计算机组织结构

3 数据的机器级表示

刘博涵

2022年9月15日

南京大学
NANJING UNIVERSITY

教材对应章节

第2章 数据的机器级表示

第10章 计算机算术

信息的二进制编码

- 在冯·诺依曼结构中，所有信息（代码和数据）都采用二进制编码
  - 编码：用少量简单的基本符号对复杂多样的信息进行一定规律的组合
  - 采用二进制的原因
    - 制造两个稳定态的物理器件容易
    - 二进制编码、计数运算规则简单
    - 对应逻辑命题中的“真”和“假”
    - 便于使用逻辑电路实现算术运算
  - 真值和机器数
    - K位的二进制编码**至多**表示2^k个不同的值

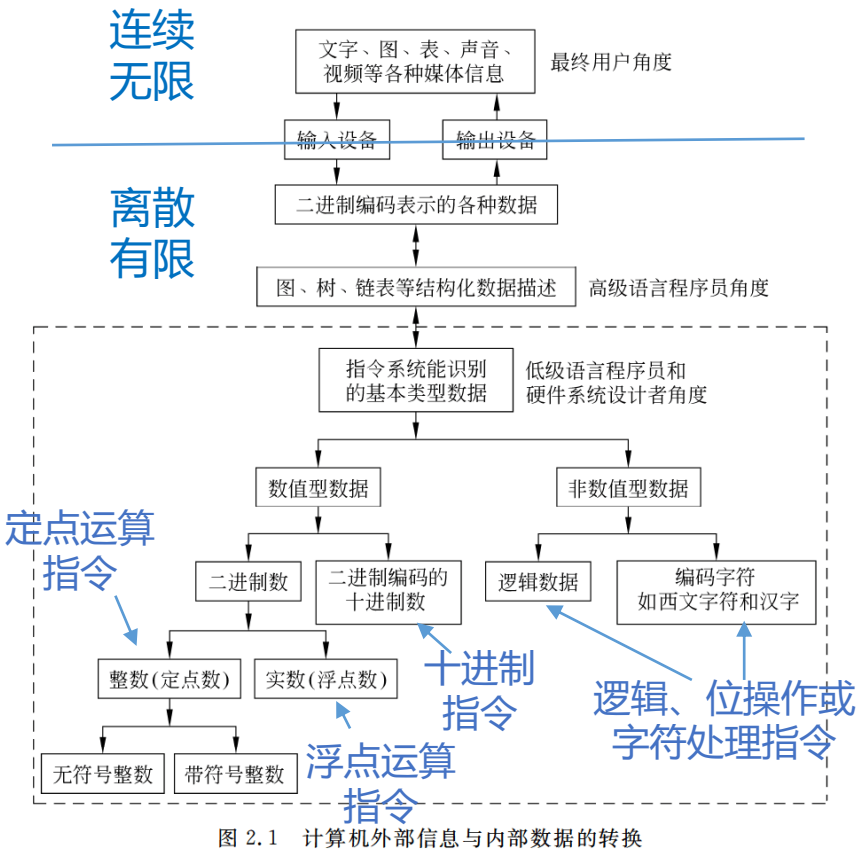


图 2.1　计算机外部信息与内部数据的转换

例：真值$127 = 2^7 - 1$，机器数为0000 0000 0111 1111

回顾：整数的二进制数表示

整数的二进制数表示

- 无符号整数
- 有符号整数：原码，反码，移码，补码
  - 编码是为了**解决正负号问题**
  - 计算机中几乎**不用反码**，运算**普遍使用补码**
  - 二进制补码的运算
  - 二进制-十进制转换

原码表示

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

浮点数的尾数用原码定点小数表示

优点:

- 最直接，便于理解

缺点:

