绪论2：数据和程序的存储与表示

内　容

（1）计算机的数字系统有哪些？
（2）如何进行不同进制数之间的转换？
（3）计算机中如何表示正、负数？
（4）计算机中如何表示整数和小数？

1、计算机的数字系统

■基数为R的进制数：逢R进1

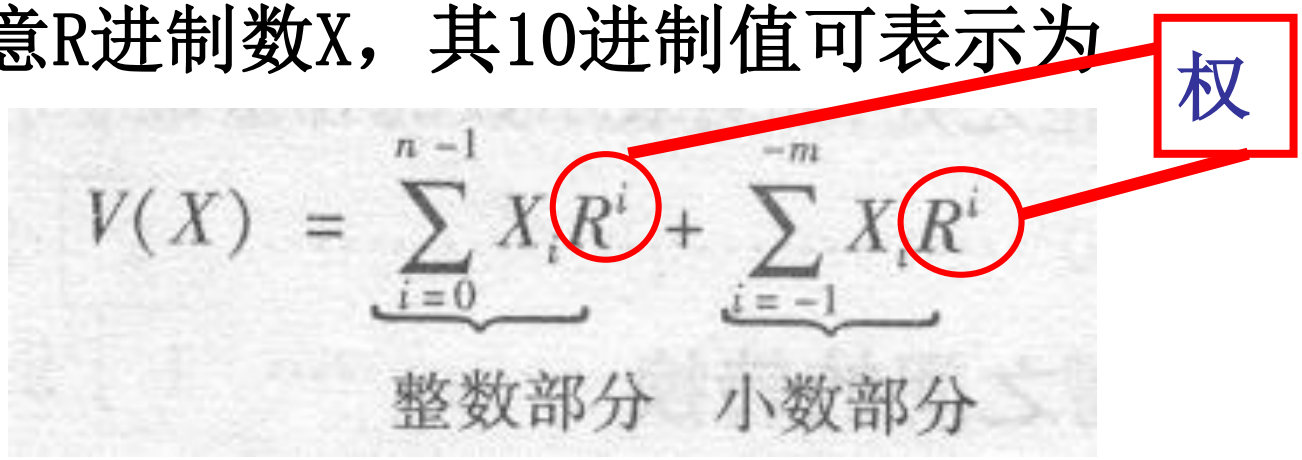
进制	基数	进位原则	基本符号
2进制	2	逢2进1	0 1
10进制	10	逢10进1	0 1 2 3 4 5 6 7 8 9
16进制	16	逢16进1	0 1 2 3 4 5 6 7 8 9 A B C D E F

■所有的计算机都采用2进制的数字系统

■优点：易于实现、运算简单、可靠性高、通用性强

2、不同进制数之间的转换

■任意R进制数X，其10进制值可表示为



■10进制数

8844.43=8000 +800 +40 +4 +0.4 +0.03

$=8\times10^3+8\times10^2+4\times10^1+4\times10^0+4\times10^{-1}+3\times10^{-2}$

■常用的转换形式有：

（1）2进制、16进制←→10进制

（2）2进制←→16进制

2.A、2/16进制数→十进制的转换

$$V(X) = \underbrace{\sum_{i=0}^{n-1} X_i R^i}_{\text{整数部分}} + \underbrace{\sum_{i=-1}^{-m} X_i R^i}_{\text{小数部分}}$$

