计算机组成与结构
—计算机中的数据表示1

计算机科学与技术学院

本章学习要点

- 数字的表示
  - 进制编码基础
  - 整数及小数进制转换（给人看）
  - 整数及小数在计算机里的表示（给机器看）
  - 有符号数的原、反、补、移码
  - 浮点表示
    - IEEE 754
  - BCD
- 非数字的表示
  - 字符表示
    - ASCII
    - 汉字
    - Unicode
  - 校验码
    - 奇偶校验
    - CRC

进制表示基础

进制表示基础

- 人使用10进制计算，理解数量问题；优点很多，除了计算机看不懂…
- 计算机，核心是2进制驱动的，2进制是生产力输出；优点很多，除了人看不懂…
  - 主力辅助：16进制，和2进制快速转换，小小数字表示大大内容…
  - 次辅助，8进制，和2进制快速转换，其实没人用…

进制表示基础

- **几种主要的进制表示，例如，对于10进制数值2020.3**
  - **10进制，Decimal，表示为$(2020.3)_{10}$或$(2020.3)_D$**
  - **2进制，Binary，表示为$(11111100100.010011..)_2$或$(11111100100.010011..)_B$**
  - **8进制，Octal，表示为$(3744.2314..)_8$或$(3744.2314..)_O$**
  - **16进制，Hexadecimal，表示为$(7E4.4CC...)_{16}$或$(7E4.4CC...)_H$**

怎么算的！？太复杂了吧！？

莫慌，下一小节传授~

进制表示基础

任何一个r进制数N_r，可表示为多项式之和：

$$N_r = \sum_{i=-m}^{n-1} D_i \times r^i$$

式中：r表示进制，m表示小数位的位数，n表示整数位的位数，D_i为第i位上的数符，$D_i \in [0, r-1]$

进制表示基础

任何一个r进制数N_r，可表示为多项式之和：

$$N_r = \sum_{i=-m}^{n-1} D_i \times r^i$$

式中：r表示进制，m表示小数位的位数，n表示整数位的位数，D_i为第i位上的数符，$D_i \in [0, r-1]$

如何理解10进制?

逢10进1

$$N_D = \sum_{i=-m}^{n-1} D_i \times 10^i$$

$$(2020.3)_{10} = 2 \times 10^3 + 2 \times 10 + 3 \times 10^{-1}$$

进制表示基础

任何一个r进制数N_r，可表示为多项式之和：

$$N_r = \sum_{i=-m}^{n-1} D_i \times r^i$$

式中：r表示进制，m表示小数位的位数，n表示整数位的位数，D_i为第i位上的数符，$D_i \in [0, r-1]$

如何理解2进制?

逢2进1

$$N_D = \sum_{i=-m}^{n-1} B_i \times 2^i$$

$$(2020.3)_{10}$$
$$= 2^{10} + 2^9 + 2^8 + 2^7 + 2^9 + 2^5$$
$$+2^2 + 2^{-2} + 2^{-5} + 2^{-6} + \ldots$$
$$=(11111100100.010011\ldots)_B$$

进制表示基础

任何一个r进制数N_r，可表示为多项式之和：

$$N_r = \sum_{i=-m}^{n-1} D_i \times r^i$$

式中：r表示进制，m表示小数位的位数，n表示整数位的位数，D_i为第i位上的数符，$D_i \in [0, r-1]$

如何理解8进制?

逢8进1

$$N_D = \sum_{i=-m}^{n-1} O_i \times 8^i$$

$$(2020.3)_{10} = 3 \times 8^3 + 7 \times 8^2 + 4 \times 8^1 + 4 \times 8^0 + 2 \times 8^{-1} + 3 \times 8^{-2} + 1 \times 8^{-3} + \ldots = (3744.231\ldots)_O$$

进制表示基础

任何一个r进制数N_r，可表示为多项式之和：

$$N_r = \sum_{i=-m}^{n-1} D_i \times r^i$$

式中：r表示进制，m表示小数位的位数，n表示整数位的位数，D_i为第i位上的数符，$D_i \in [0, r-1]$

如何理解16进制?