- 0的表示不唯一，不利于程序员编程
- 加、减运算方式不统一，尤其当a<b时，实现a-b比较困难
- 需要额外对符号位进行处理，不利于硬件设计

移码表示

将每一个数值加上一个偏置常数（excess/bias）
通常当编码位数为n时，bias取2^{n-1}或$2^{n-1}-1$（如IEEE 754）
例如n=4时， bias=8

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

当bias为2^{n-1}，
移码和补码仅第一位不同

0的表示唯一

浮点数的阶用移码表示

例如计算 $1.01 \times 2^{-1} + 1.11 \times 2^{3}$时，需要将**低阶**（$2^{-1}$）**转为高阶**（$2^{3}$）
那么首先需要**比较-1和3的大小**

补码：111<011？
移码：011<111

此外，全为负数时，移码比原码更容易比较

补码表示

在一个模运算系统中，一个数于它除以“模”后的余数等价。

时钟是一种模12系统

- **例：**图中时针指向的是10点，要将它拨向6点，有两种拨法
  - 1、逆时针拨4格，10-4=6
  - 2、顺时针拨8格，10+8=18=6[mod 12]
- **加和减的统一：**
  - 一个负数的补码等于模减该负数的绝对值。
  - 对于某一确定的模，某数减去小于模的另一数，总可以加上另一数负数的补码来代替。
  - 一个**负数的补码**等于将**对应正数补码各位取反、末位加一**

8位二进制示例：

0111 1111 – 0100 0000 = 0111 1111 + (1 0000 0000 – 0100 0000)

= 0111 1111 + 1100 0000

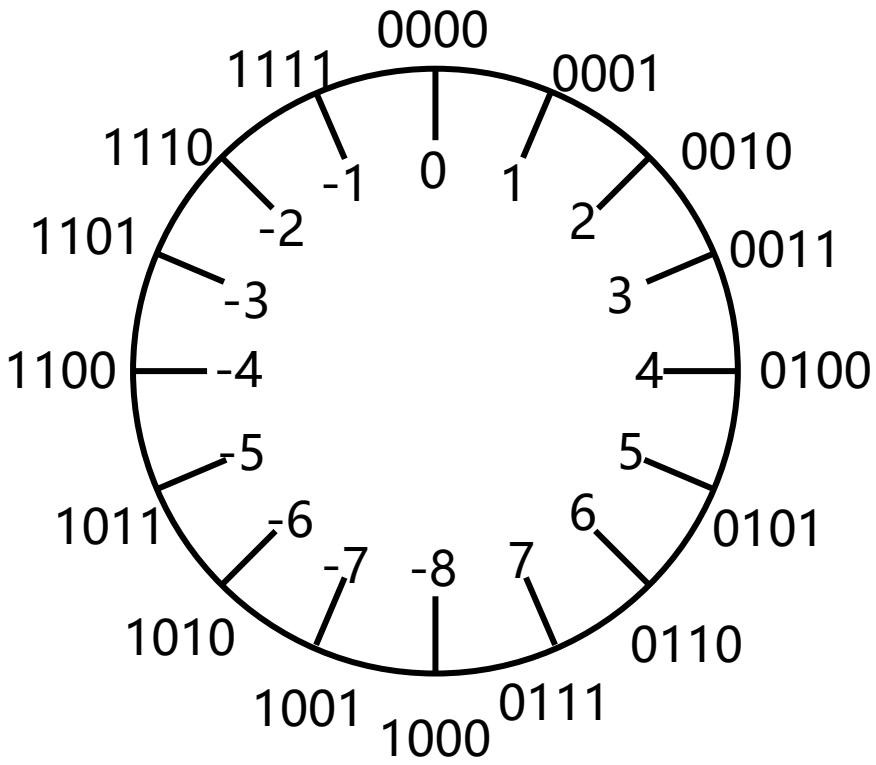
只留余数 = 1 0011 1111 (mod 1 0000 0000)