■各位数与权相成，积相加

$(10001001.11)_2=1x2^7+1x2^3+1x2^0+1x2^{-1}+1x2^{-2}$

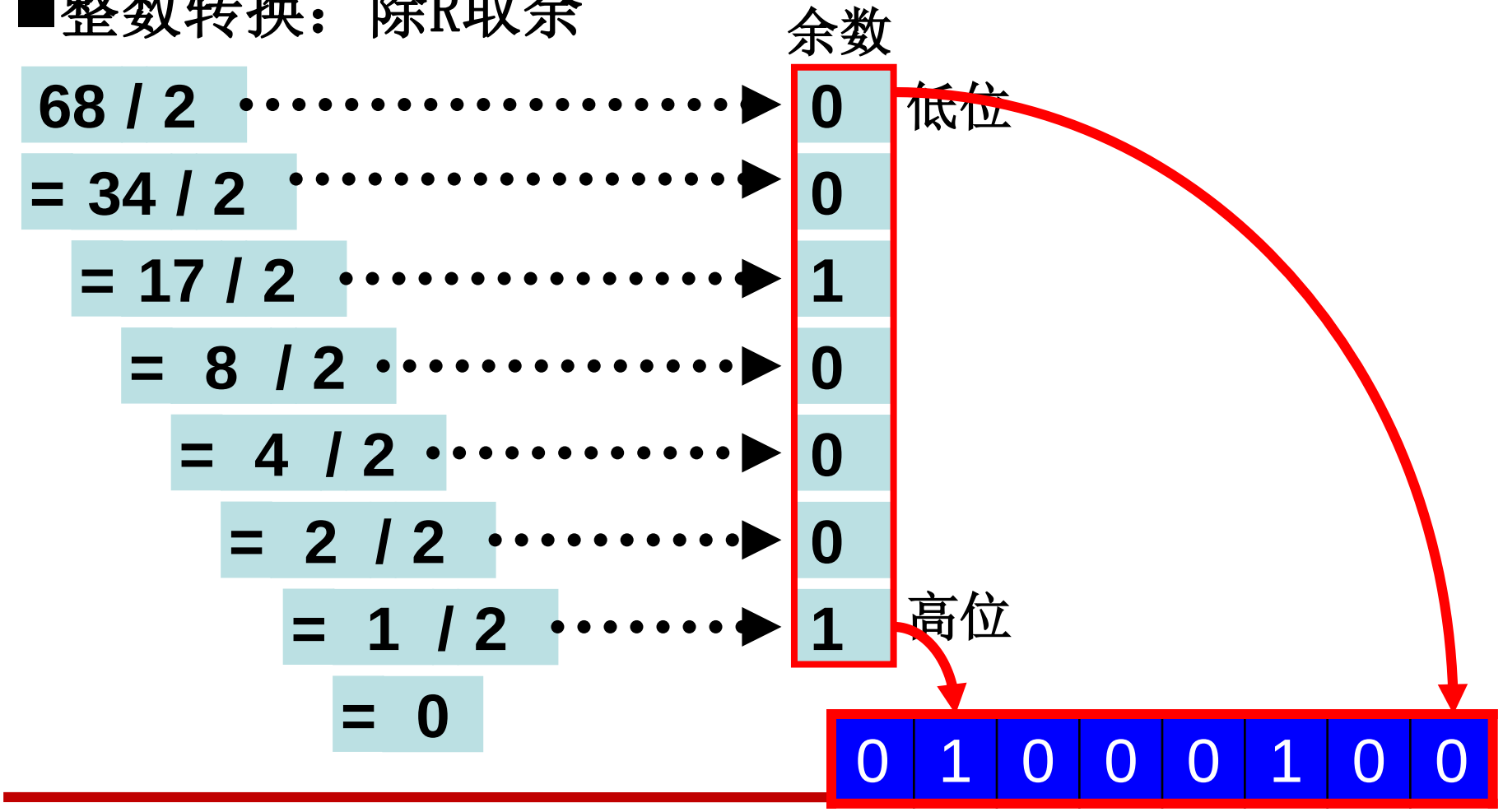
$=(137.75)_{10}$

$(0.2A)_{16}=2x16^{-1}+10x16^{-2}$

$=(0.1640625)_{10}$

2.B、十进制数→R进制的转换

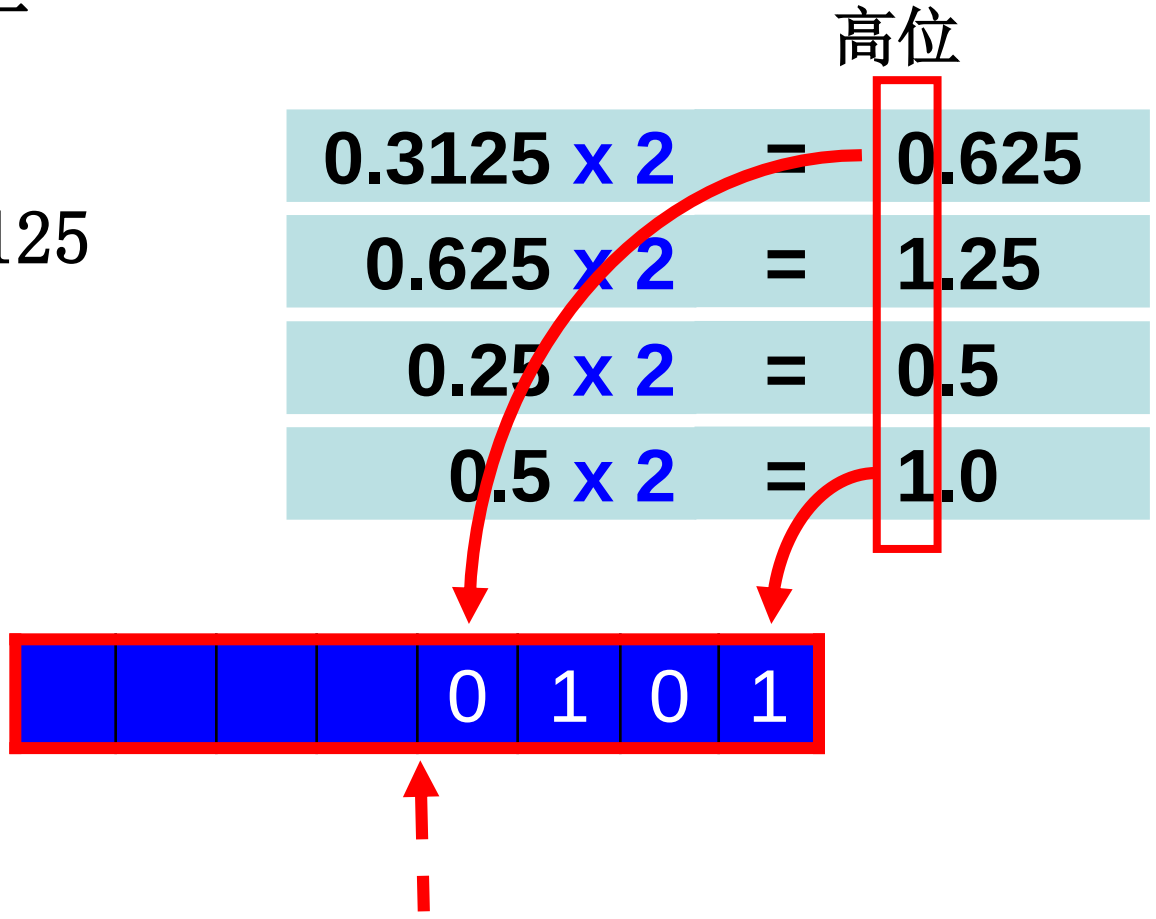
■整数转换：除R取余



2.C、十进制数→R进制的转换

■小数转换：乘R取整

例：10进制整0.3125



2.D、16进制←→2进制

- **1:4**（每位16进制数相当于4位2进制数）

$(1011010.10)_2=(0101\ 1010\ .1000)_2=(5A.8)_{16}$

$(F7)_{16}=(1111\ 0111)_2=(11110111)_2$

练习：2进制数到10进制数

2进制数	10进制值	
0000 0001	$=2^0$	=1
0000 0010	$=2^1$	=2
0000 0100	$=2^2$	=4
0000 1000	$=2^3$	=8
0001 0000	$=2^4$	=16
0010 0000	$=2^5$	=32
0100 0000	$=2^6$	=64
1000 0000	$=2^7$	=128
0100 0010	$=2^6+2^1$	=66

练习：2进制数到16进制数

2进制	16进制
0000 0001	=0x01
0000 0010	=0x02
0000 0011	=0x03
0000 0100	=0x04
0000 0101	=0x05
0000 0110	=0x06
0000 0111	=0x07
0000 1000	=0x08
0000 1001	=0x09

2进制	16进制
0000 1010	=0x0A
0000 1011	=0x0B
0000 1100	=0x0C
0000 1101	=0x0D
0000 1110	=0x0E
0000 1111	=0x0F

