Welcome

数据科学与大数据技术专业

计 算 机 系 统 基 础

上海体育学院经济管理学院

Wu Ying

回顾要点（I）

理解计算机

1. 个人计算机的硬件组成
2. 存储程序 原理
3. 可编程
4. 冯·诺依曼体系结构
5. 总线
6. 0 - 1 序列 指挥电路动作
7. 高级语言、汇编语言、机器码
8. 计算机发展阶段、特点以及应用
9. 总线概述、总线分类、系统总线分类、总线仲裁
10. 位、字节、地址、存储单元、地址总线位数与寻址范围

理解运算

1. 电路怎样实现运算，与或非逻辑运算
2. 传统逻辑、布尔代数、香农开关
3. CPU的演化
4. 与主存一起完成自动加法计算

深入 CPU 和 主存

1. 一条指令的执行过程
2. 程序，多条指令的连续执行
3. 冯·诺依曼机的基本工作原理
4. 指令和机器码（指令的分类、格式）
5. 模型机
6. 指令周期（Instruction Cycle）、机器周期（CPU/Machine Cycle）、时钟周期（Clock Cycle）
7. 微操作
8. 抽象

指令集和系统抽象层次

1. 指令集 + 指令集体系结构 = 指令系统
2. 计算机系统的抽象层次、不同用户
3. 操作系统、用户接口、编译与解释

回顾要点（II）

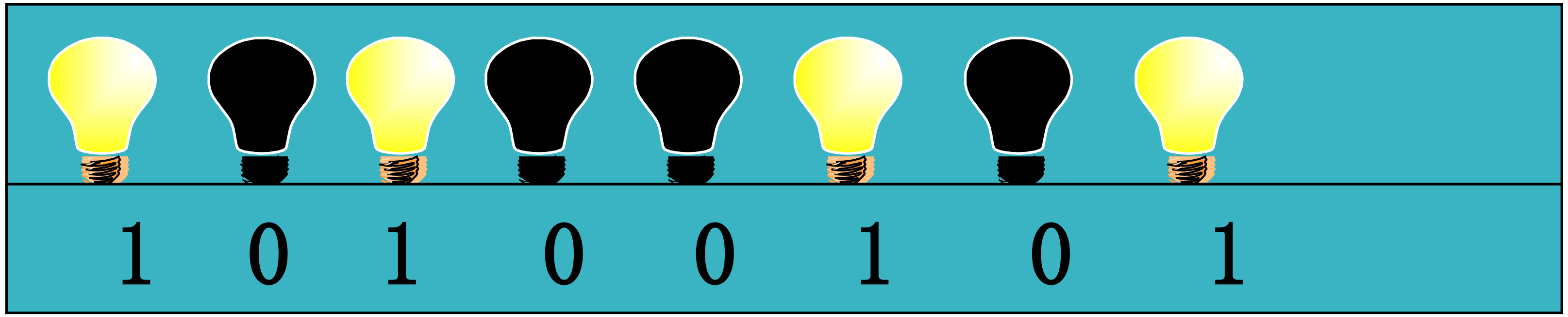
计算机系统

1. 硬件系统及软件系统的抽象层次
2. 软件系统及其分类（系统软件、支持软件、应用软件）
3. 应用操作：Windows tips、文字处理软件（Word、记事本、Markdown等）
4. 系统性能评价

   程序执行时间、MIPS、Amdahl定律

数据的机器表示与处理

1. 编码、数字化
2. 数制、进制转换
3. 定点小数与定点整数
4. 浮点表示，规格化，IEEE754
5. 定点数的编码
6. 原码表示
7. 补码表示、特殊数据的补码表示
8. 整数的表示



输入设备

数值 十 / 二进制转换

西文 ASCII 码

汉字 输入码 / 机内码转换

声音、图像 模 / 数转换

内存

数值型数据：

整数、浮点数

非数值型数据：

位串

指令系统能识别的基本类型数据

输出设备

二 / 十进制转换 数值

西文字形码 西文

汉字字形码 汉字

数 / 模转换 声音、图像

数据的机器级表示与处理

01 数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

02 定点与浮点表示

- 定点小数与定点整数
- 浮点表示，规格化，IEEE754
- 定点数的编码
    - 原码表示
    - 补码表示、特殊数据的补码表示
- 整数的表示

2.1.3 定点与浮点表示

在计算机中，小数点的位置事先约定好

定点数 小数点位置约定在固定位置

定点小数
(纯小数)

符号	数值部分

小数点

定点整数
(纯整数)