逢16进1

0-9不够用+ABCDEF

$$N_D = \sum_{i=-m}^{n-1} H_i \times 16^i$$

$$(2020.3)_{10}$$
$$= 7 \times 16^2 + E(14) \times 16^1 + 4 \times 16^0$$
$$+ 4 \times 16^{-1} + C(12) \times 16^{-2}$$
$$+ C(12) \times 16^{-3} + \ldots$$
$$= (7E4.4CC \ldots)_H$$

进制表示基础

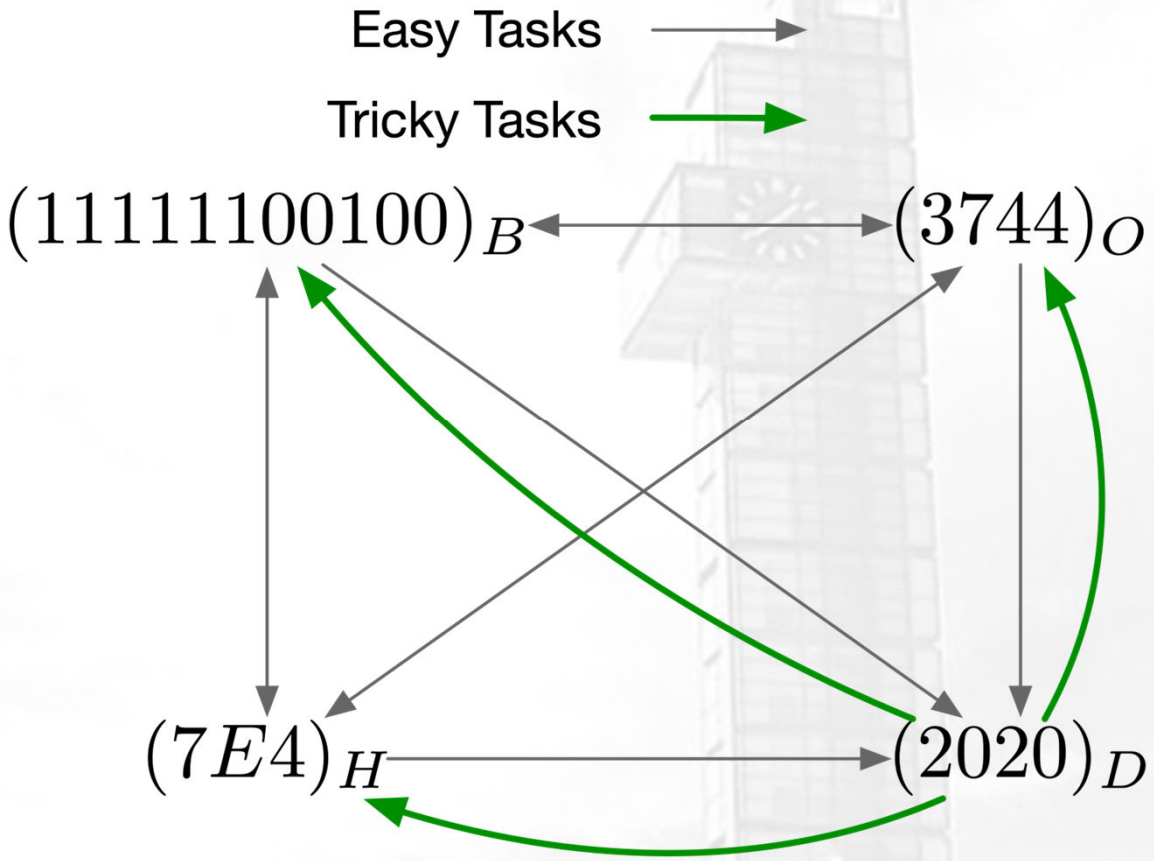
Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A

Decimal	Binary	Octal	Hexadecimal
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15

进制表示转换

进制表示转换

- 数制转换有易有难
  - X进制转10进制，easy
    - 逐位加权并相加
  - 2/8/16进制互转，easy
    - 按位合并/展开
  - 10进制转X进制，tricky
    - 整数部分
    - 小数部分