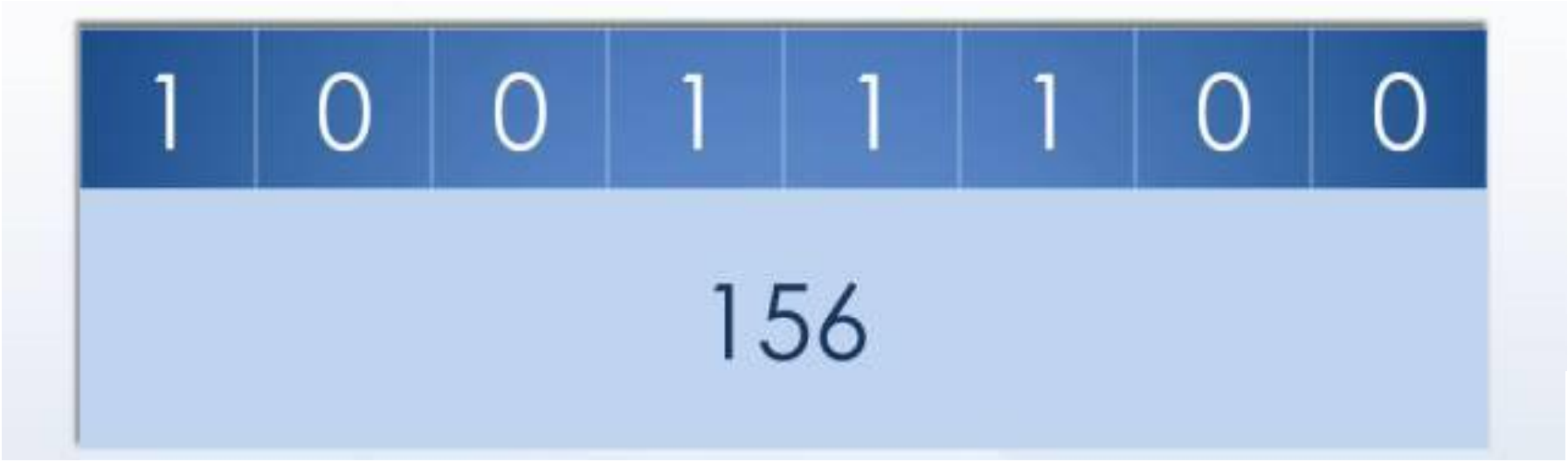
练习：2进制数到16进制数

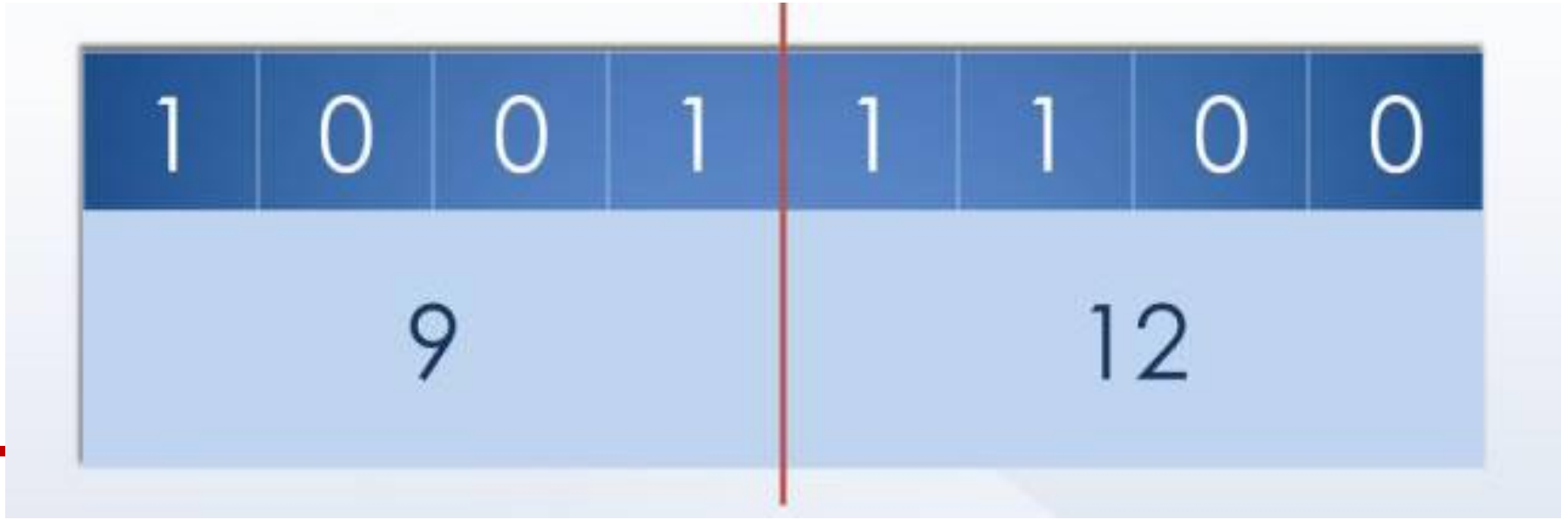
2进制	16进制
0001 1000	=0x18
0011 1100	=0x3C
1110 0000	=0xE0
0101 0100	=0x54
0111 0111	=0x77
1001 1010	=0x9A
1011 0010	=0xB2
0110 0001	=0x61
1111 0100	=0xF4

