

计算机导论

第二章 计算机中的数据



- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

- 进制也就是进位制，是人们规定的一种进位方法。对于任何一种进制—— X 进制，就表示某一位置上的数运算时是逢 X 进一位。

- **十**进制计数制 (0-9, 逢10进1; 后缀D, 常省略后缀D)

123D, -78D, **3**.141D, -1.414

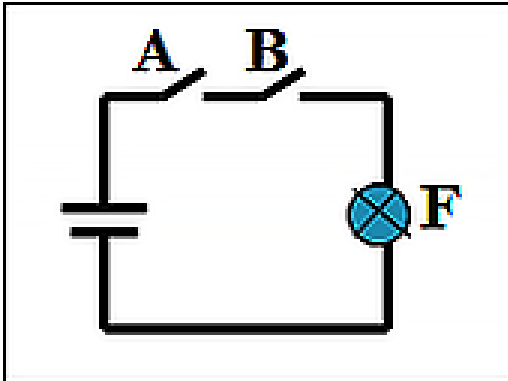
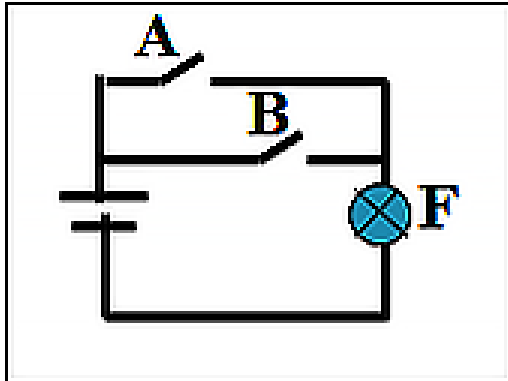
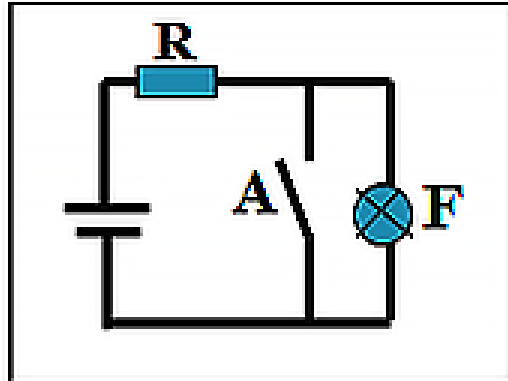
↓
一个十进制位

- **二**进制计数制 (0,1; 逢2进1; 后缀B)

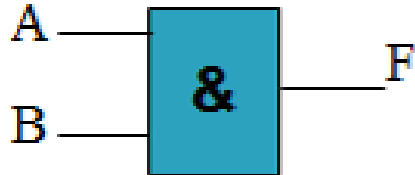
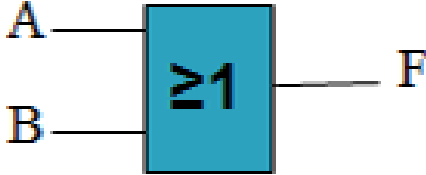
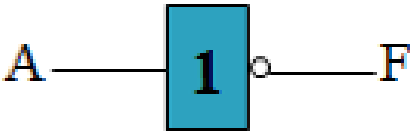
1011B, -110B, **1**01.11011B, -101.11011B