进制表示转换

X进制转10进制，方法：逐位加权并相加

【例】将二进制数$(11001.01)_2$、八进制数$(216.3)_8$、十六进制数$(7A.C)_{16}$转换成十进制数。

【解】“逐位加权，再相加”

$$(11001.01)_2 = (1\times2^4+1\times2^3+1\times2^0+1\times2^{-2})_{10} = (25.25)_{10}$$

$$(216.3)_8 = (2\times8^2+1\times8^1+6\times8^0+3\times8^{-1})_{10} = (142.375)_{10}$$

$$(7A.C)_{16} = (7\times16^1+10\times16^0+12\times16^{-1})_{10} = (122.75)_{10}$$

进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高**
- **小数部分，乘基取整，先高后低**

进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高；小数部分，乘基取整，先高后低**

【例】将十进制数730.8125转换成8进制数。

解：整数部分，迭代÷基数并取余，后面是高位

730 / 8 = 91 ——2 低位

91 / 8 = 11 ——3

11 / 8 = 1 ——3

1 / 8 = 0 ——1

0 / 8 = 0 ——0 高位

=> $(730)_{10} = (1332)_8$

进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高；小数部分，乘基取整，先高后低**

【例】将十进制数730.8125转换成8进制数。

解：整数部分，迭代÷基数并取余，后面是高位

730 / 8 = 91 ——2　　低位

91 / 8 = 11 ——3

11 / 8 = 1 ——3

1 / 8 = 0 ——1

0 / 8 = 0 ——0　　高位

=> $(730)_{10} = (1332)_8$

另一种写法

			余数
8	730	……	2
8	91	……	3
8	11	……	3
8	1	……	1
	0	……	0

进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高；小数部分，乘基取整，先高后低**

【例】将十进制数730.8125转换成8进制数。

解：小数部分，小数部分迭代×基数取整，前面是高位

0.8125 x 8 = 6.5 ——6 高位

0.5 x 8 = 4 ——4

0.0 x 8 = 0 ——0 低位

=> $(0.8125)_{10} = (0.64)_8$

=> $(730.8125)_{10} = (1322.64)_8$

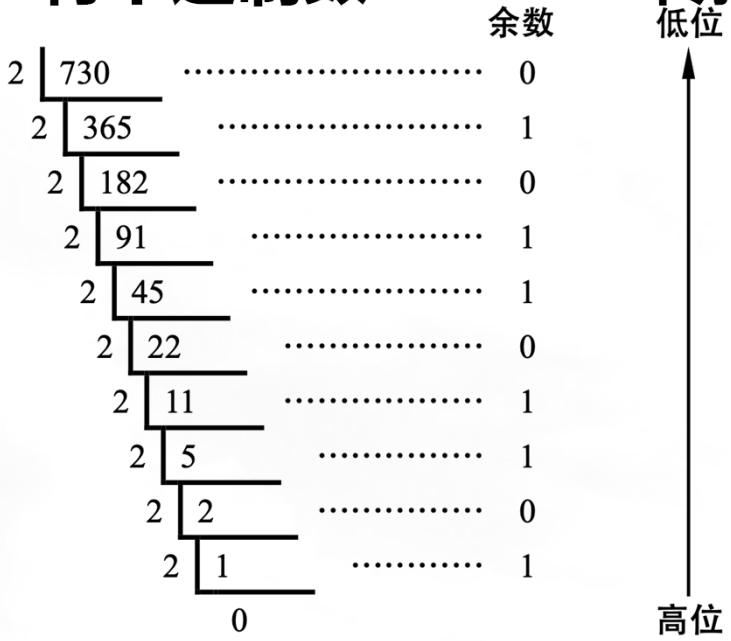
进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高；小数部分，乘基取整，先高后低**