2进制	16进制
1000 0101	=0x85
1010 1010	=0xAA
0101 1011	=0x5B
0110 1101	=0x6D
1101 0000	=0xD0
1110 0100	=0xE4
0001 0111	=0x17
0011 1000	=0x38
1001 1001	=0x99

- 3.A 数据的不同理解

- 数据在存储器中的表示





- 数据在存储器中的表示

- 3.B -1如何表示

$$\begin{array}{r} 0000 \\ -\ 0001 \\ \hline ???? \end{array}$$

$$\begin{array}{r} 10000 \\ -\ 0001 \\ \hline ???? \end{array}$$

- -1的表示

$$\begin{array}{r} 0000 \\ -\quad 0001 \\ \hline 1111 \end{array}$$

- 3.C 负数的一般表达

二进制补码	表示的数字
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

特点：

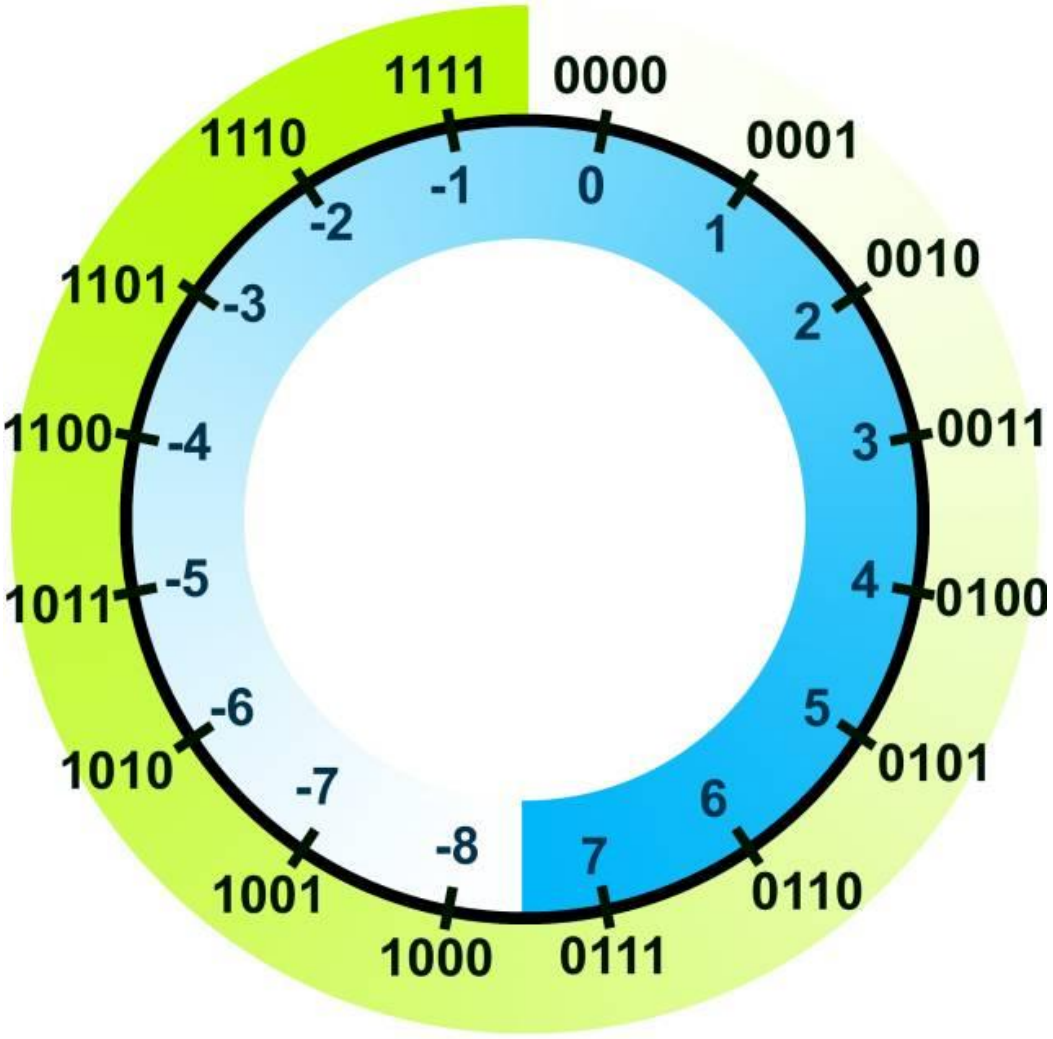
1. 以0开头的为非负数，以1开头的为负数。

2. 从0000到0111是递增的。从1000 ~ 1111是递增的。

3. 二进制数据0111表达的是最大整数7，0111只要加上1，就会变成1000，而1000表达的是最小数字 -8 。

4. 0和-1同样只有一步之遥，但是表达他们的二进制数字却是最小的0000和最大的1111。

- 补码环


0000
0
0001
1
0010
2
0011
3
0100
4
0101
5
0110
6
0111
7
1000
-8
1001
-7
1010
-6
1011
-5
1100
-4
1101
-3
1110
-2
1111
-1

3.D 补码的计算

补码的一大优点就是计算机在做加法或减法运算时，不需要考虑补码的存在而直接运算，结果却依然正确。我们可以利用竖式计算 -2+5，以及2-5（-2、2、5在四位补码系统中分别为1110、0010和0101）：

$$\begin{array}{rr} & 1110 \\ + & 0101 \\ \hline (1) & 0011 \end{array} \qquad \begin{array}{rr} & (1)0010 \\ - & 0101 \\ \hline & 1101 \end{array}$$