符号	数值部分

小数点

小数点位置约定为可浮动的数 **浮点数**

尾数 M

阶码 E

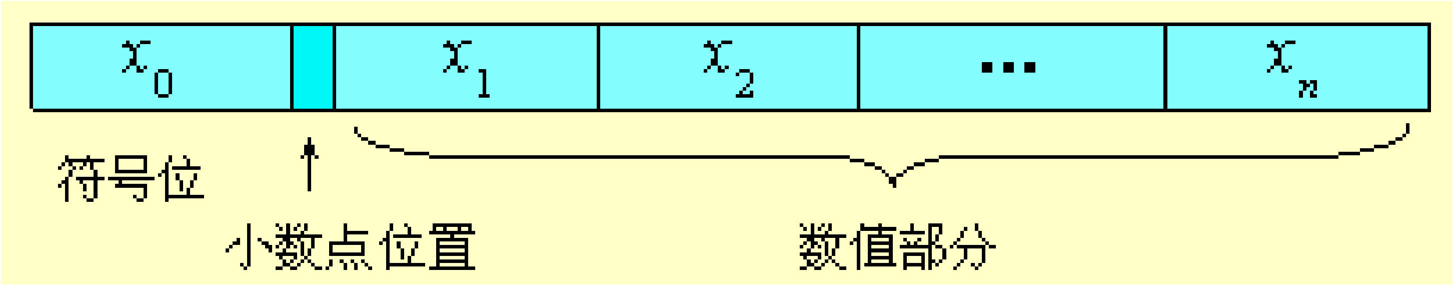
$$X = (-1)^{s} \times M \times R^{E}$$

s 为 0或1

IEEE单精度浮点数存储：23位小数f、8位偏置指数e、1位符号s

定点小数

定点小数是纯小数，约定小数点位于符号位后（带符号的定点小数）



定点小数，绝对值最小的非零数 ？

$|x|_{min} = 2^{-n}$

符号	1	0	0	0
位序 .	1	2	3	4

定点小数，绝对值何时最大?

$|x|_{max} = 1-2^{-n}$

定点小数的表示范围

$2^{-n} \le |x| \le 1 - 2^{-n}$

定点整数

定点整数是纯整数，约定小数点位于有效数值部分最低位之后

符号位	x_{m-1}	…	x_1	x_0

$\underbrace{x_{m-1} \ \cdots \ x_1 \ x_0}_{\text{数值部分}}$

定点整数，绝对值最小的非零数 ？

$|x|_{min} = 1$

符号	0	0	0	1
位序	3	2	1	0 .

定点整数，绝对值何时最大?

$|x|_{max} = 2^m - 1$

定点整数的表示范围

$1 \le |x| \le 2^m - 1$

$-(2^m - 1) \le x \le 2^m - 1$

- 当数据小于定点数能表示的最小值时，计算机将它们作0处理，称为下溢
- 大于定点数能表示的最大值时，计算机将无法表示，称为上溢
- 上溢和下溢统称为溢出

IEEE754 浮点数标准【2.3.3 P45-46】

规定：小数点前总是“1”，可隐含。

隐含一位后，使得单精度格式的23位尾数能够表示24位有效数字，

双精度52位尾数能够表示53位有效数字

单精度，偏置常数为127

0.011011×2^{-3}

0.000011011×2^{0}

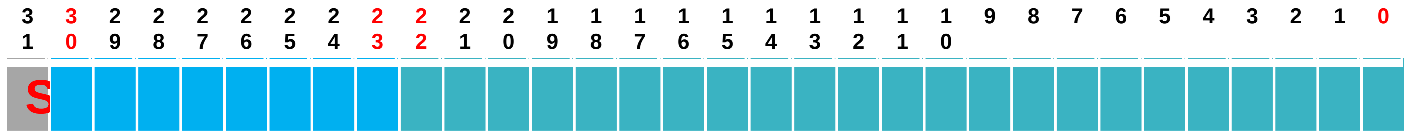
0.11011×2^{-4}

1.1011×2^{-5}

$X = (-1)^{s} \times M \times R^{E}$

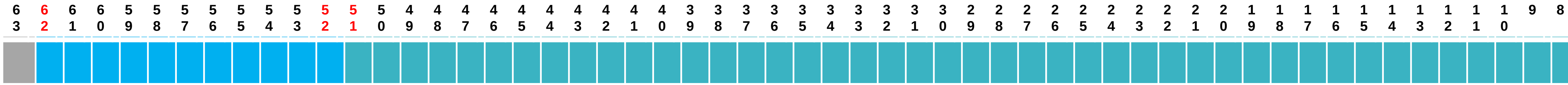
$(-1)^{S} \times 1.M \times 2^{(E-127)}$