= 0011 1111

补码表示

一个n位运算器，只能保留低n位的运算结果，即模为2^n的补码运算

模为2^4的时钟系统



补码的定义：

$$[X]_c = 2^n + X \ (-2^n \le X \le 2^n, mod\ 2^n)$$

补码表示：

000...000 ~ 011...111：表示的值不变

100...000 ~ 111...111：表示的值由

$$2^{k-1} \sim 2^k - 1$$

变为

$$-2^{k-1} \sim -1$$

补码表示法的优势

- 补码表示法 vs. 原码表示法

	补码表示法		原码表示法	
9	0000 1001		0000 1001	
+ 8	+ 0000 1000		+ 0000 1000	
17	0001 0001	17	0001 0001	17
9	0000 1001		0000 1001	
+ -8	+ 1111 1000		+ 1000 1000	
1	10000 0001	1	1001 0001	-17

补码无论同号还是异号都可以直接相加

求真值的补码

特殊的真值：

$$[-2^{n-1}]_c = 2^n - 2^{n-1} = 10\ldots0\ (n-1个0)$$

$$[-1]_c = 2^n - 1 = 11\ldots1\ (n个1)$$

$$[+0]_c = [-0]_c = 00\ldots0\ (n个0)$$

在32位机器中n为32
int型　32位
short型 16位
char型　8位

一般的例子：

$$123 = 127 - 4 = 0111\ 1111\mathrm{B} - 0000\ 0100\mathrm{B} = 0111\ 1011\mathrm{B}$$

$$-123 = -0111\ 1011\mathrm{B}$$

$$[0111\ 1011]_c = 2^8 + 0111\ 1011 = 1\ 0111\ 1011\ (mod\ 2^8) = 0111\ 1011$$

$$[-0111\ 1011]_c = 2^8 - 0111\ 1011 = 1111\ 1111 - 0111\ 1011 + 1 = 1000\ 0101$$

0的补码唯一，为$00\ldots0\ (n个0)$
正数的补码等于它本身
负数的补码为各位取反，末位加1

求补码的真值

令： $[X]_C = x_{n-1}x_{n-2}\ldots x_2x_1$

则： $X = -x_{n-1}\times 2^{n-1} + \cdots + x_2\times 2^1 + x_1\times 2^0$

真值的范围

$-2^{n-1} \le X \le 2^{n-1}-1$

例： 补码**1101 0110**的真值为

$$-2^7\times 1 + 2^6\times 1 + 2^5\times 0 + 2^4\times 1 + 2^3\times 0 + 2^2\times 1 + 2^1\times 1 + 2^0\times 0 = -42$$

