浮点数的编码表示

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

C语言支持的基本数据类型

C语言声明	操作数类型	存储长度（位）
(unsigned) char	整数 / 字节	8
(unsigned) short	整数 / 字	16
(unsigned) int	整数 / 双字	32
(unsigned) long int	整数 / 双字	32
(unsigned) long long int	-	2×32
char *	整数 / 双字	32
float	单精度浮点数	32
double	双精度浮点数	64
long double	扩展精度浮点数	80 / 96

实数类型分：单精度浮点、浮点双精度和扩展精度浮点

科学计数法(Scientific Notation)与浮点数

对于科学计数法（十进制数）：

mantissa (尾数) *exponent(阶、指数)*

6.02×10^{21}

小数点 *基 (base，radix)*

° Normalized form（规格化形式）：小数点前只有一位非0数

° 同一个数有多种表示形式。例：对于数 1/1,000,000,000

- Normalized (规格化形式): 1.0×10^{-9} 唯一
- Unnormalized（非规格化形式）: 0.1×10^{-8}, 10.0×10^{-10}

不唯一

对于二进制数实数

mantissa（尾数） *exponent（阶、指数）*

$1.101_{two} \times 2^{-10}$

小数点 *基为2*

只要对尾数和指数分别编码，就可表示一个浮点数（即：实数）

浮点数(Floating Point)的表示范围

例：画出下述32位浮点数格式的规格化数的表示范围。

0	1　　　　8	9　　　　　　　　　　31
S	阶码E	尾数M

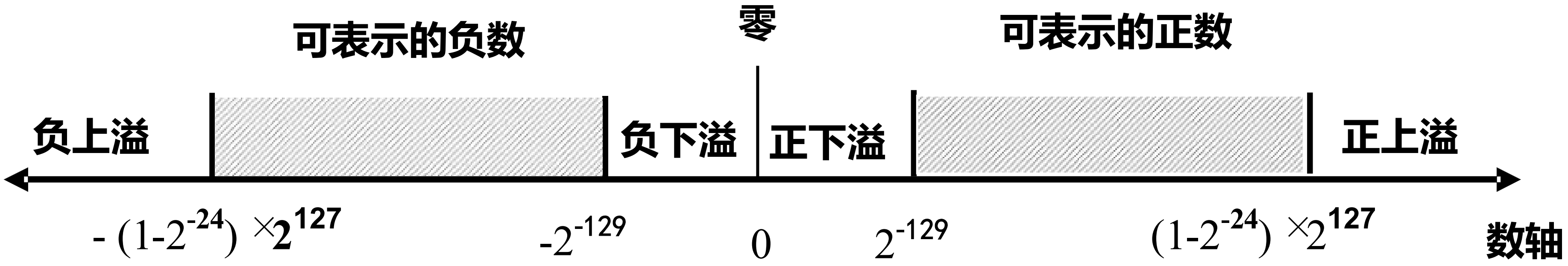
+/-0.1xxxxx * 2^E

第0位数符S；第1～8位为8位移码表示阶码E（偏置常数为128）；第9～31位为24位二进制原码小数表示的尾数M。规格化尾数的**小数点后第一位总是1**，故规定第一位默认的“1”不明显表示出来。这样可用23个数位表示24位尾数。

最大正数：0.11…1 x $2^{11…1}$ ＝$(1-2^{-24})$ x 2^{127}

最小正数：0.10…0 x $2^{00…0}$ ＝$(1/2)$ x 2^{-128}

因为原码对称，故其表示范围关于原点对称。



机器0：尾数为0 或 落在下溢区中的数

浮点数范围比定点数大，但数的个数没变多，故数之间更稀疏，且不均匀

浮点数的表示

° Normal format（规格化数形式）：

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

为了能表示更多有效数字，通常规定规格化数的小数点前为1！

°32-bit 规格化数：

31 ... 0

S	Exponent	Significand
1 bit	? bits	? bits

S 是符号位（Sign）

Exponent用移码（增码）来表示

Significand 表示 xxxxxxxxxxxxxx（部分尾数）

(基可以是 2/ 4 / 8 / 16，约定信息，无需显式表示）

°早期的计算机，各自定义自己的浮点数格式

问题：浮点数表示不统一会带来什么问题？

“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，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

