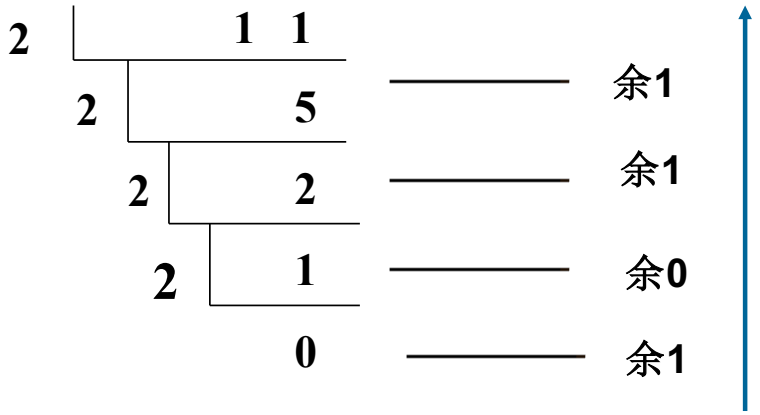
2. 数据的机器级表示与处理

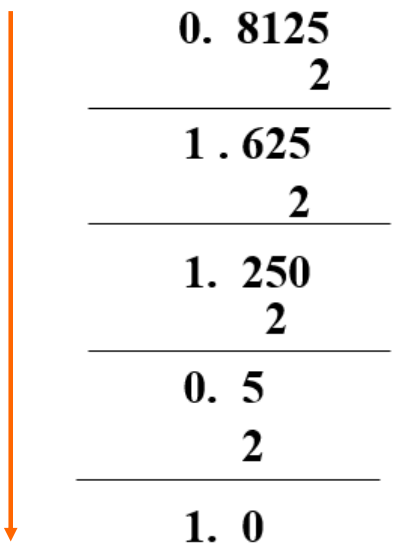
2.1. 整数和字符的表示
2.2. 数据的转换和运算
2.3. 浮点数表示和运算

浮点数的表示

- $(11.8125)_{10}=(1011.1101)_2$



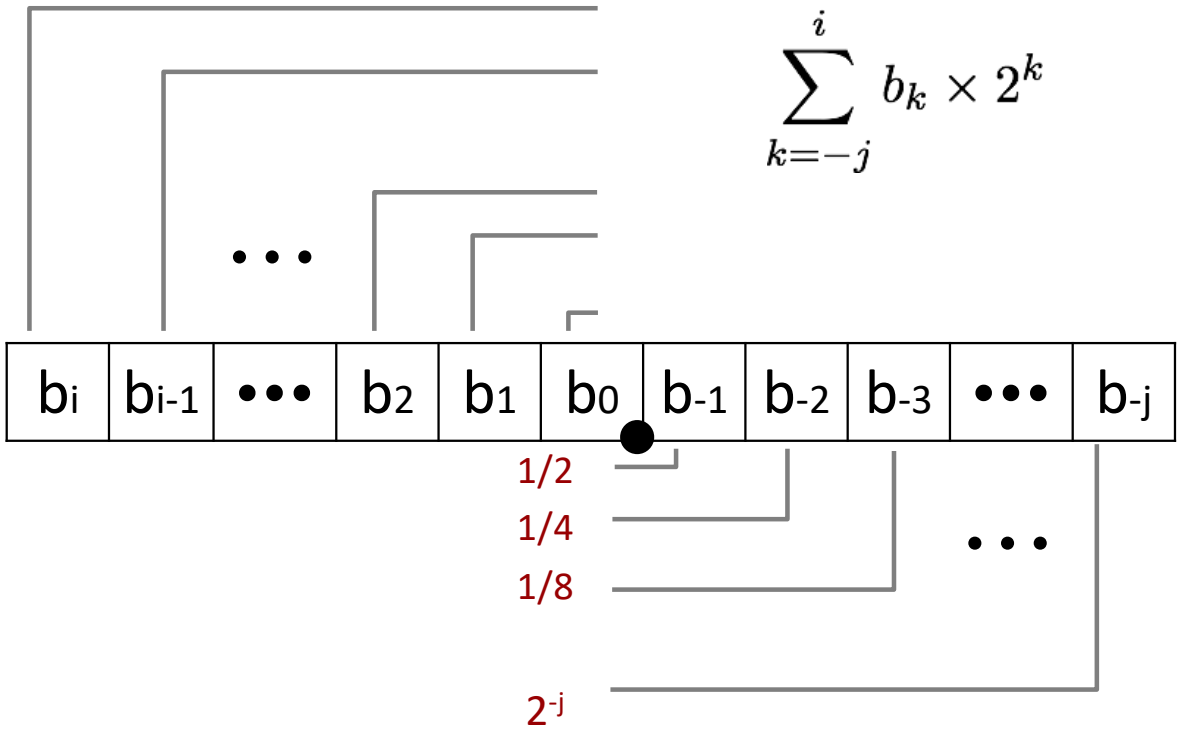
十进制整数→二进制数
除2，直到商0，余数倒排



十进制数纯小数→二进制数
乘2取整，直到乘积的小数部分为0，顺排

浮点数的表示

- $(1011.1101)_2$



浮点数的表示

- IEEE 754 标准，1985
  - 解决了此前不同机器浮点数格式不统一带来的麻烦
  - 由UCB数学系教授William Kahan独立设计，获得了1989年图灵奖

