



第二章 数据的表示与处理

主讲：李博涵

bhli@nuaa.edu.cn

数据的机器级表示
数据的存储与运算



第三讲：浮点数的表示

主要内容

- 整数的表示
 - 十进制数与二进制数
 - 无符号整数与带符号整数
 - 补码与真值的转换、LSB/MSB
- C语言程序中的整数
- 数与数值的意义
 - 数的编码
 - 数的语义
- 二进制符号数
 - 原码方案
 - 补码方案
- 课堂练习



回顾：整数的表示

重要知识点：

1. 补码（对照原码）
2. 无符号数与带符号数（C语言）
3. 编码与语义（上下文，整体与局部）

接下来学习，计算机如何表示带小数点的数？

- 浮点数
 - 格式（精巧）
 - IEEE 754



带小数点的数

根据补码的定义 假定补码有 n 位, 则:

定点整数: $[X]_{\text{补}} = 2^n + X \quad (-2^n \leq X < 2^n, \text{ mod } 2^n)$

$[-2^{n-1}]_{\text{补}} = 2^n - 2^{n-1} = 10\dots 0$ ($n-1$ 个0) $(\text{mod } 2^n)$

$[-1]_{\text{补}} = 2^n - 0\dots 01 = 11\dots 1$ (n 个1) $(\text{mod } 2^n)$

定点小数: $[X]_{\text{补}} = 2 + X \quad (-1 \leq X < 1, \text{ mod } 2)$

总结: 定点整数可能无法达到要求的精度, 定点小数范围很小。

示例: $[-1.0]_{\text{补}} = 2 - 1.0 = 1.00\dots 0$ ($n-1$ 个0) $(\text{mod } 2)$

特殊情况: 除了理解就只能脑补记忆。

那想表示范围更大更精准的数值, 就需要浮点数的帮助。

浮点数再现



- 计算机除了处理整数外，很多时候也需要处理浮点数
 - 例如 π ，就无法用前面讲的二进制整数编码方案表示
 - 在工程计算中，大量计算涉及浮点数（图片，视频等）
- 浮点数具有表示范围和精度都较高的特点
 - 它解决了整数和小数位长度固定的限制
 - 允许表示一个很大的数或者很小的数

**1989
ACM Turing
Award Winner!**



William M. Kahan（威廉·凯亨）

浮点数标准之父；1989年图灵奖获得者
领导开发了Intel的8087浮点协处理器
1985年完成浮点数标准IEEE 754的制定

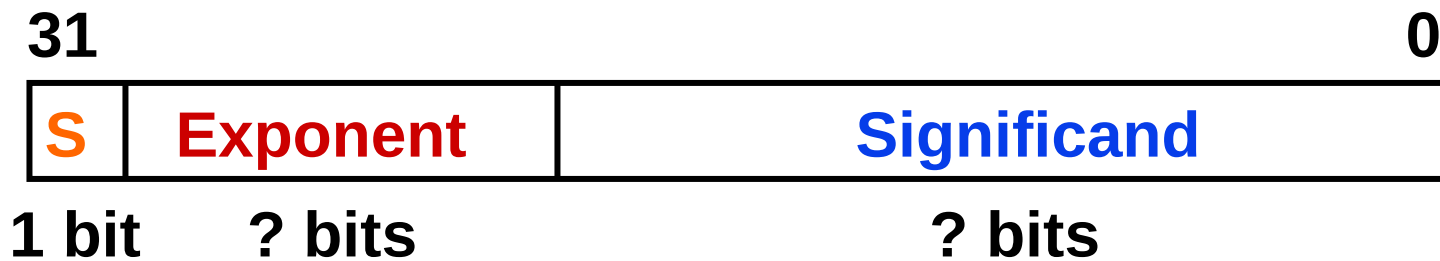


浮点数的一般表示格式

- Normal format（规格化数形式）：

$$+/-1.\text{xxxxxxxxxxx} \times R^{\text{Exponent-bias}}$$

- 32位规格化数：



S 是符号位（Sign）

Exponent 用移码来表示（为什么用移码？）

Significand 表示 xxxxxxxxxxxxxx，尾数部分（用原码）

（基可以是 2 / 4 / 8 / 16，约定信息，无需显式表示）

32位是不是随便切？

规格化数



- Exponent (阶码 / 指数) :
 - 偏置常数 : 127 (single), 1023 (double) (为什么偏置常数是127 ?)
 - SP规格化数阶码范围为0000 0001 (-126) ~ 1111 1110 (127)
- Significand (尾数) :
 - 规格化尾数最高位总是1 , 所以隐含表示 , 省1位
 - 1 + 23 bits (single) , 1 + 52 bits (double)
- 已知机器数求真值
 - $+/-1.\text{xxxxxxxxxx}_{\text{two}} \times 2^{\text{Exponent-bias}}$
 - SP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$
 - DP: $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-1023)}$