32位单精度格式



1 bit 8 bits 阶码 23 bits 尾数

64位双精度格式

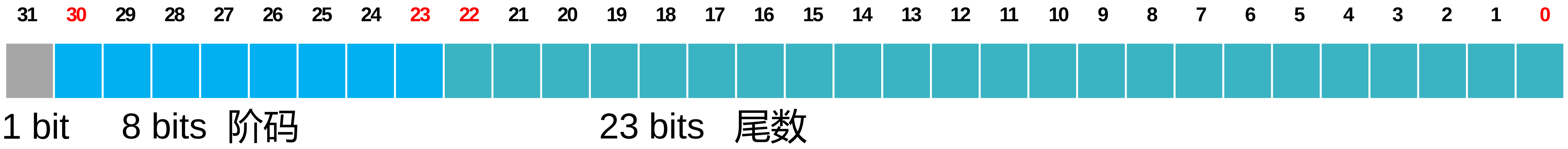


1 bit 11 bits 阶码 52 bits 尾数

IEEE754 浮点数标准【2.3.3 P45-46】

BEE00000H is the hex. Rep. Of an IEEE 754 SP FP number

32位单精度格式



IEEE754 浮点数标准【2.3.3 P45-46】

BEE00000H is the hex. Rep. Of an IEEE 754 SP FP number

32位单精度格式

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 bit　8 bits 阶码　23 bits 尾数

符号位: 1 => 负数　-1

阶码:

$(01111101)_2 = (125)_{10}$

单精度偏置常数为127: 125 - 127 = -2

尾数: $(1.1100000000000000000000)_2$

该单精度数二进制数对应的十进制值为:

$-(1.1100000000000000000000)_2 \times 2^{-2} = -(0.0111)_2 = -(1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}) = -0.4375$

“Father” of the IEEE 754 standard

直到80年代初，各个机器内部的浮点数表示格式还没有统一，因而相互不兼容，在机器之间传送数据时带来麻烦

1970年代后期，IEEE成立委员会着手制定浮点数标准

1985年完成浮点数标准IEEE 754的制定

This standard was primarily the work of one person, UC Berkeley math professor William Kahan.

现在所有计算机都采用IEEE 754来表示浮点数

www.cs.berkeley.edu/~wkahan/ieee754status/754story.html

Prof. William Kahan

浮点数X的范围

阶码 E
定点纯整数

s 为 0或1

$$X = \boxed{(-1)^{s} \times M} \times R^{E}$$

尾数 M
定点纯小数

绝对值最小的非零数 ？ $|x|_{min}$ ？

绝对值最大的数 ？ $|x|_{max}$ ？

定点小数的表示范围

$$2^{-n} \le |x| \le 1 - 2^{-n}$$

定点整数的表示范围

$$1 \le |x| \le 2^{m}-1$$

$$-(2^{m}-1) \le x \le 2^{m}-1$$

浮点数 X 的范围

阶码 E
定点纯整数

s 为 0或1

$$X = \boxed{(-1)^{s} \times M} \times R^{E}$$

尾数 M
定点纯小数

浮点数X的表示范围

$$\frac{2^{-n}}{2^{(2^m-1)}} \le |X| \le (1-2^{-n}) \times 2^{(2^m-1)}$$

绝对值最小的非零数 ？ $|x|_{min}$

$$0.0\ldots01 \times 2^{-111\ldots1} = 2^{-n} \times 2^{-(2^m-1)}$$

$$= \frac{2^{-n}}{2^{(2^m-1)}}$$

绝对值最大的数 ？ $|x|_{max}$

$$0.1\ldots11 \times 2^{111\ldots1} = (1-2^{-n}) \times 2^{(2^m-1)}$$

例如，$n = 7$，$m = 7$

浮点数 X 的范围

阶码 E
定点纯整数

s 为 0或1

$$X = \boxed{(-1)^{s} \times M} \times R^{E}$$

尾数 M
定点纯小数

浮点数X的表示范围

$$\frac{2^{-n}}{2^{(2^m-1)}} \le |X| \le (1-2^{-n}) \times 2^{(2^m-1)}$$

绝对值最小的非零数 ？ $|x|_{\min}$

$$0.0\ldots01 \times 2^{-111\ldots1} = 2^{-n} \times 2^{-(2^m-1)}$$

$$= \frac{2^{-n}}{2^{(2^m-1)}}$$

绝对值最大的数 ？ $|x|_{\max}$

$$0.1\ldots11 \times 2^{111\ldots1} = (1-2^{-n}) \times 2^{(2^m-1)}$$

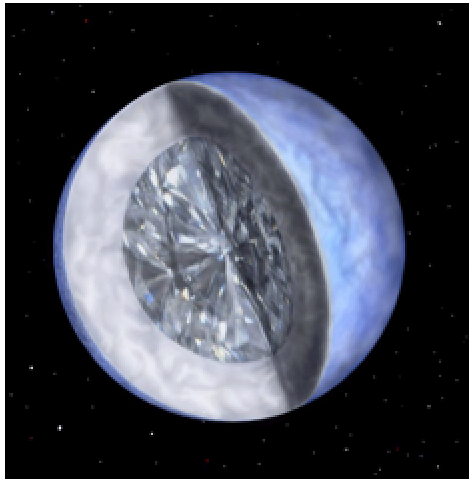
例如，$n = 7$，$m = 7$

$$\frac{2^{-7}}{2^{(2^7-1)}} \le |X| \le (1-2^{-7}) \times 2^{(2^7-1)}$$

$$\frac{2^{-7}}{2^{(127)}} \le |X| \le (1-2^{-7}) \times 2^{127}$$

$$\frac{0.0078125}{1.70\times10^{38}} \le |X| \le 0.9921875 \times 1.70 \times 10^{38}$$