【例】将十进制数730.8125转换成2进制数。

解：



=> $(730)_{10} = (1011011010)_2$

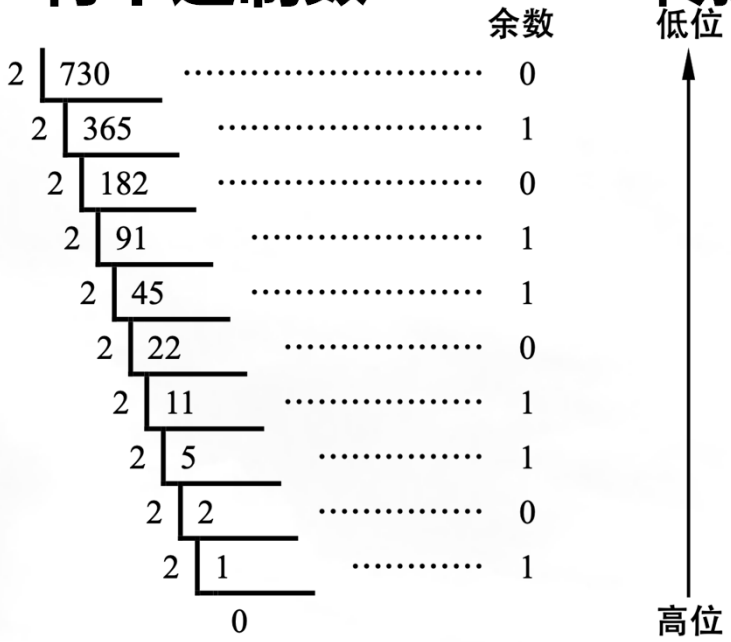
进制表示转换

10进制转X进制，方法：

- **整数部分，除基取余，先低后高；小数部分，乘基取整，先高后低**

【例】将十进制数730.8125转换成2进制数。

解：



0.8125×2＝1.625 ——1（高位）

0.625×2＝1.25 ——1

0.25×2＝0.5 ——0

0.5×2＝1.0 ——1（低位）

=> $(730)_{10} = (1011011010)_2$ $(0.8125)_{10} = (0.1101)_2$

=> $(730.8125)_{10} = (1011011010.1101)_2$

进制表示转换

2/8/16进制互转：

- **2转8/16 => 每隔3/4位合并；**
- **8/16转2 => 每个数字展开为3/4位；**
- **8/16互转 => 先转为2，再互转**

进制表示转换

例：将$(1011011010.1101)_2$ 转为8/16进制

注意：合并时，以小数点为基准向左/向右合并

二进制位置	9	8	7	6	5	4	3	2	1	0	小数点	-1	-2	-3	-4
Binary	1	0	1	1	0	1	1	0	1	0	.	1	1	0	1
Octal	1	3			3			2			.	6			4
Hexadecimal	2		D			A					.	D			

注意这一位的合并方向

注意这一位的合并方向

有符号数的原、反、补、移四码表示

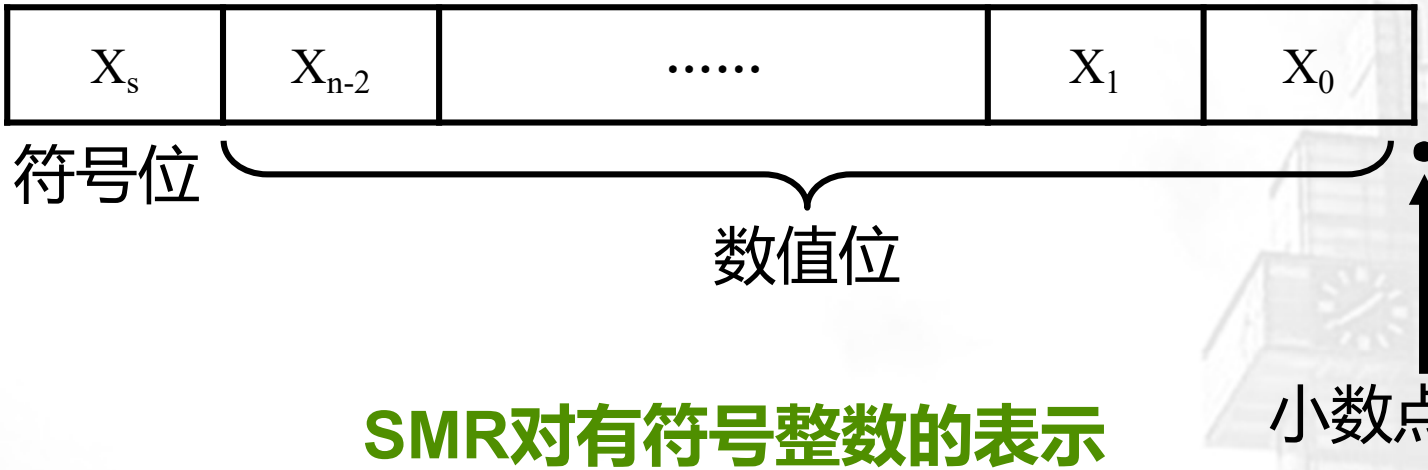
有符号数，有什么问题?