↓
一个二进制位

基本逻辑运算

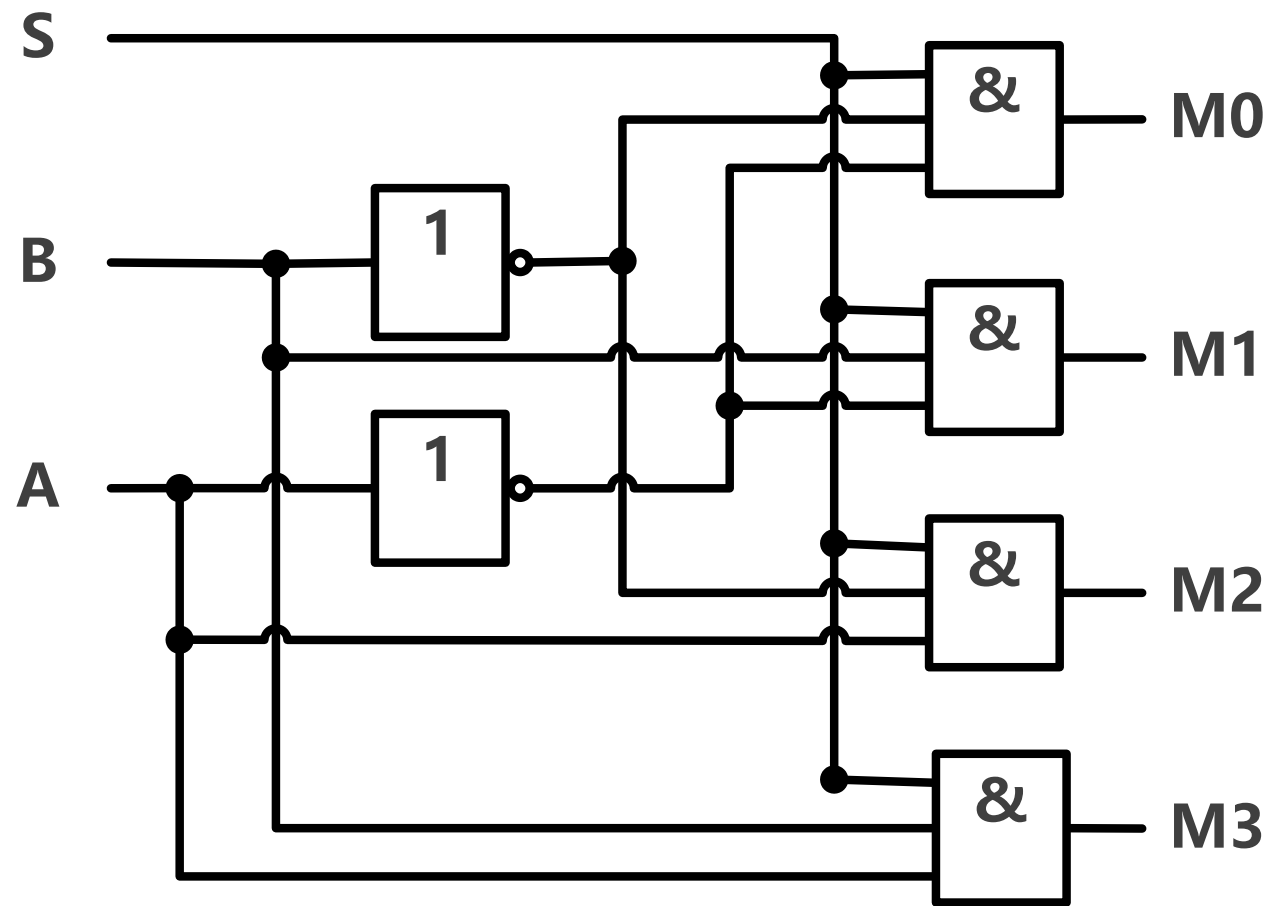
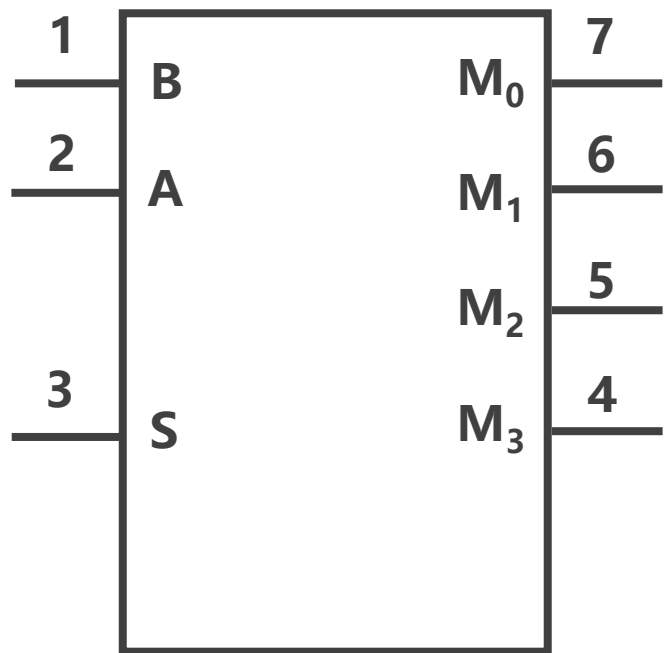
	“与” 运算(逻辑乘) <i>Logic Multiplication</i>	“或” 运算(逻辑加) <i>Logic Addition</i>	“非” 运算(逻辑非) <i>Logic Negation</i>																																				
示意 电路																																							
真值 表	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0
A	B	F																																					
0	0	0																																					
0	1	0																																					
1	0	0																																					
1	1	1																																					
A	B	F																																					
0	0	0																																					
0	1	1																																					
1	0	1																																					
1	1	1																																					
A	F																																						
0	1																																						
1	0																																						