$10^{38} = 10^4 \times 10^8 \times 10^8 \times 10^8 \times 10^8$



宇宙的钻石星球，代码为BPM37093，是一颗白矮星。位于靠近南十字座的半人马座，距离地球约50光年。

这颗星球的外层覆盖着氢氦气体，宝石藏在气体下面。它原本是一颗太阳大小的恒星，这颗恒星燃烧尽了所有的核燃料，最终由于恒星内部强大的压力使碳结晶，形成钻石。有着一个直径3000千米的钻石内核，是迄今人类已知的银河系中最大的钻石，**10的34次方克拉**（即重量达100**亿亿亿亿克拉,2000亿亿亿吨**）。

1 克拉=0.2克或200毫克=200×10^{-9}吨=2e-7 吨

$10^{34} \times 2 \times 10^{-7} = 2 \times 10^{27} = 2 \times 10^3 \times 10^8 \times 10^8 \times 10^8$

数据的机器级表示与处理

01

数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

02

定点与浮点表示

- 定点小数与定点整数
- 浮点表示，规格化，IEEE754
- 定点数的编码
    - 原码表示
    - 补码表示、特殊数据的补码表示
- 整数的表示

2.1.4 定点数的编码表示

机器数：数值数据在计算机内部**编码后的01序列**

真值：现实世界带有正负号的数，真正的值

假设 真值 X_T 的 n 位二进制码如下：

X为定点整数时： $X_T = \pm X'_{n-2} \ldots X'_1 X'_0$

X为定点小数时： $X_T = \pm 0.X'_{n-2} \ldots X'_1 X'_0$

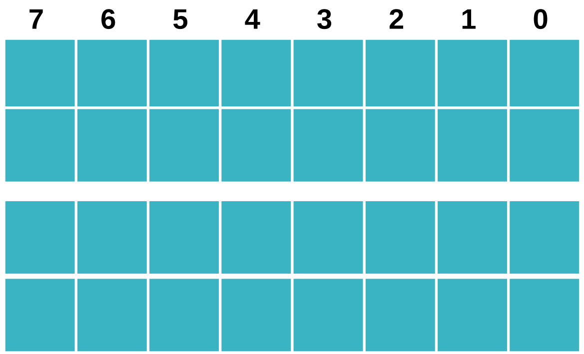
±1100100. B ±100

±0.0011001 B ±0.1953125

对真值 X_T 用 n 位二进制编码后，机器数 X 表示为：

$$X = X_{n-1}\ X_{n-2} \ldots X_0$$

符号 数值



n=8

2.1.4 定点数的编码表示

机器数：数值数据在计算机内部**编码后的01序列**

真值：现实世界带有正负号的数，真正的值

假设 真值 X_T的 n 位二进制码如下：

X为定点整数时： $X_T = \pm X'_{n-2} \ldots X'_1 X'_0$　　±1100100. B　　±100

X为定点小数时： $X_T = \pm 0.X'_{n-2} \ldots X'_1 X'_0$　　±0.0011001 B　　±0.1953125

对真值 X_T用 n 位二进制编码后，机器数 X 表示为：

$$X = X_{n-1}\ X_{n-2} \ldots X_0$$

符号 数值

7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	0
1	1	1	0	0	1	0	0
0	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1

n=8

定点数的编码

原码表示法

符号位+数值位构成，正数和负数的编码仅符号位不同，数值部分完全相同

±100　　n=8

编码规则如下：

①当真值 X_T为正数时，X_{n-1} = 0，$X_i = X_i'$ $(0 \le i \le n-2)$

②当真值 X_T为负数时，X_{n-1} = 1，$X_i = X_i'$ $(0 \le i \le n-2)$

	7	6	5	4	3	2	1	0
真值	+	1	1	0	0	1	0	0
原码	0	1	1	0	0	1	0	0

	7	6	5	4	3	2	1	0
真值	-	1	1	0	0	1	0	0
原码	1	1	1	0	0	1	0	0

对于真值 -10 （1010B），若用8位原码表示，原码为?

7	6	5	4	3	2	1	0

对于真值 -0.625（-0.101B），若用8位原码表示，原码为?