- 符号怎么表示?
  - $(-2020)_{10} = (????)_2$,

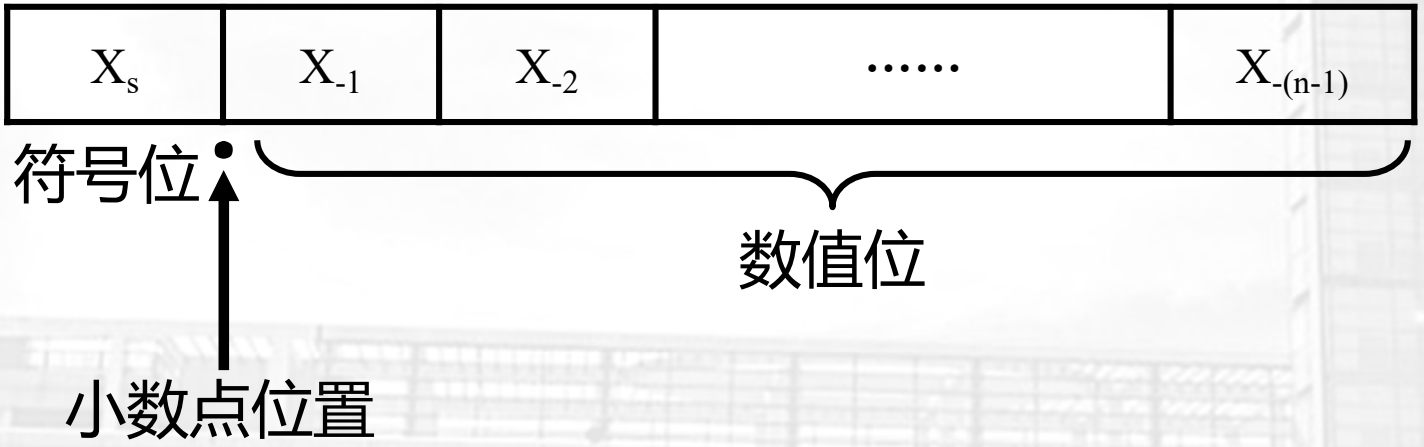
    $(2020)_{10} = (????)_2$
- 如何方便计算?
  - 加、减、乘、除

- 无符号数:
- 有符号数:
  - 原码（Signed-Magnitude Representation, SMR)
  - 反码（One's Complement)
  - 补码（Two's Complement)
  - 移码（Offset Binary)

简单的原码表示: Sign Magnitude Representation (SMR)



SMR对有符号整数的表示



SMR对有符号纯小数的表示

简单的原码表示: Sign Magnitude Representation (SMR)

设机器字长为8位，如下整数/纯小数的原码表示如下：

符号位

十进制数	7	6	5	4	3	2	1	0
35	0	0	1	0	0	0	1	1
-35	1	0	1	0	0	0	1	1
127	0	1	1	1	1	1	1	1
-127	1	1	1	1	1	1	1	1
+0	0	0	0	0	0	0	0	0
-0	1	0	0	0	0	0	0	0