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

	“与” 运算(逻辑乘) <i>Logic Multiplication</i>	“或” 运算(逻辑加) <i>Logic Addition</i>	“非” 运算(逻辑非) <i>Logic Negation</i>
代数式	$F = A \times B = A \cdot B$	$F = A + B$	$F = \overline{A}$
逻辑符号			

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

二四译码器的实现原理



- 容易理解：1根地址线可以产生2 (2^1) 个输出，即可以区分2个内存块， 3根地址线可以产生8 (2^3) 个输出，即可以区分8个内存块， 4根地址线可以产生16 (2^4) 个输出，即可以区分16个内存块，
- 一般的， n根地址线可以产生 2^n 个输出，即可以区分 2^n 个内存块。
- 计算机换算规则： $1\text{K}=2^{10}$, $1\text{M}=2^{10} \text{ K}$, $1\text{G}=2^{10}\text{M}$

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

- 八进制计数制 (0-7; 逢8进1; 后缀Q、O)

-123Q, 777Q, 123.456Q, -61.456Q

↓
一个八进制位

- 十六进制计数制

(0-9, A(a), B(b), C(c), D(d), E(e), F(f); 逢16进1; 后缀H)

123H, AFH, BC9.4DH, -AD.CH

一个十六进制位

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

- R进制转换成十进制

- 按权展开求和

R进制数M的整数部分有n位，小数部分有m位，转换成十进制数D。

$$\begin{aligned} D &= \pm \sum_{i=-m}^{n-1} (M_i \times R^i) \\ &= \pm (M_{-m} \times R^{-m} + \dots + M_{n-1} \times R^{n-1}) \end{aligned}$$

其中 M_i 代表第i位的数字，R为**基数**， R^i 为第i位的**权**。

- 二进制转换成十进制

$$\begin{aligned}(1101.11)_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \\ &= 8 + 4 + 0 + 1 + 0.5 + 0.25 \\ &= 13.75\end{aligned}$$

- 八进制转换成十进制

$$\begin{aligned}(173.3)_8 &= 1 \times 8^2 + 7 \times 8^1 + 3 \times 8^0 + 3 \times 8^{-1} \\ &= 64 + 56 + 3 + 0.375 \\ &= 123.375\end{aligned}$$

- 十六进制转换成十进制

$$\begin{aligned}(4C.6)_{16} &= 4 \times 16^1 + 12 \times 16^0 + 6 \times 16^{-1} \\ &= 64 + 12 + 0.375 \\ &= 76.375\end{aligned}$$