7	6	5	4	3	2	1	0

原码表示法

符号位+数值位构成，正数和负数的编码仅符号位不同，数值部分完全相同

±100 n=8

编码规则如下：

①当真值 X_T为正数时， $X_{n-1}=0$，$X_i = X_i'$ $(0 \le i \le n-2)$

②当真值 X_T为负数时， $X_{n-1}=1$，$X_i = X_i'$ $(0 \le i \le n-2)$

	+	1	1	0	0	1	0	0
真值	+	1	1	0	0	1	0	0
	7	6	5	4	3	2	1	0
原码	0	1	1	0	0	1	0	0
真值	-	1	1	0	0	1	0	0
	7	6	5	4	3	2	1	0
原码	1	1	1	0	0	1	0	0

对于真值 -10 （1010B），若用8位原码表示，原码为?

7	6	5	4	3	2	1	0	
1	0	0	0	1	0	1	0	8AH 或 0x8A

对于真值 -0.625（-0.101B），若用8位原码表示，原码为?

7	6	5	4	3	2	1	0	
1	1	0	1	0	0	0	0	D0H 或 0xD0

原码表示法

原码 0？ n=8	
$[+0]_{原} = 0\ 0000000$	$[-0]_{原} = 1\ 0000000$

原码表示

原码表示法，与真值对应关系直观、方便，与真值之间的转换简单

0 的表示不唯一，带来不便

现代计算机中不用原码表示整数，仅仅用定点原码小数来表示**浮点数的尾数部分**

习题，P79 4，假定机器数为8位（1位符号，7位数值）

写出下列各定点小数的原码表示

数值	原码
(1）+0.1001	
(2）-0.1001	
(3）+1.0	
(4）-1.0	
(5）+0.010100	
(6）-0.010100	
(7）+0.0	
(8）-0.0	

习题，P79 4，假定机器数为8位（1位符号，7位数值）

写出下列各**定点小数**的原码表示

数值	原码
(1) +0.1001	01001000
(2) -0.1001	11001000
(3) +1.0	溢出
(4) -1.0	溢出
(5) +0.010100	00101000
(6) -0.010100	10101000
(7) +0.0	00000000
(8) -0.0	10000000

数据的机器级表示与处理

01

数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

02

定点与浮点表示

- 定点小数与定点整数
- 浮点表示，规格化，IEEE754
- 定点数的编码
  - 原码表示
  - 补码表示、特殊数据的补码表示
- 整数的表示

例如： -5 + 4 用 8 位 原码表示，并计算

	10000101	············-5 的原码
+	00000100	············ 4 的原码
	10001001	············ 运算结果为-9

例如： -5 + 4 用 8 位 补码表示，并计算

	11111011	………… **−5** 的补码
+	00000100	………… **4** 的补码
	11111111	………… 运算结果为 **−1** 的补码

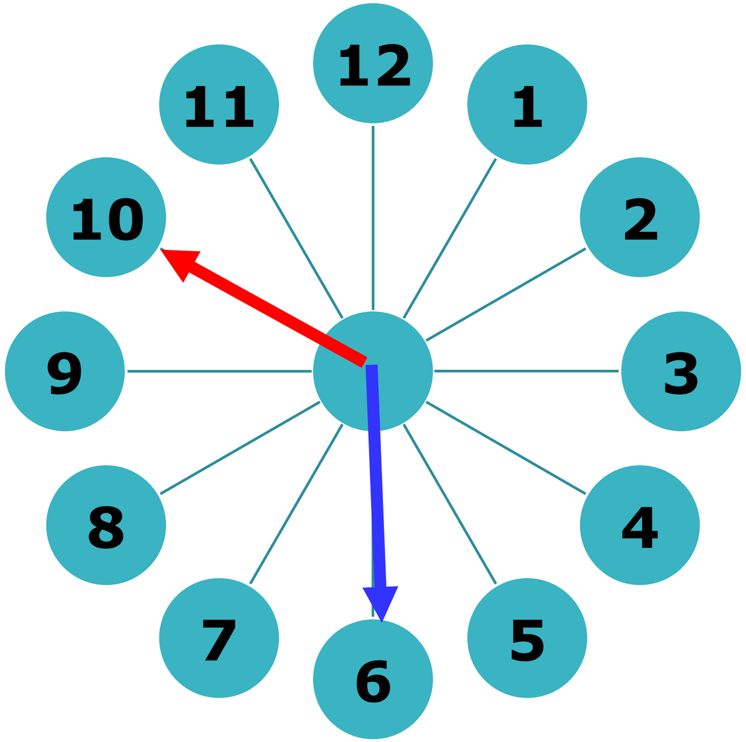
补码

补码表示可以实现加减运算的统一，用加法实现减法运算

例如： -5 + 4

	11111011	…………	**−5** 的补码
+	00000100	…………	**4** 的补码
	11111111	…………	运算结果为 **−1** 的补码