简便求法：

符号为0，则为正数，数值部分相同

符号位1，则位负数，数值各位取反，末位加1

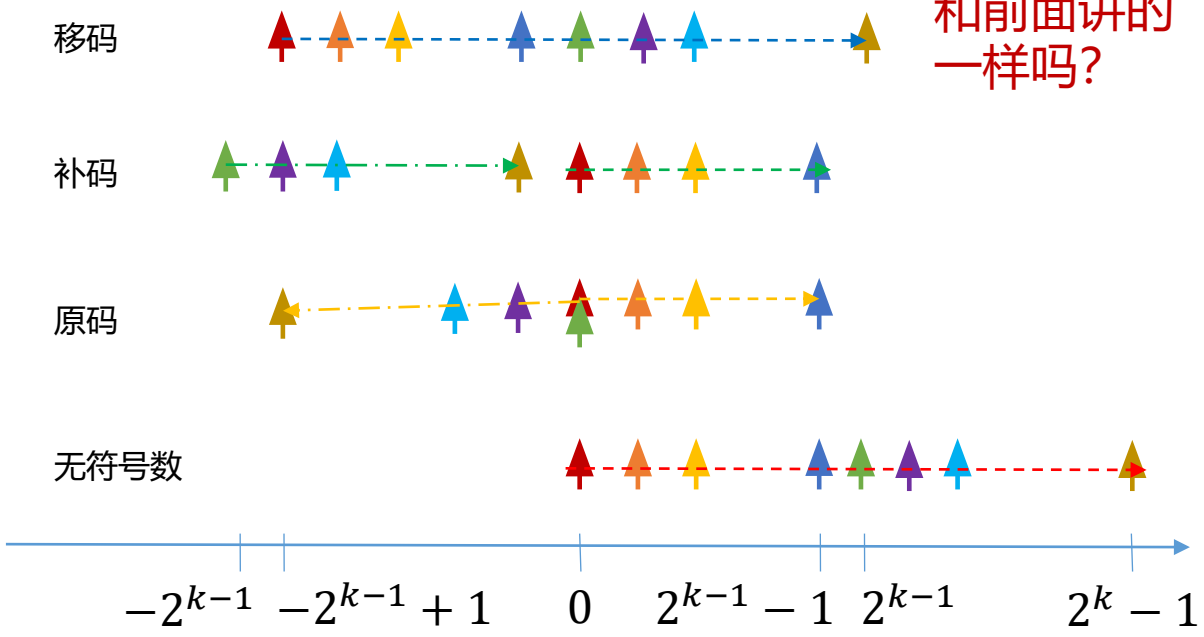
例： 补码**0101 0110**的真值为

$$+\mathbf{101\ 0110} = 64 + 16 + 4 + 2 = 86$$

例： 补码**1101 0110**的真值为

$$-\mathbf{010\ 1001} = -(32 + 8 + 2) = -42$$

扩展：不同的整数编码



和前面讲的一样吗？

以8位为例，真值的表示范围

- 无符号：0~255　　$0 \sim 2^k-1$
- 原码：-127~127　　$-2^{k-1}+1 \sim 2^{k-1}-1$
- 移码：-127~128　　$-2^{k-1}+1 \sim 2^{k-1}$
- 补码：-128~127　　$-2^{k-1} \sim 2^{k-1}-1$

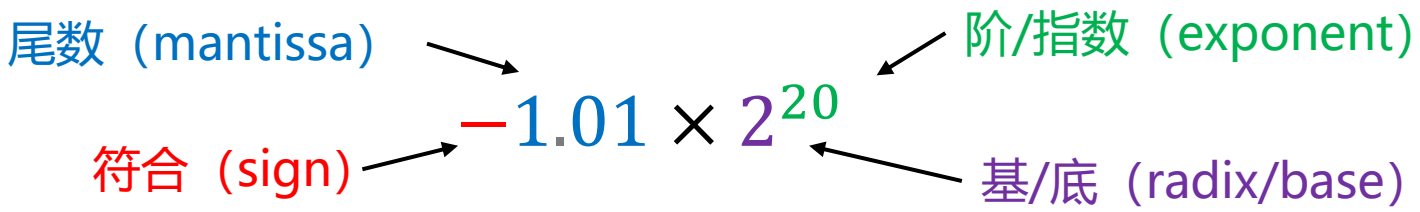
移码在IEEE754中
尾数需要右移一位，所以
从-128~127变成了-127~128
实际可用范围是**-126~127**