从左式中我们可以看到，1110（-2）和0101（5）求和，得到10011，去掉首位进位1，留下0011，即为答案3。右式中，0010（2）强行借位后减去0101（5），得到1101，在表1 5中我们看到，1101即为答案-3。

3.E 补码的计算中的溢出

这种计算其实不总是正确的，如果 0110+0110，也就是 6+6，会得到1100。而1100在补码表示的是-4，却不是我们希望的12。究其原因是12已经超出了四位补码系统所能表达的最大数字7（0111）。我们把这种现象叫做“溢出”。“溢出”的问题我们之前在上一讲中已经介绍了，防止溢出通常是采用更多的数据位，有时候调整一下计算次序也可以避免溢出。

3.F 扩展到N位的补码

我们可以将四位补码推广到N位补码系统。在N位补码系统中，有以下规律：

1. 各位全部为0，总是表示0；

2. 各位全部为1，总是表示 -1；

3.首位为0，后面全部为1，表示的是最大整数：2的N-1次方减去1；

4.首位为1，后面全部为0，表示的是最小整数：负的2的N-1次方 。

	数值	8位
正数部分	127	0111 1111
	126	0111 1110
	125	0111 1101
	…	…
	2	0000 0010
	1	0000 0001
	0	0000 0000
负数部分	−1	1111 1111
	−2	1111 1110
	−3	1111 1101
	…	…
	−126	1000 0010
	−127	1000 0001
	−128	1000 0000

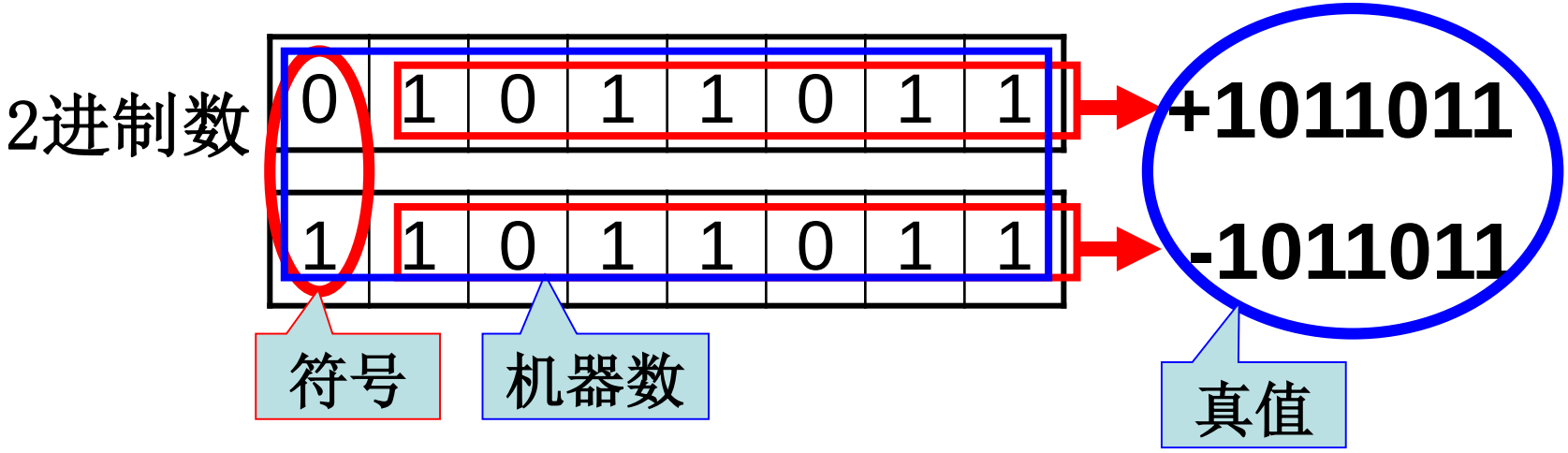
	数值	16位
正数部分	32767	0111 1111 1111 1111
	32766	0111 1111 1111 1110
	32765	0111 1111 1111 1101
	…	…
	2	0000 0000 0000 0010
	1	0000 0000 0000 0001
	0	0000 0000 0000 0000
负数部分	−1	1111 1111 1111 1111
	−2	1111 1111 1111 1110
	−3	1111 1111 1111 1101
	…	…
	−32766	1000 0000 0000 0010
	−32767	1000 0000 0000 0001
	−32768	1000 0000 0000 0000