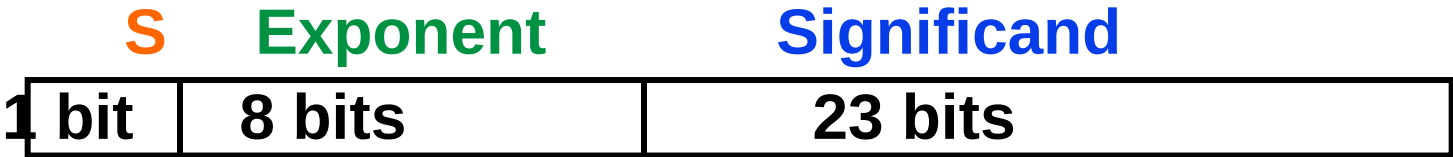
指数和尾数，增加任何一位数都会减少另一方位数

目前方案是在精度与表示范围之间反复权衡后的结果



IEEE 754标准

- 单精度SP :



符号：1位，0~正，1~负
指数：8位，用于表示范围
尾数：23位，用于表示精度

表示的数	单精度浮点数			双精度浮点数		
	符号	指数	尾数	符号	指数	尾数
0	X	0	0	X	0	0
$+\infty$	0	255	0	0	2047	0
$-\infty$	1	255	0	1	2047	0
NaN（非数）	X	255	非0	X	2047	非0
正的浮点数	0	1~254	任意	0	1~2046	任意
负的浮点数	1	1~254	任意	1	1~2046	任意

DP
1,11,52



对比两个绝对值小于1的数

- 绝对值小于1的浮点数，其指数是负的
 - 示例：0.5和0.25，其对应的就是 1.0×2^{-1} 与 1.0×2^{-2}
- 缺点：负指数的浮点数虽较小，但其二进制却似乎是个较大的数
 - 示例：0.5与2.0

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
S	指数								尾数																						
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

0.5 (1.0×2^{-1}) 的浮点数格式为：0x7F800000

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
S	指数								尾数																						
0	0	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	0	0	0	0	

2.0 ($1.0 \times 2^{+1}$) 的浮点数格式：0x00800000

浮点数：0.5 < 2.0

二进制：0x7F800000 > 0x00800000



偏置常数

- 目标： 00000000_2 对应最小的负指数，而 11111111_2 对应最大的正指数
- 思路： 编码用的指数=真实指数+127； 也称为偏阶记数法
- 示例： 0.5与2.0
 - 0.5的偏阶编码指数： $11111111_2 + 01111111_2 = 01111110_2$
 - 2.0的偏阶编码指数： $00000001_2 + 01111111_2 = 10000000_2$

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
S	指数								尾数																						
0	0	1	1	1	1	1	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

0.5 (1.0×2^{-1}) 的浮点数格式为： 0x3F000000

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
S	指数								尾数																						
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	

2.0 ($1.0 \times 2^{+1}$) 的浮点数格式为： 0x40000000

浮点数精度的重要性



■ **SpaceX**火箭成功上天，两位宇航员乘坐着载人**龙飞船**飞向天空。

然而，在**1996**年欧洲航天局耗时八年的**Ariana 5**火箭在发射后**37**秒爆炸。原因是控制火箭飞行的软件存在故障，这件事可以说是历史上损失最惨重的软件故障事件之一了。事故发生后一项资料给出了真因，是**64**位浮点数转换为**16**位带符号整数时产生了缓冲区溢出。这个原本可以通过修改数的变量类型就能避免的错误，造成了**80**多亿美金的损失。

■ 这个事件也告诉我们，计算机中的数据看似简单，但往往隐藏着一些不容易被察觉的错误，这种错误有时会带来重大损失，因此，对数据的处理要格外注意。