回顾：浮点数的二进制数表示

浮点数的二进制数表示

十进制数的科学计数法

二进制数的科学计数法



基底不变，1位即可以表示符号位
对尾数和阶分别编码即可以表示一个**浮点数**

规格化数

- 任何浮点数都能以多种样式来表示

$$0.110 \times 2^5, \qquad 110 \times 2^2, \qquad 0.0110 \times 2^6$$

- 规格化表示

$$\pm 1.bbb\ldots b \times 2^E$$

第一位永远是1

$$X = (-1)^S \times M \times 2^E$$

1位　8位**移码**　　23位**原码**表示**24位**尾数

-127 ~ 128
（ bias取$2^{n-1}-1$ ）

$$\mathbf{-1 - 0.11\ldots11 \sim 1 + 0.11\ldots11}$$

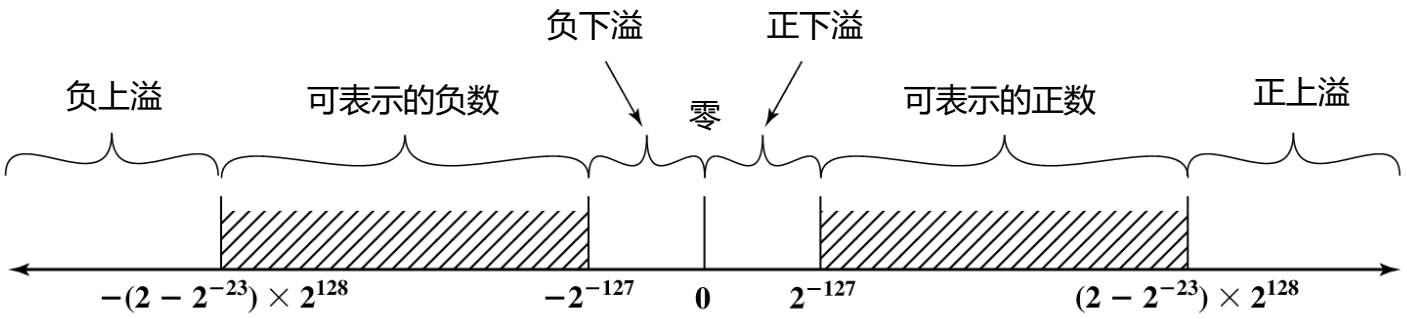
$\mathbf{2^{23}}$个1

$$\mathbf{0.11\ldots11 = 1 - 0.00\ldots01 = 1 - 2^{-23}} \longrightarrow \text{最大正数：}\ \mathbf{(2 - 2^{-23}) \times 2^{128}}$$

最小正数：*E=0，M=0.00…00* $\longrightarrow$ $\mathbf{2^{-127}}$

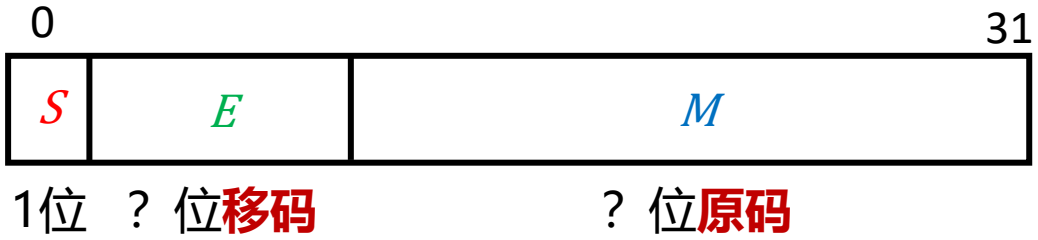
规格化数的值的范围

- 值的范围
  - 介于 $-(2-2^{-23})\times 2^{128}$ 和 -2^{-127} 之间的负数
  - 介于 2^{-127} 和 $(2-2^{-23})\times 2^{128}$ 之间的正数



原码的表示范围是对称的，所以浮点数值的范围也关于原点对称

规格化数的变化



$$X = (-1)^S \times M \times B^E$$

其他规格化数的表示**形式**： $\pm 0.1bb\ldots b \times 2^E$

- 对于一定长度的规格化数，表示范围和精度之间存在权衡（**总位数不变**）
  - **增加**阶码（E）位数：**扩大**表示**范围**，**降低**表示**精度**
  - **增加**尾数（M）位数： **减少**表示**范围**，**提高**表示精度，
  - **更大**的底（B）：如4，8，16，**扩大**表示**范围**，**降低**表示精度

需要一个统一的标准

IEEE 754 标准


Larger Photo

William M. Kahan

Professor Emeritus

Research Areas

Computer Architecture & Engineering (ARC)
Scientific Computing (SCI)
Computer architecture; Scientific computing; Numerical analysis