模 ——一个系统所能表示的数据个数



当前**10**点

逆时针拨**4**格：**10-4 = 6**

顺时针拨**8**格：**10+8=18** $\equiv 6 \ (mod\ 12)$

在模12系统中，

$10-4 \equiv 10 + (12-4)$

$\equiv 10 + 8 \quad (\text{mod } 12)$

即 $-4 \equiv 8\ (mod\ 12)$ 8 是 -4 对模12的补码

对某一确定的模，某数 A 减去小于模的数B，可以用A加上**-B的补码**来代替

“模”——一个计量系统的计量范围，即产生“溢出”的量

$A = B + K \times M(K\text{为整数})$
$18 = 6 + 1 \times 12$

$$A \equiv B \pmod{M}$$

B和A模M同余

例如，钟表系统，模为12

$$18 \equiv 6 \pmod{12}$$

补码（8位）	十进制数
0111 1111	127
…	…
0000 0100	4
0000 0011	3
0000 0010	2
0000 0001	1
0000 0000	0
1111 1111	-1
1111 1110	-2
1111 1101	-3
1111 1100	-4
1111 1011	-5
…	…
1000 0000	-128

例2.11 P38

假定算盘有4档，“4位10进制模运算系统”，且只能做加法，计算 9828 – 1928 ?

“4位10进制数模运算系统”的模为 10^4

例2.11 P38

假定算盘有4档，“4位10进制模运算系统”，且只能做加法，计算 9828 – 1928？

“4位10进制数模运算系统”的模为 10^4

$$\boxed{9828-1928} \equiv 9828+(10000-1928) \equiv \boxed{9828+8072} \equiv 7900(mod\ 10000)$$

用 9828 加上8072 （-1928的补码）来实现 9828-1928 的功能

两个异号数相加。舍去高位的操作相当于“将一个多于4位的数去除以 10^4 ，保留其余数作为结果”，模运算

补码的定义

根据**同余**概念和数的互补，补码的编码规则：

①正数的补码，符号为0，数值部分是它本身

当真值 X_T为正数时， $[X_T]_{补} = X_T = M + X_T (\mathrm{mod}\ \mathrm{M})$

②负数的补码，**等于模 与 该负数的绝对值 之差**

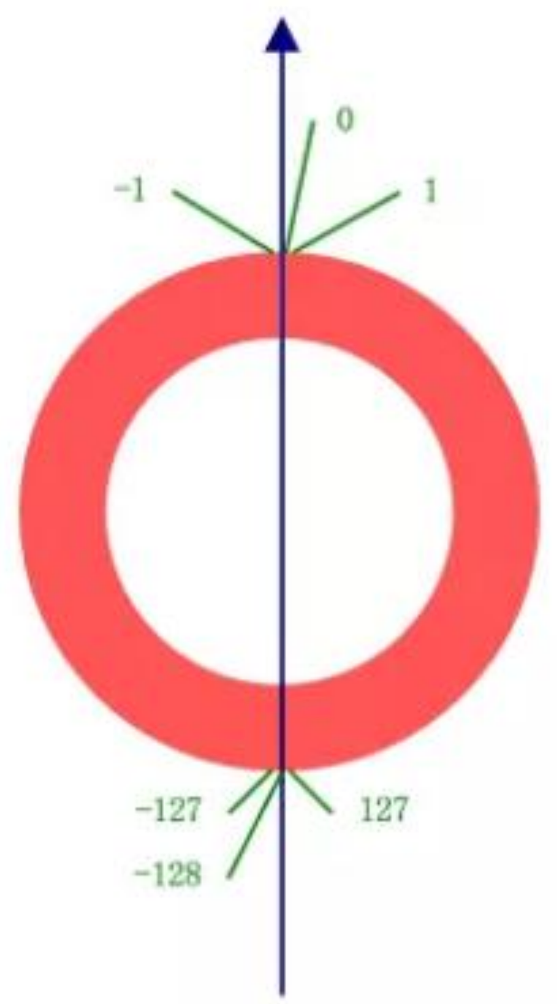
当真值 X_T为负数时， $[X_T]_{补} = M - |X_T| = M + X_T (\mathrm{mod}\ \mathrm{M})$