- 练习

$$10111.101B = 23.625D$$

$$7035Q = 3613D$$


$$-8FD.C H = -2301.75D$$

- 十进制转换成R进制

- 对于整数，通常采用“**除R取余**”的方法，即用R不断地去除要转换的整数，直至**商等于0**为止，然后将所得余数**从后向前**顺序写出，即为转换成的R进制数。

$$\begin{array}{r} 2 \overline{) 43} \\ \underline{2} \\ 2 \overline{) 10} \\ \underline{2} \\ 2 \overline{) 50} \\ \underline{2} \\ 2 \overline{) 20} \\ \underline{2} \\ 2 \overline{) 10} \\ \underline{2} \\ 0 \end{array} \begin{array}{l} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{array}$$

(余数)



$$43D = 101011B$$

$$\begin{array}{r} 8 \overline{) 43} \\ \underline{8} \\ 0 \end{array} \begin{array}{l} 3 \\ 5 \end{array}$$

(余数)

$$= 53Q$$

$$\begin{array}{r} 16 \overline{) 43} \\ \underline{16} \\ 0 \end{array} \begin{array}{l} 3 \\ B \end{array}$$

(余数)

$$= 2BH$$

- 十进制转换成R进制

- 对于小数，通常采用 “**乘R取整**” 的方法，即用R不断地乘以要转换的小数，直至小数部分等于0或满足所要求的精度为止，将每次得到的乘积的整数部分，从前向后顺序写出，即为转换成的R进制数。

$$\begin{array}{r} 0.8125 \\ * \quad 2 \\ \hline 1.6250 \cdots \cdots \text{整数部分为1 (最高位)} \\ 0.625 \\ * \quad 2 \\ \hline 1.250 \cdots \cdots \text{整数部分为1} \\ 0.25 \\ * \quad 2 \\ \hline 0.50 \cdots \cdots \text{整数部分为0} \\ 0.5 \\ * \quad 2 \\ \hline 1.0 \cdots \cdots \text{整数部分为1 (最低位)} \end{array}$$

$$0.8125D = 0.1101B$$

- 二进制数据转换成八进制数据

- 将二进制数据从小数点开始，分别向前向后3位分成一组，不足3位补0，然后写出对应的八进制即可。

$$(101011.101)_2 = \frac{101}{5} \frac{011}{3} . \frac{101}{5} = (53.5)_8$$

- 八进制数据转换成二进制数据

- 将每位八进制数据写出对应的3位二进制数即可。

$$(173.3)_8 = \frac{1}{001} \frac{7}{111} \frac{3}{011} . \frac{3}{011} = (1111011.011)_2$$

- 二进制数据转换成十六进制数据
 - 将二进制数据从小数点开始，分别向前向后4位分成一组，不足4位补0，然后写出对应的十六进制即可。

$$(101011.101)_2 = \frac{0010}{2} \frac{1011}{B} . \frac{1010}{A} = (2B.A)_{16}$$

