1 0 0 0

思考题： 不同符号补码加法运算时是否存在溢出问题？



判别电路