小数点位置

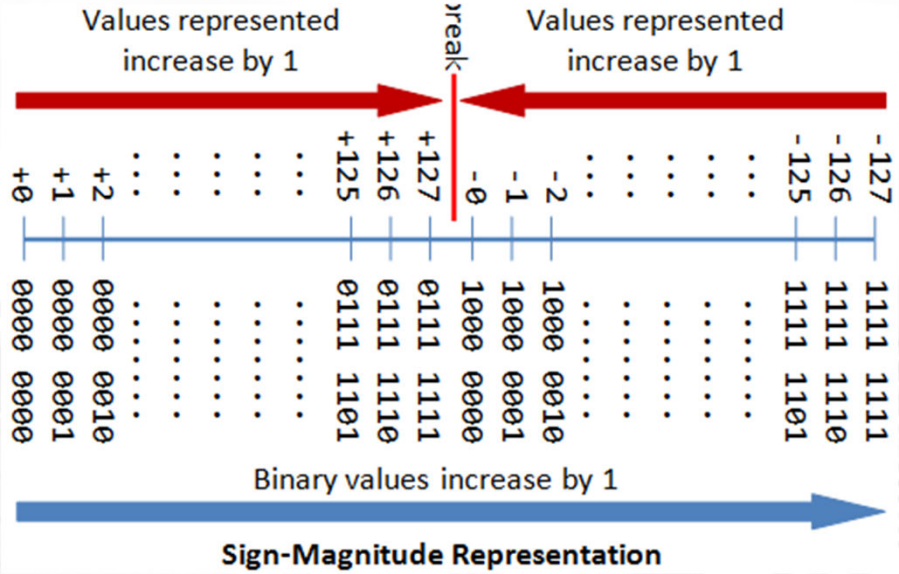
符号位

十进制数	7	6	5	4	3	2	1	0
0.5	0	1	0	0	0	0	0	0
-0.5	1	1	0	0	0	0	0	0
0.125	0	0	0	1	0	0	0	0
-0.125	1	0	0	1	0	0	0	0
0.3	0	0	1	0	0	1	1	0
-0.3	1	0	1	0	0	1	1	0

小数点位置

简单的原码表示: Sign Magnitude Representation (SMR)

- 总结
  - 原码比较直观，其表示方法同英文名SMR一致
    - 符号位（一位） + 数值（绝对值，N-1位）位
  - 原码表示中，0存在+0和-0两个值
  - 机器位数/表示长度决定了表达的
    - 整数范围$[-2^{N-1}+1, 2^{N-1}-1]$，最小表示单位1
    - 纯小数范围$\left[-1+2^{-(N-1)}, 1-2^{-(N-1)}\right]$，最小表示单位$2^{-(N-1)}$



直观的反码表示: One’s Complement

整数/纯小数的反码变换方法（精华版）如下：

- **正数：就是原码！**
- **负数：除了符号位，将原码逐位取反！**
- **整数/纯小数，都适用~**

- **Done!**

直观的反码表示: One’s Complement

整数/纯小数的反码变换方法（精华版）如下：for 正数：就是原码！for 负数：原码，除了符号位，全部取反!

设机器字长为8位，如下整数/纯小数的反码表示如下：

十进制数	7	6	5	4	3	2	1	0
35	0	0	1	0	0	0	1	1
-35	1	1	0	1	1	1	0	0
127	0	1	1	1	1	1	1	1
-127	1	0	0	0	0	0	0	0
+0	0	0	0	0	0	0	0	0
-0	1	1	1	1	1	1	1	1