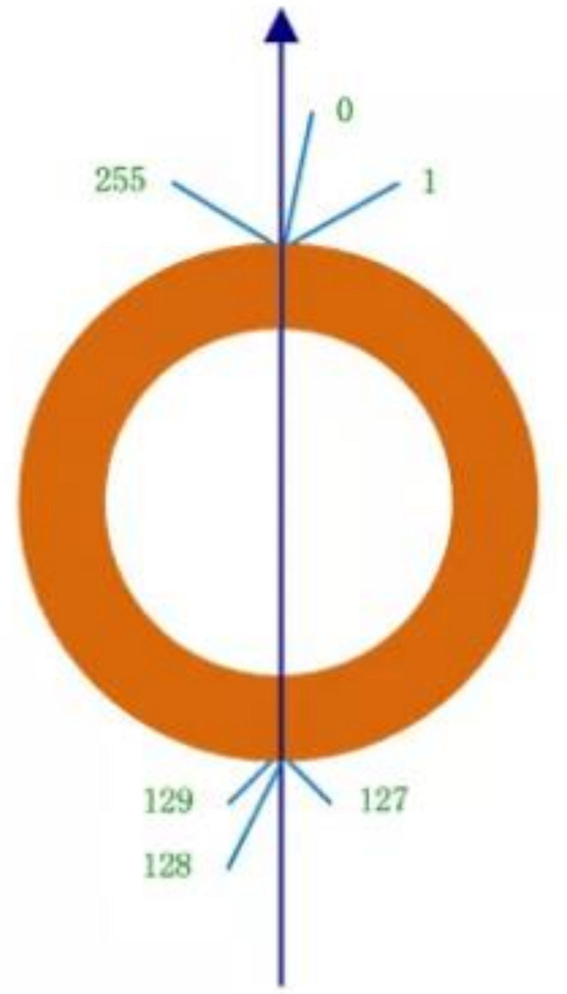
补码（8位）	十进制数
0111 1111	127
…	…
0000 0100	4
0000 0011	3
0000 0010	2
0000 0001	1
0000 0000	0
1111 1111	-1
1111 1110	-2
1111 1101	-3
1111 1100	-4
1111 1011	-5
…	…
1000 0000	-128

n=8

8位机器数

十进制数	原码	补码
127	0111 1111	0111 1111
126	0111 1110	0111 1110
…	…	…
3	0000 0011	0000 0011
2	0000 0010	0000 0010
1	0000 0001	0000 0001
0	0000 0000 <u>1000 0000：-0</u>	0000 0000
-1	1000 0001	1111 1111
-2	1000 0010	1111 1110
-3	1000 0011	1111 1101
…	…	…
-126	1111 1110	1000 0010
-127	1111 1111	1000 0001
-128		<u>1000 0000</u>





带符号整数（补码），8位为例，简单理解为关于环形对称

特殊数据的补码表示 【P39】

补码位数为 8 时，求$-2^{8-1}=-128$的**补码表示** $[X_T]_{补}=\boldsymbol{M}-|\boldsymbol{X_T}|=M+\ \boldsymbol{X_T}(\mathbf{mod\ M})$

补码位数为 9 时，求$-2^{8-1}=-128$的**补码表示**

特殊数据的补码表示 【P39】

位数为 8 时，求$-2^{8-1}=-128$的**补码表示**

$$[X_T]_{补}=M-|X_T|=M+X_T(\text{mod } M)$$

当补码为8位时，其模为 $2^8=256$，因此:

$[-2^{8-1}]_{补}=2^8-2^{8-1}=256-128=256+128$（mod 2^8）= 128 = 10000000

位数为 9 时，求$-2^{8-1}=-128$的**补码表示**

当补码为9位时，其模为 $2^9=512$，因此:

$[-2^{8-1}]_{补}=2^9-2^{8-1}=512-128=512+384$（mod 512) = 384 =110000000

特殊数据的补码表示 【P39】

位数为 10 时，求$-2^{8-1}=-128$的**补码表示**　　$[X_T]_{补}=M-|X_T|=M+X_T(\mathrm{mod\ M})$

当补码为10 位时，其模为 $2^{10}=1024$，因此:

$[-2^{8-1}]_{补}=2^{10}-128=1024-128=1024+896\ (\mathrm{mod}\ 1024)=896=11\ 1000\ 0000$

位数为 11 时，求$-2^{8-1}=-128$的**补码表示**

当补码为11位时，其模为 $2^{11}=2048$，因此:

$[-2^{8-1}]_{补}=2^{11}-128=2048-128=2048+1920\ (\mathrm{mod}\ 2048)=1920=111\ 1000\ 0000$

同一个真值在不同位数的补码表示中，对应的机器数不同

因此一定要明确编码位数，即运算部件的字长

特殊数据的补码表示 【P39】

位数为 16 时，求$-2^{8-1}=-128$的**补码表示**

当补码为16位时，其模为 $2^{16}=65536$，因此：

$[-2^{8-1}]_补=2^{16}-128=65536-128=$ 65536+ 65408（mod 65536) = 65408 =1111 1111 1000 0000

特殊数据的补码表示 【P39】

位数为 8 时，求**−1**的**补码表示**

当补码为8位时，其模为 $2^8 = 256$，因此：

$[-1]_补 = 2^8 - 1 = 256 + 255$（mod 2^8）= 255 = 11111111

位数为 8 时，求**0** 的**补码表示**

当补码为8位时，其模为 $2^8 = 256$，因此：

$[0]_补 = 2^8 - 0 = 256 + 256$（mod 2^8）= 00000000

补码0的表示是唯一的，减少+0和-0之间的转换

少占用一个编码表示，补码比原码能多表示一个最小负数 10000000 -128补码

补码的设计

(1) 消除正零负零转换，少占用一个编码表示，**最大负数向外拓展1位。**

(2) 使符号位能与有效值部分一起参加运算

	11111011	…………	**−5** 的补码
+	00000100	…………	**4** 的补码
	11111111	…………	运算结果为 **−1** 的补码