20世纪70年代，IEEE成立委员会着手制定浮点数标准。

20世纪80年代，Intel邀请Kahan教授设计8087处理器的浮点运算单元。

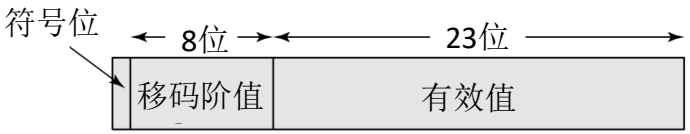
IEEE邀请Kahan基于8087中的浮点标准起草一份通用标准。

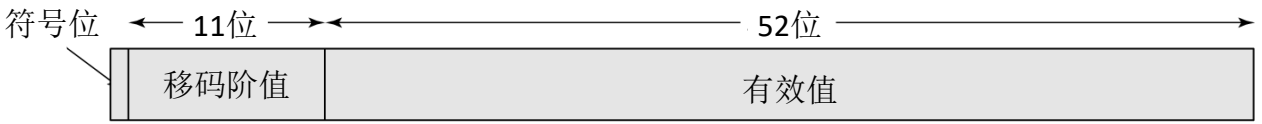
1985年提出浮点数标准**IEEE 754**。

1989年Kahan教授因此获得ACM A.M Turing Award

IEEE 754 标准

- **定义32位的单精度和64位的双精度两种格式**

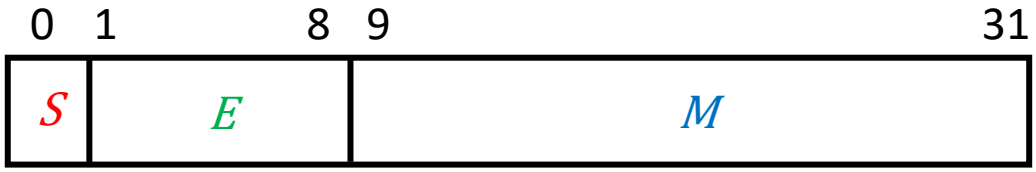




- **定义两种拓展格式**
  - 扩展单精度浮点格式（≥43 位，不常用）。
  - 扩展双精度浮点格式（≥79 位，一般情况下，Intel x86 结构的计算机采用的是 80 位，而 SPARC 结构的计算机采用的是 128 位）。

IEEE 754 标准

单精度浮点数



1位　8位**移码**　　23位**原码**表示**24位**尾数

S：**1表示负，0表示正**

E：**全0和全1表示特殊值，**
范围是 0000 0001 (-126) ~ 1111 1110 (127)