- 十六进制数据转换成二进制数据
 - 将每位十六进制数据写出对应的4位二进制数即可。

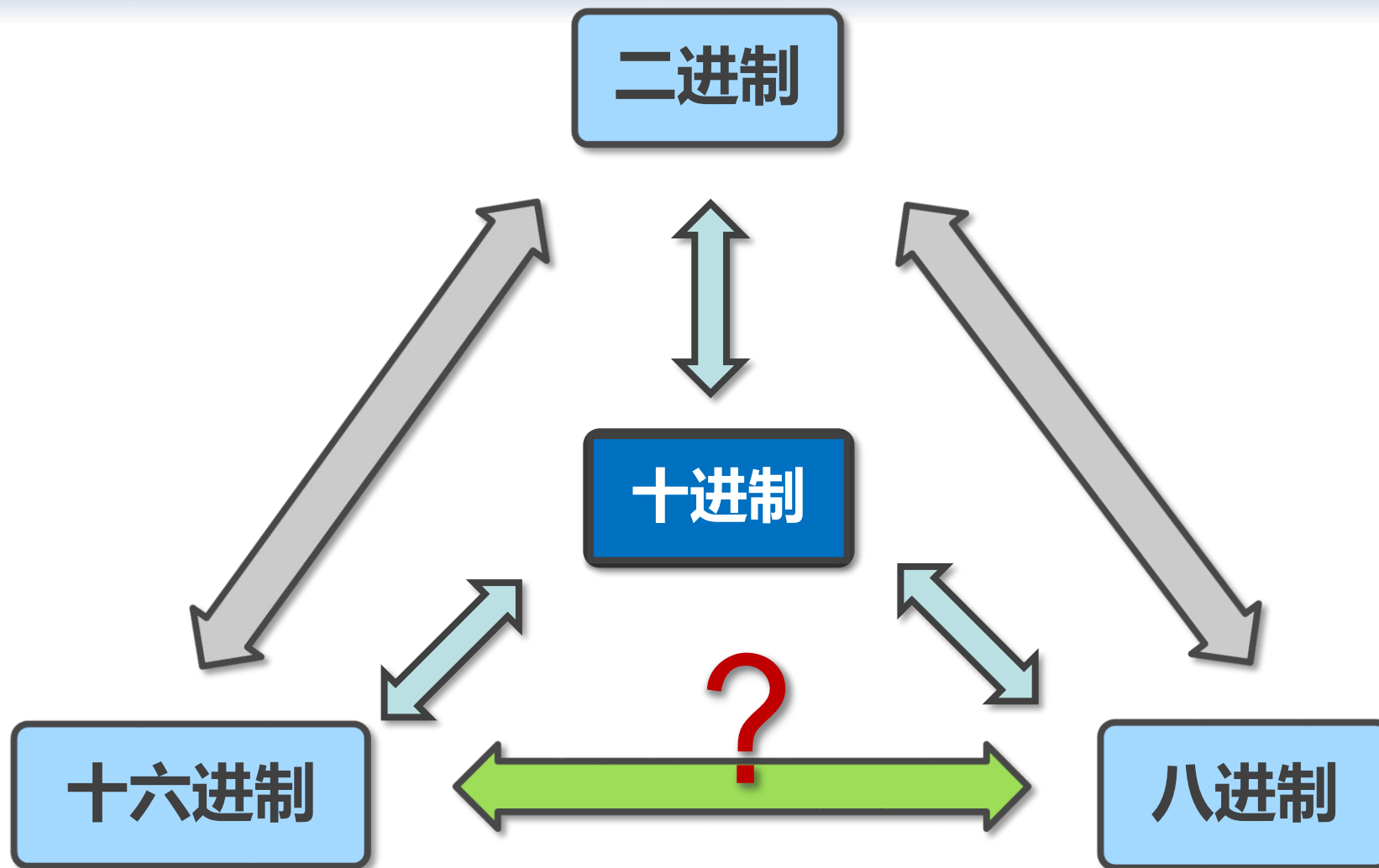
$$(173.3)_H = \frac{1}{0001} \frac{7}{0111} \frac{3}{0011} . \frac{3}{0011} = (000101110011.0011)_B$$

进制对照表

十进制	二进制	八进制	十六进制
0	00000000	0	0
1	00000001	1	1
2	00000010	2	2
3	00000011	3	3
4	00000100	4	4
5	00000101	5	5
6	00000110	6	6
7	00000111	7	7
8	00001000	10	8

十进制	二进制	八进制	十六进制
9	00001001	11	9
10	00001010	12	A
11	00001011	13	B
12	00001100	14	C
13	00001101	15	D
14	00001110	16	E
15	00001111	17	F
16	00010000	20	10
17	00010001	21	11

数制转换



- 十六进制转八进制

$$(4E2D)_H = \frac{4}{0100} \frac{E}{1110} \frac{2}{0010} \frac{D}{1101} = (0100111000101101)_B$$

$$(0100111000101101)_B = \frac{100}{4} \frac{111}{7} \frac{000}{0} \frac{101}{5} \frac{101}{5} \\ = (47055)_Q$$