	数值	32位
正数部分	$2^{31}-1$	0111 … 1111
	$2^{31}-2$	0111 … 1110
	$2^{31}-3$	0111 … 1101
	…	…
	2	0000 … 0010
	1	0000 … 0001
	0	0000 … 0000
负数部分	-1	1111 … 1111
	-2	1111 … 1110
	-3	1111 … 1101
	…	…
	$-2^{31}+2$	1000 … 0010
	$-2^{31}+1$	1000 … 0001
	-2^{31}	1000 … 0000

	数值	64位
正数部分	$2^{63}-1$	0111 … 1111
	$2^{63}-2$	0111 … 1110
	$2^{63}-3$	0111 … 1101
	…	…
	2	0000 … 0010
	1	0000 … 0001
	0	0000 … 0000
负数部分	-1	1111 … 1111
	-2	1111 … 1110
	-3	1111 … 1101
	…	…
	$-2^{63}+2$	1000 … 0010
	$-2^{63}+1$	1000 … 0001
	-2^{63}	1000 … 0000

3.C、正数和负数的表示

- 在计算机中，各种信息都是以二进制编码形式存储
- 一般用最高位作为符号位，0表示正，1表示负
- 计算机中一般用原码表示“正数”，用“补码”表示负数
- 概念：原码、反码、补码



- “符号－绝对值”表示的编码称为原码

原码的表示规则和优缺点

- 最左1位作符号位，“符号－绝对值”表示法
- 表示规则：正数不变、负数用“1－绝对值”表示

2进制真值	原码表示的机器数
＋0101011	00101011
－0101011	10101011
＋0.1011	0.1011
－0.1011	1.1011
＋0	00000000
－0	10000000

- 优点：
  - 简单直观
  - 与真值转换方便
- 缺点：
  - 判0麻烦
  - 符号处理且很复杂

反码的表示规则

- 正数的反码与原码相同
- 负数的反码符号位与原码相同，其余取反（0变1、1变0）

2进制真值	原码表示的机器数	反码表示的机器数
＋0101011	00101011	00101011
－0101011	10101011	11010100
＋0.1011	0.1011	0.1011
－0.1011	1.1011	1.0100
＋0	00000000	00000000
－0	10000000	11111111

补码的表示规则

- 正数：原码、反码、补码相同
- 负数：补码＝反码的最后一位＋1

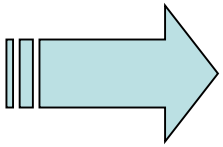
2进制真值	反码表示的机器数	补码表示的机器数
＋0101011	00101011	00101011
－0101011	11010100	11010101
＋0.1011	0.1011	0.1011
－0.1011	1.0100	1.0101
＋0	00000000	00000000
－0	11111111	00000000

7、整数的表示

- 用“补码”表示
- 正整数：原码
- 负整数：反码最后一位+1

8、浮点数的表示

3.1415926535	$0.31415926535 \times 10^{1}$
312563810.28	$0.31256381028 \times 10^{9}$
-11.2357823122	$-0.112357823122 \times 10^{2}$
0.000000000963	0.963×10^{-9}