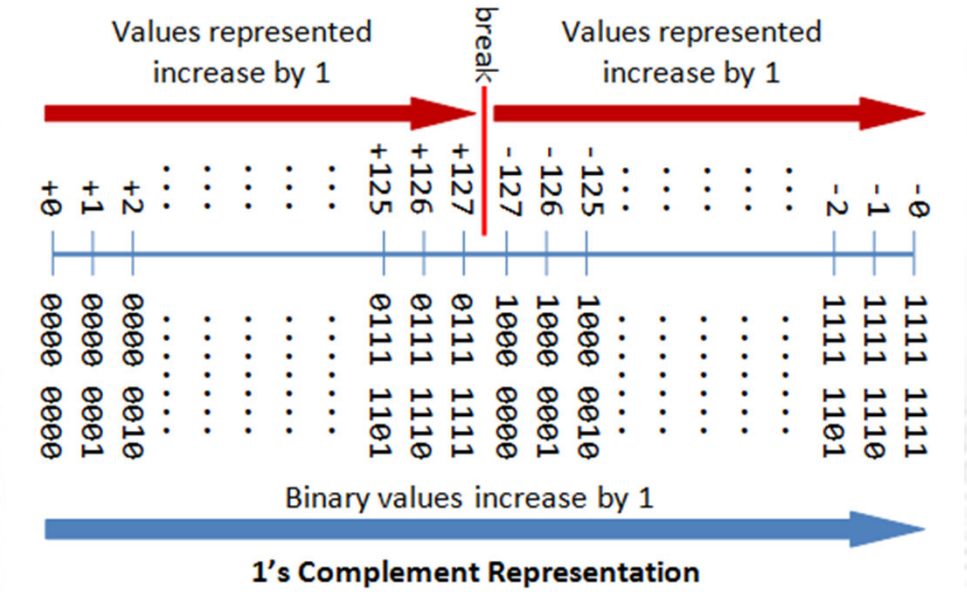
小数点位置

十进制数	7	6	5	4	3	2	1	0
0.5	0	1	0	0	0	0	0	0
-0.5	1	0	1	1	1	1	1	1
0.125	0	0	0	1	0	0	0	0
-0.125	1	1	1	0	1	1	1	1
0.3	0	0	1	0	0	1	1	0
-0.3	1	1	0	1	1	0	0	1

小数点位置

直观的反码表示: One’s Complement

- 总结
  - 正数的反码即原码，负数的反码为原码除符号位，逐位取反
  - 反码表示中，0也存在+0和-0两个值
  - 机器位数/表示长度决定了表达的
    - 整数范围$[-2^{N-1}+1, 2^{N-1}-1]$，最小表示单位1
    - 纯小数范围$\left[-1+2^{-(N-1)}, 1-2^{-(N-1)}\right]$，最小表示单位$2^{-(N-1)}$



有点复杂的补码表示: Two’s Complement

整数/纯小数的补码变换方法（精华版）如下：

- **正数：就是原码！**
- **负数：先算反码，反码最末位+1**
- **整数/纯小数，都适用~**

- **Done!**

有点复杂的补码表示: Two's Complement

整数/纯小数的补码变换方法（精华版）如下：for 正数：就是原码！for 负数：反码+1

设机器字长为8位，如下整数/纯小数的反码表示如下：

十进制数	7	6	5	4	3	2	1	0
35	0	0	1	0	0	0	1	1
-35	1	1	0	1	1	1	0	1
127	0	1	1	1	1	1	1	1
-127	1	0	0	0	0	0	0	1
+0	0	0	0	0	0	0	0	0
-0	0	0	0	0	0	0	0	0

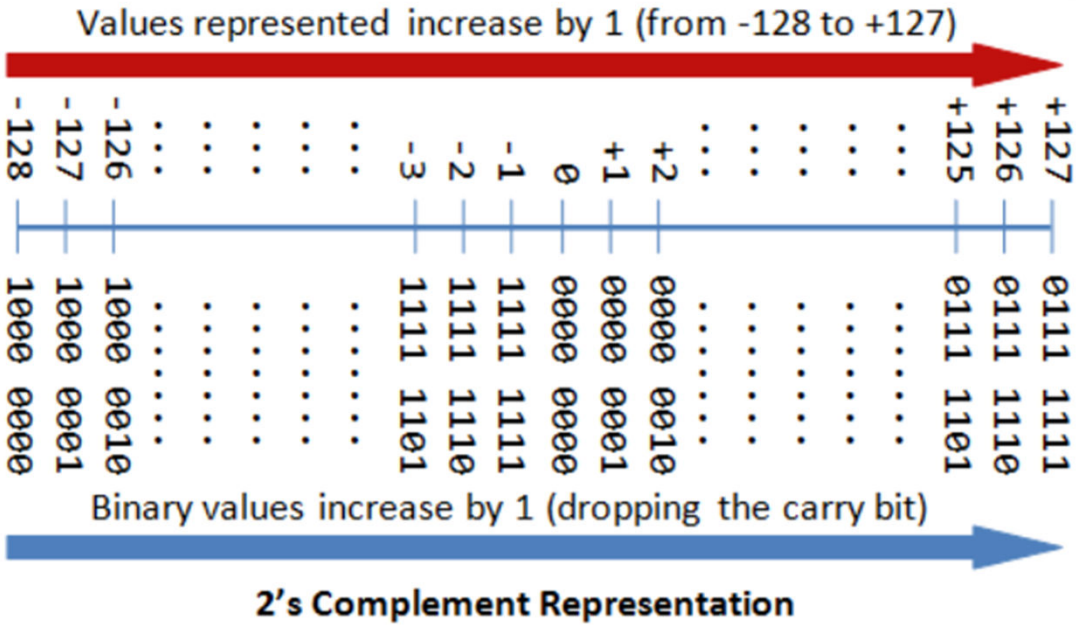
小数点位置

十进制数	7	6	5	4	3	2	1	0
0.5	0	1	0	0	0	0	0	0
-0.5	1	1	0	0	0	0	0	0
0.125	0	0	0	1	0	0	0	0
-0.125	1	1	1	1	0	0	0	0
0.3	0	0	1	0	0	1	1	0
-0.3	1	1	0	1	1	0	1	0