(3) 减法运算转换为加法运算，进一步简化计算机中运算器的线路设计。

(4) 对补码求补码，得到原码。

求补码的直接方法

①正数的补码，符号为0，数值部分是它本身

②负数的补码，符号位取1，其余对原码的数值位“各位取反，末位加1”

7	6	5	4	3	2	1	0	
								-5 原码
								符号位不变，各位取反
								末位+1

11111011 ………… −5 的补码

7	6	5	4	3	2	1	0	
								-5 补码
								符号位不变，各位取反
								末位+1

对补码求补码，得到原码

求补码的直接方法

①正数的补码，符号为0，数值部分是它本身

②负数的补码，符号位取1，其余对原码的数值位“各位取反，末位加1”

7	6	5	4	3	2	1	0	
1	0	0	0	0	1	0	1	-5 原码
1	1	1	1	1	0	1	0	符号位不变，各位取反
1	1	1	1	1	0	1	1	末位+1

11111011 ………… −5 的补码

7	6	5	4	3	2	1	0	
1	1	1	1	1	0	1	1	-5 补码
1	0	0	0	0	1	0	0	符号位不变，各位取反
1	0	0	0	0	1	0	1	末位+1

对补码求补码，得到原码

习题，P79 5，假定机器数为8位（1位符号，7位数值）

写出下列各定点整数的补码表示

数值	补码
(1) +1001	
(2) -1001	
(3) +1	
(4) -1	
(5) +10100	
(6) -10100	
(7) +0	
(8) -0	

习题，P79 5，假定机器数为8位（1位符号，7位数值）

写出下列各定点整数的补码表示

数值	补码
(1) +1001	00001001
(2) -1001	11110111
(3) +1	00000001
(4) -1	11111111
(5) +10100	00010100
(6) -10100	11101100
(7) +0	00000000
(8) -0	00000000

数据的机器级表示与处理

01

数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

02

定点与浮点表示

- 定点小数与定点整数
- 浮点表示，规格化，IEEE754
- 定点数的编码
    - 原码表示
    - 补码表示、特殊数据的补码表示
- 整数的表示

2.2 整数的表示 P41

整数表示

无符号数（非负）

带符号整数（补码）

CPU寄存器位数（字长）	无符号整数的表示范围
8	
16	
32	

有符号整数的表示范围

2.2 整数的表示 P41

整数表示

无符号数（非负）　　带符号整数

CPU寄存器位数（字长）	无符号整数的表示范围
8	0～255（2^8–1）
16	0～65535（2^{16}–1）
32	0～2^{32}–1

有符号整数的表示范围
–128 (–2^7) ～ 127（2^7–1）
–32768 (–2^{15}) ～ 32767（2^{15}–1）
–2^{31}～ 2^{31}–1

int型在16位机上是16位，取值范围 –32768 (–2^{15}) ～ 32767

2.2.2 C语言中的整数及其相互转换

类型	字长	位数	范围
int	16位机，16位	16	-32768 (-2^{15}) ～ 32767 ($2^{15}-1$)
int	32位、64位机	32	-2147483648 (-2^{31}) ～ 2147483647 ($2^{31}-1$)
long	32位机	32	-2147483648 (-2^{31}) ～ 2147483647 ($2^{31}-1$)
long	64位机	64	-9,223,372,036,854,775,808 (-2^{63}) ～ 9,223,372,036,854,775,807 ($2^{63}-1$)
long long	ISO C99规定	64	-9,223,372,036,854,775,808 (-2^{63}) ～ 9,223,372,036,854,775,807 ($2^{63}-1$)

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<math.h>

int main()
{
    int x = -1;
    unsigned u = 2147483648;
    printf("带符号整数 x 的值为：%d\n", x);
    printf("带符号整数 x，无符号解读为：%u\n", x);
    printf("u 无符号整数：%u\n", u);
    printf("u 有符号整数值为：%d\n", u);
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
带符号整数 x 的值为：-1
带符号整数 x，无符号解读为：4294967295
u 无符号整数：2147483648
u 有符号整数值为：-2147483648
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<math.h>

int main()
{
    int x = -1;
    unsigned u = 2147483648;
    printf("带符号整数 x 的值为: %d\n", x);
    printf("带符号整数 x，无符号解读为: %u\n", x);
    printf("u 无符号整数：%u\n", u);
    printf("u 有符号整数值为：%d\n", u);
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
带符号整数 x 的值为: -1
带符号整数 x，无符号解读为: 4294967295
u 无符号整数：2147483648
u 有符号整数值为：-2147483648
```

-1 补码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

$2^{32} - 1 = 4294967295$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$2^{31} =$ **2147483648**

-2147183648的补码