M：**最高位总是1，所以隐含表示**

阶码的范围为什么是-126~127？为什么不是-128~127？

为什么采用$\pm 1.bbb\ldots b \times 2^E$？

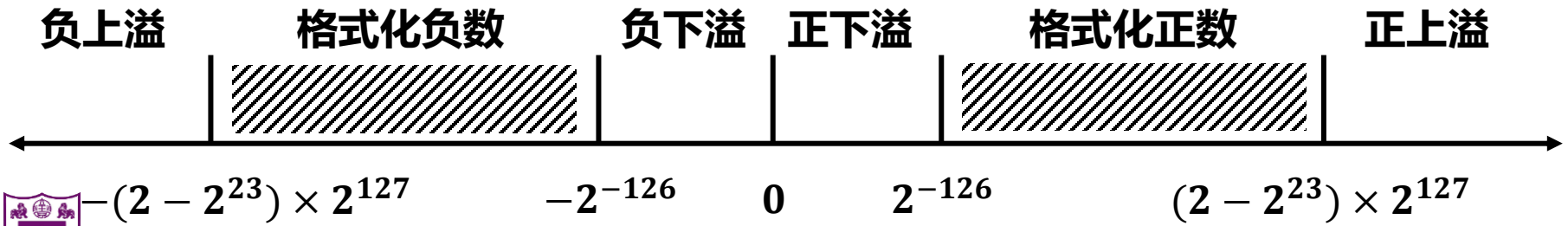
IEEE 754 标准

IEEE754 格式化浮点数表示范围

格式	最小正数	最大正数	最小负数	最大负数
单精度	$E=1$ $M=0$ 1.0×2^{-126}	$E=254$ $M=1-2^{23}$ $(2-2^{23})\times 2^{127}$	$E=254$ $M=1-2^{23}$ $-(2-2^{23})\times 2^{127}$	$E=1$ $M=0$ -1.0×2^{-126}
双精度	$E=1$ $M=0$ 1.0×2^{-1022}	$E=2046$ $M=1-2^{52}$ $(2-2^{52})\times 2^{1023}$	$E=2046$ $M=1-2^{52}$ $-(2-2^{52})\times 2^{1023}$	$E=1$ $M=0$ -1.0×2^{-1022}

注：均为真值

原码的表示范围是对称的，所以浮点数值的范围也关于原点对称



溢出区域怎么表示?

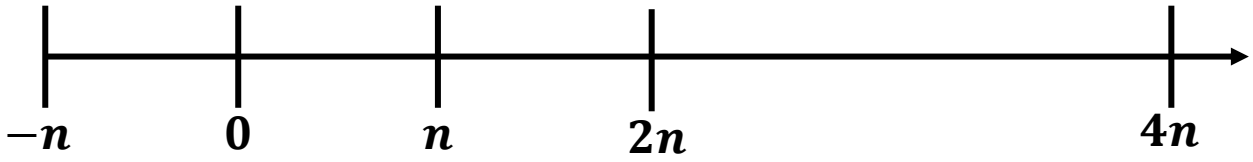
IEEE 754 标准

精度不够怎么办？
舍入原则

单精度每格表示2^{23}个数

增大取值范围

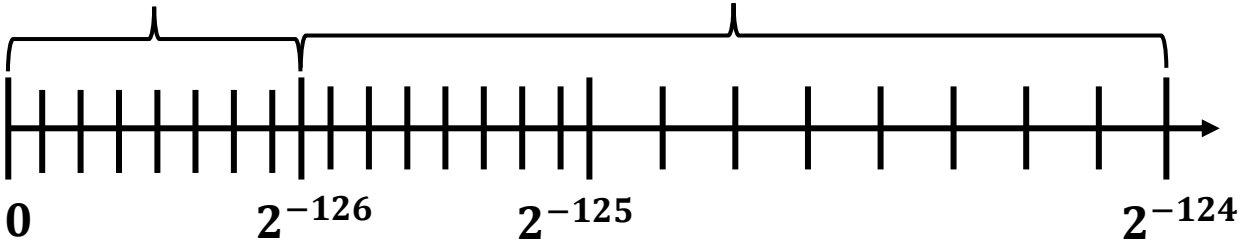
- 采用$\pm 1.bbb\ldots b \times 2^E$
- bias取$2^{n-1}-1$



精度随着绝对值的增大而降低

阶码全0，尾数非0 → **非格式化数**　　**格式化数**

$(-1)^s \times 0.bb\ldots b \times 2^{-126}$



全0和全1

- 用于表示格式化数以外的情况
- 负上溢和正上溢中只能表示无穷

阶码的值	尾数的值	表示
0（全0）	0	+/- 0
0（全0）	非0	非规格化数
1~254	任意	规格化数
255（全1）	0	+/- ∞
255（全1）	非0	NaN

IEEE 754 标准 (cont.)