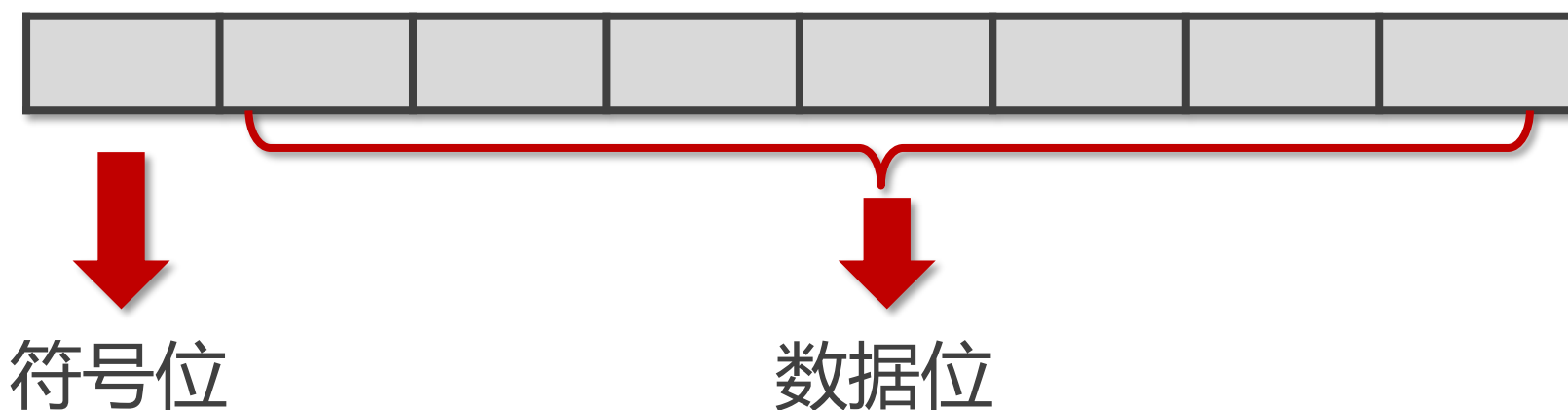
- 八进制转十六进制

$$(4327)_Q = \frac{4}{100} \frac{3}{011} \frac{2}{010} \frac{7}{111} = (100011010111)_B$$

$$(100011010111)_B = \frac{1000}{8} \frac{1101}{D} \frac{0111}{7} = (8D7)_H$$

- 二进制数据的基本运算
- 二进制数据基本运算的硬件实现
- 二进制数据运算的应用
- 其他进制数据
- 不同进制数据间的转换
- 有符号数据在计算机中的存储

对于数的符号 "+" 或 "-", 计算机是无法识别的, 因此需要把数的符号数码化。通常, 约定二进制数的最高位为符号位, "0" 表示正号, "1" 表示负号。这种在计算机中使用的表示数的形式称为**机器数**。规定: 以后没有特别指明的话, 规定1个字节, 即8位来存储整数。



- 例如：

$N1 = 00001001$ 表示带符号数 $+9$

$N2 = 11111001$ 根据不同的机器数表示不同的值，如：

原码时：表示带符号数 -121

反码时：则表示 -6

补码时：则表示 -7

- 定义：符号占1位（最高位），0表示正、1表示负；数值部分按二进制书写（占剩下的位置）。
- 例如，+11
原码： 00001011
- 例如，-11
原码： 10001011