现在所有通用计算机都采用IEEE 754来表示浮点数

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

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

Prof. William Kahan

IEEE 754 标准

规格化数：+/-1.xxxxxxxxxx$_{two}$ x $2^{Exponent}$

规定：小数点前总是“1”，故可隐含表示。

Single Precision（单精度）：

S	Exponent	Significand
1 bit	8 bits	23 bits

- Sign bit: 1 表示negative；0表示 positive
- Exponent（阶码）：

  全0和全1用来表示特殊值！

  - SP规格化阶码范围为0000 0001 (-126) ~ 1111 1110 (127)
  - bias为127 (single), 1023 (double)

  为什么用127？若用128, 则阶码范围为多少？

  0000 0001 (-127) ~ 1111 1110 (126)

- Significand（部分尾数）：
  - 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）

SP: $(-1)^{S}$ x (1 + Significand) x $2^{(Exponent-127)}$

DP: $(-1)^{S}$ x (1 + Significand) x $2^{(Exponent-1023)}$

举例：机器数转换为真值

已知float型变量x的机器数为BEE00000H，求x的值是多少？

$\boxed{1}\boxed{011\ 11101}\boxed{110\ 0000\ 0000\ 0000\ 0000\ 0000}$

$$(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$$

- **数符**: 1 （负数）
- **阶（指数）**：
  - **阶码**: 0111 1101B = 125
  - **阶码的值**: 125 - 127 = -2
- **尾数数值部分**:

  $1 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 0\times2^{-4} + 0\times2^{-5} + \ldots$

  $=1+2^{-1} +2^{-2} = 1+0.5 +0.25 = 1.75$
- **真值**: $-1.75\times2^{-2} = -0.4375$

为避免混淆，用**阶码**表示**阶**的编码，用**阶**或**指数**表示阶码的值

举例：真值转换为机器数

已知float型变量x的值为-12.75，求x的机器数是多少？

-12.75=-1100.11B

=-1.10011B x 2^3 **阶（指数）为3**

因此，符号 S=1

阶码 E=127+3=128+2=1000 0010

显式表示的部分尾数 Significant

= 100 1100 0000 0000 0000 0000

x 的机器数表示为：

1	1000 0010	100 1100 0000 0000 0000 0000

转换为十六进制表示为：C14C0000H

规格化数（Normalized numbers）

前面的定义是针对规格化形式（normalized form）的数

那么，其他形式的机器数表示什么样的信息呢？

Exponent	Significand	
1-254	任意 小数点前隐含1	规格化形式
0 (全0)	0	?
0 (全0)	nonzero	?
255（全1）	0	?
255（全1）	nonzero	?

0的机器数表示

How to represent 0?

exponent: all zeros

significand: all zeros

What about sign? Both cases valid.

+0: 0 00000000 00000000000000000000000

-0: 1 00000000 00000000000000000000000

Single Precision（单精度）：

S	Exponent	Significand
1 bit	8 bits	23 bits

+∞/-∞的机器数表示

浮点数除0的结果是 +/- ∞, 而不是溢出异常.（整数除0为异常）

为什么要这样处理? ∞ : infinity

- **可以利用+∞/-∞作比较。 例如：X/0>Y可作为有效比较**

How to represent +∞/-∞?

- **Exponent : all ones (11111111B = 255)**
- **Significand: all zeros**

+∞ : 0 11111111 00000000000000000000000

-∞ : 1 11111111 00000000000000000000000

相关操作：

5.0 / 0 = +∞, -5.0 / 0 = -∞

5+(+∞) = +∞, (+∞)+(+∞) = +∞

5 - (+∞) = -∞, (-∞) - (+∞) = -∞ etc

“非数”的表示

Sqrt (- 4.0) = ?   0/0 = ?

- 称为 Not a Number (NaN) - “非数”

How to represent NaN

Exponent = 255

Significand: nonzero

NaNs 可以帮助调试程序

相关操作：

sqrt (-4.0) = NaN   0/0 = NaN

op (NaN,x) = NaN   +∞+(-∞) = NaN

+∞- (+∞) = NaN   ∞/∞ = NaN

etc.