- 格式参数

参数	格式			
	单精度	单精度拓展	双精度	双精度拓展
字宽（位数）	32	≥43	64	≥79
阶值位宽（位数）	8	≥11	11	≥15
阶值偏移量	127	未指定	1023	未指定
最大阶值	127	≥1023	1023	≥16383
最小阶值	−126	≤−1022	−1022	≤−16382
数的范围（底为10）	$10^{-38}, 10^{+38}$	未指定	$10^{-308}, 10^{+308}$	未指定
有效值位宽（位数）	23	≥31	52	≥63
阶值的数目	254	未指定	2046	未指定
小数的数目	2^{23}	未指定	2^{52}	未指定
值的数目	1.98×2^{31}	未指定	1.99×2^{63}	未指定

IEEE 754 标准 (cont.)

小数第一位：0
用法：表示未初始化的值，用于捕获异常

小数第一位：1
用法：表示未定义的算术结果，如除数等于0

	单精度（32位）				双精度（64位）			
	符号	移码阶值	小数	值	符号	移码阶值	小数	值
正零	0	0	0	0	0	0	0	0
负零	1	0	0	−0	1	0	0	−0
正无穷大	0	255 (all 1s)	0	∞	0	2047 (all 1s)	0	∞
负无穷大	1	255 (all 1s)	0	−∞	1	2047 (all 1s)	0	−∞
静默式非数	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
通知式非数	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
正的规格化非零数	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
负的规格化非零数	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
正的非规格化数	0	0	$f \neq 0$	$2^{e-126}(0.f)$	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
负的非规格化数	1	0	$f \neq 0$	$-2^{e-126}(0.f)$	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

IEEE 754 标准 (cont.)

- 例子

0.5 = 0.100…0B = (1.00..0)2×2^(-1)

0 01111110 000…00 (23)

-0.4375 = -0.01110…0B = - (1.110..0)2×2^(-2)

1 01111101 110…00 (21)

二进制编码的十进制数表示

- 浮点运算的问题
  - 精度限制
  - 转换成本高
- 应用需要
  - 长数字串的计算：会计，……
- 解决方法
  - 用4位二进制编码十进制（BCD）表示0, 1, ..., 9，直接计算

二进制编码的十进制数表示 (cont.)

- 自然BCD码 (NBCD, 8421 码)
  - 0 ~ 9: 0000 ~ 1001
  - 符号: 使用四个最高有效位
    - 正: 1100 / 0
    - 负: 1101 / 1
  - 例子
    - +2039: **1100** 0010 0000 0011 1001 / **0** 0010 0000 0011 1001
    - -1265: **1101** 0001 0010 0110 0101 / **1** 0001 0010 0110 0101
- 其他BCD码
  - 2421, 5211, 4311, …

非数值数据的编码表示

- **逻辑值**
  - **逻辑数据**和**数值数据**在形式上**没有差异**
  - 通过**指令的操作码**类型来**识别运算类型**，如逻辑运算指令，算术运算指令。
- **西文字符**
  - 有**多种**不同的**字符集**
  - 最广泛使用的是**ASCII码**
    - **7位或8位二进制表示**
- **汉字字符**
  - 一个**字**就是一个**方块图形**
  - 如GB2312-80**字符集**
  - 超过**6万**个字（**至少16位**）
- **多媒体信息（图像、音频等）**
  - 同样用0和1表示，**数据结构**各异

表 2.6 ASCII 码表

$b_3b_2b_1b_0$ \ $b_6b_5b_4$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	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	SYN	&	6	F	V	f	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

总结

- 信息的二进制编码
- 整数的二进制表示
  - 补码表示的优势，表示方法，真值计算
  - 不同的整数二进制表示
- 浮点数的二进制表示
  - 浮点数表示方法，规格化数，非规格化数，IEEE 754标准
- 二进制编码的十进制数表示
  - NBCD码表示方法
- 非数值数据的编码表示
  - 逻辑值，西文字符，汉字字符，多媒体信息

谢谢

bohanliu@nju.edu.cn

南京大學
NANJING UNIVERSITY