- 根据定义，0有两种表示方法：+0，-0。

00000000

10000000

- 1个字节能表示 -127 到 +127 这 255 个数

最大数：01111111

最小数：11111111

- 定义：如果是正数，与原码相同；如果是负数，符号位不变，数据位取反。
- 例如，+11
反码： 00001011
- 例如，-11
反码： 11110100

- 根据定义，0有两种表示方法：+0，-0。

00000000

11111111

- 1个字节能表示 -127 到 +127 这 255 个数

最大数：01111111

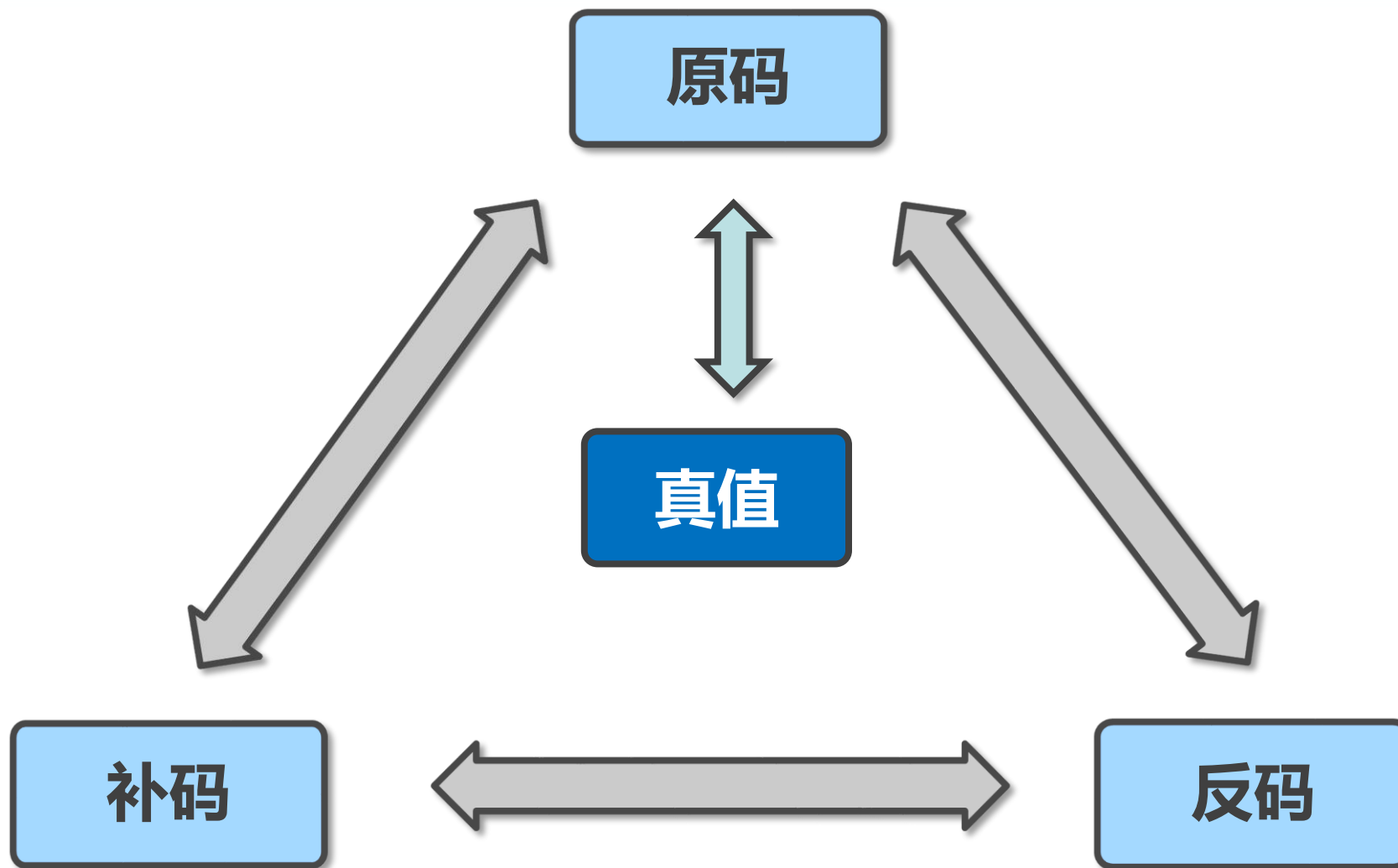
最小数：10000000

- 定义：如果是正数，与原码相同；如果是负数，符号位不变，数据位取反，末位加1。
- 例如，+11
补码： 00001011
- 例如，-11
补码： 11110101

- 根据定义，0只有一种表示方法：00000000
- 特殊值 -128 表示方法：10000000
- 1个字节能表示 -128 到 +127 这 256 个数

最大数：01111111

最小数：10000000



- 写出下列真值的原码、反码、补码。

-33

-56

40

Questions?