- 整数部分. 小数部分
- 科学表示法：$N = M \times R^{E}$

尾数，N的有效数字

基数

阶码：小数点的位置

8、浮点数的表示

■float科学表示法：$N=M\times R^{E}$

符号位（s）	阶码（E）	尾数（M）
1	8	23

■float表示法范围：

-3.4×10^{38} —— -3.4×10^{-38} ,0, 3.4×10^{-38} —— 3.4×10^{38}

■double科学表示法：$N=M\times R^{E}$

符号位（s）	阶码（E）	尾数（M）
1	11	52

■double表示法范围：

-1.7×10^{308} —— -1.7×10^{-308} ,0, 1.7×10^{-308} —— 1.7×10^{308}

9、字符的表示

（1）西文字符（如：a, b, c, d, 1, 2, 3, 4, A, B, C等）

■ASCII码：用7位二进制数表示一个字符，可表示128个字符

■EBCDIC码：用8位二进制数表示一个字符，可表示256个字符

（2）汉字

■国标码（GB2312-80标准）：

应用较为广泛的是“国家标准信息交换用汉字编码”

二字节码，用二个七位二进制数编码表示一个汉字。

（3）日、韩、俄等全球其他语言字符（如：サぶアセ）

■Unicode

全球统一多字节编码字符集

编七位ASCII码表

低4位

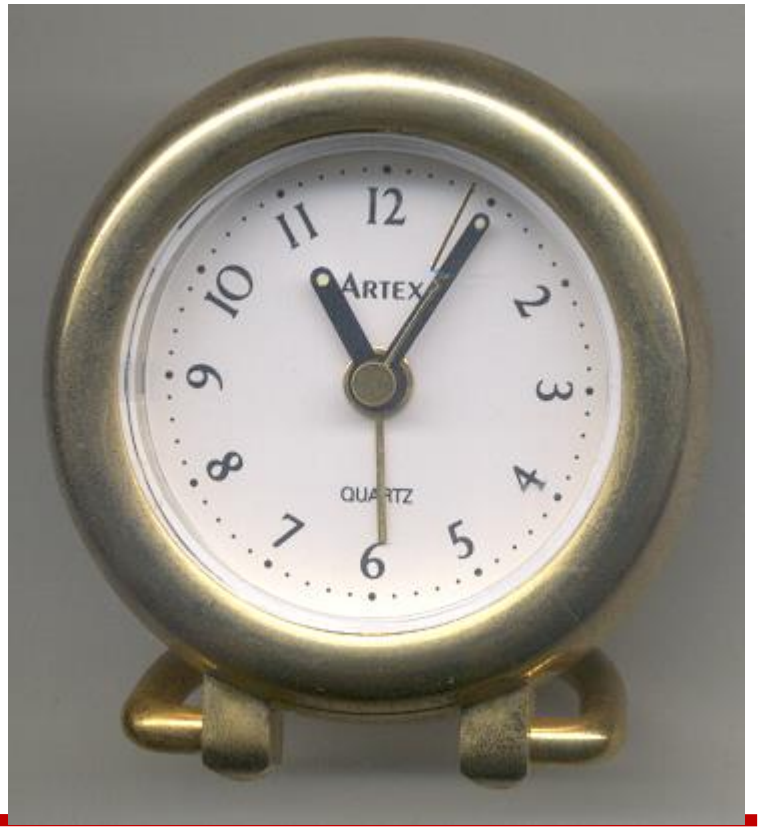
高3位

	000	001	010	011	100	101	110	111
0000	NUL	DLE		0	@	P		p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	S		6		V		v
0111	BEL	ETB		7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	。	>	N	|	n	_
1111	SI	US	/	?	O	—	o	DEL

0x31 **0x41** **0x61**

模数与补码

■模数：类似于某种计量器的容量，如时钟。
■R进制的模数=R



■模运算、取余运算（mod）：
C＋＋中用％作为模运算符

■R进制的模运算结果>=0，<=R-1
■模运算、取余运算示例：
3％12 ＝3
14％12＝2

■模运算的特点：减法统一为加法
8-2 = (8+10)%12
10是-2在模12下的补码

课后阅读

（1）《计算机高级语言》学习材料，Page 4

课后作业

（1）完成《计算机高级语言》第1次作业

谢　　谢！