小数点位置

有点复杂的补码表示: Two's Complement

- **总结**
  - **正数补码即原码，负数补码为反码+1**
  - **补码表示中，0是唯一值**
  - **机器位数/表示长度决定了表达的**
    - **整数范围$[-2^{N-1}, 2^{N-1}-1]$，最小表示单位1**
    - **纯小数范围$[-1, 1-2^{-(N-1)}]$，最小表示单位$2^{-(N-1)}$**



最后的最后，移码表示: Offset Binary

整数/纯小数的移码变换方法（精华版）如下：

- **不论正负数，补码符号位取反！**
- **整数/纯小数，都适用~**

- **Done!**

最后的最后，移码表示: Offset Binary

整数/纯小数的反码变换方法（精华版）如下：补码符号位取反!

设机器字长为8位，如下整数/纯小数的反码表示如下：

十进制数	7	6	5	4	3	2	1	0
35	1	0	1	0	0	0	1	1
-35	0	1	0	1	1	1	0	1
127	1	1	1	1	1	1	1	1
-127	0	0	0	0	0	0	0	1
+0	1	0	0	0	0	0	0	0
-0	1	0	0	0	0	0	0	0

小数点位置

十进制数	7	6	5	4	3	2	1	0
0.5	1	1	0	0	0	0	0	0
-0.5	0	1	0	0	0	0	0	0
0.125	1	0	0	1	0	0	0	0
-0.125	0	1	1	1	0	0	0	0
0.3	1	0	1	0	0	1	1	0
-0.3	0	1	0	1	1	0	1	0

小数点位置

最后的最后，移码表示: Offset Binary

- **总结**
  - **移码就是补码符号位取反**
  - **移码表示中，0是唯一值**
  - **机器位数/表示长度决定了表达的**
    - **整数范围**$[-2^{N-1},2^{N-1}-1]$**，最小表示单位1**
    - **纯小数范围**$\left[-1,1-2^{-(N-1)}\right]$**，最小表示单位**$2^{-(N-1)}$

四码大PK

字长为4位，整数的四码表示

十进制数	原码	反码	补码	移码
8	N/A	N/A	N/A	N/A
7	0111	0111	0111	1111
6	0110	0110	0110	1110
5	0101	0101	0101	1101
4	0100	0100	0100	1100
3	0011	0011	0011	1011
2	0010	0010	0010	1010
1	0001	0001	0001	1001
0	0000 / 1000	0000 / 1111	0000	1000
-1	1001	1110	1111	0111
-2	1010	1101	1110	0110
-3	1011	1100	1101	0101
-4	1100	1011	1100	0100
-5	1101	1010	1011	0011
-6	1110	1001	1010	0010
-7	1111	1000	1001	0001
-8	N/A	N/A	1000	0000

四码大PK

字长为4位，纯小数的四码表示

十进制数	原码	反码	补码	移码
1	N/A	N/A	N/A	N/A
0.875	0111	0111	0111	1111
0.750	0110	0110	0110	1110
0.625	0101	0101	0101	1101
0.500	0100	0100	0100	1100
0.375	0011	0011	0011	1011
0.25	0010	0010	0010	1010
0.125	0001	0001	0001	1001
0.000	0000 / 1000	0000 / 1111	0000	1000
-0.125	1001	1110	1111	0111
-0.250	1010	1101	1110	0110
-0.375	1011	1100	1101	0101
-0.500	1100	1011	1100	0100
-0.625	1101	1010	1011	0011
-0.750	1110	1001	1010	0010
-0.875	1111	1000	1001	0001
-1.000	N/A	N/A	1000	0000

四码大PK

字长为4位整数，求二进制在不同表示下的真值（10进制）

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

四码大PK

字长为4位，二进制在不同表示下的真值

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