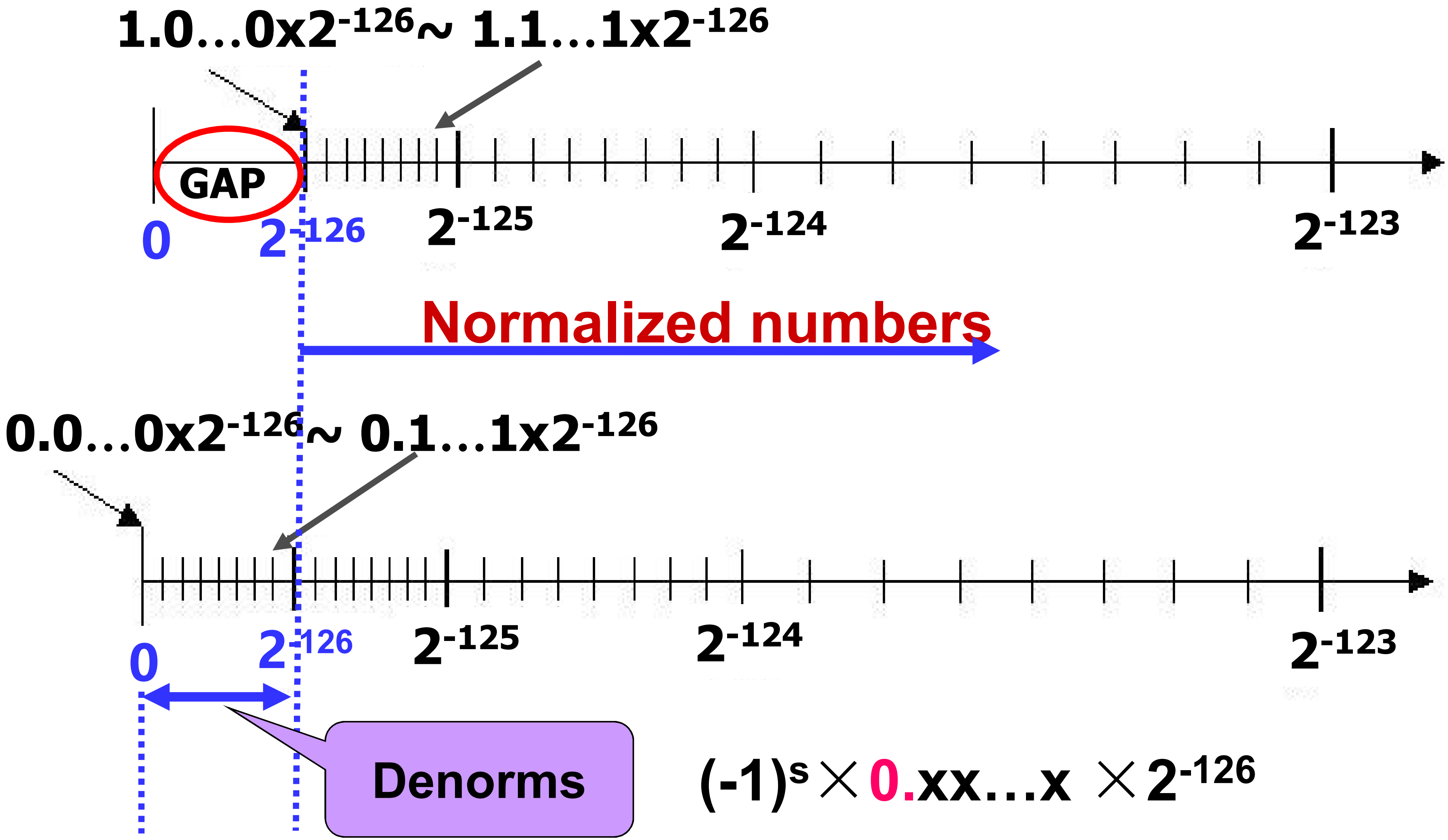
非规格化数(Denorms)的表示

对于单精度FP，还有一种情况没有定义

Exponent	Significand	
0	0	+/-0
0	nonzero	Denorms
1-254	任意 小数点前隐含1	Norms
255	0	+/- infinity
255	nonzero	NaN

用来表示
非规格化数

非规格化数(Denorms)的表示



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

关于浮点数精度的一个例子

```
#include <iostream>
using namespace std;
int  main()
{
    float   heads;
    cout.setf(ios::fixed,ios::floatfield);
    while(1)
    {
    cout << "Please   enter   a   number: ";
    cin>> heads;

}
```

61.419998和61.420002是两个可表示数，两者之间相差0.000004。当输入数据是一个不可表示数时，机器将其转换为最邻近的可表示数。

运行结果:

```
Please enter a number: 61.419997
61.419998
Please enter a number: 61.419998
61.419998
Please enter a number: 61.419999
61.419998
Please enter a number: 61.42
61.419998
Please enter a number: 61.420001
61.420002
Please enter a number:
```