Prof. Kahan

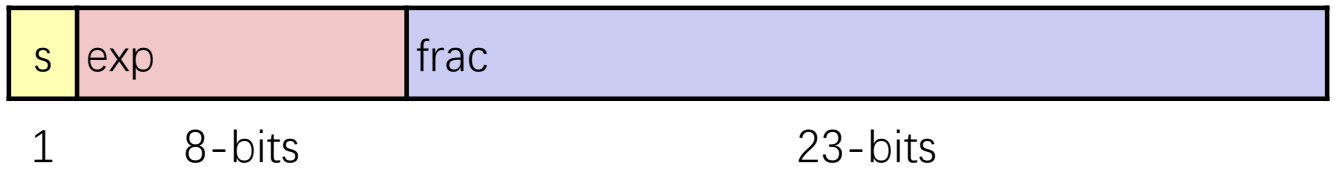
IEEE 754

- $(11.8125)_{10}=(1011.1101)_2=+1.0111101*2^3$

符号位s　　指数位exp　　尾数位frac

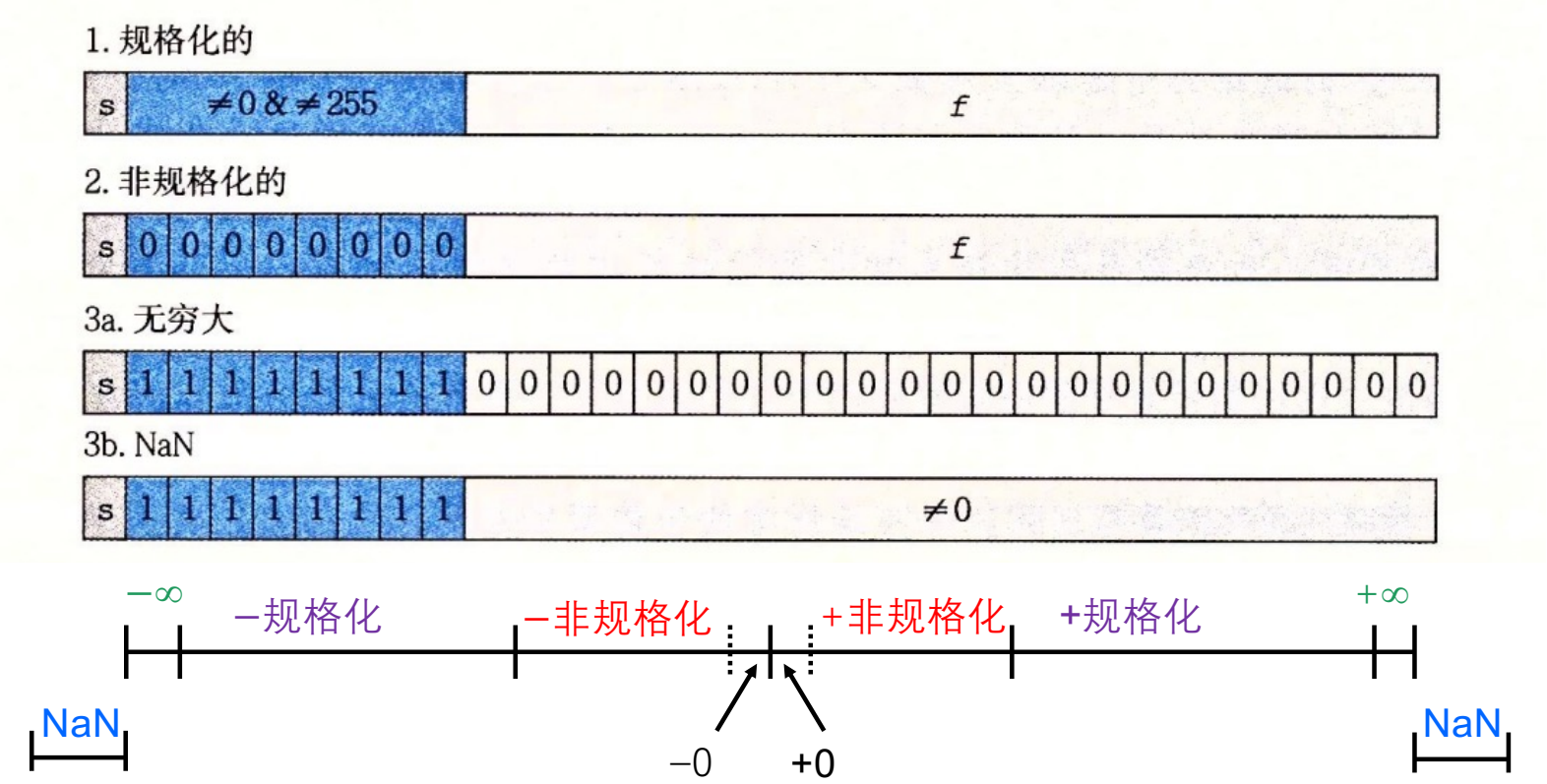
IEEE 754

- 单精度浮点数（C语言float类型）



- 双精度浮点数（C语言double类型）

IEEE 754


1. 规格化的
s
≠0 & ≠255
f
2. 非规格化的
s
0 0 0 0 0 0 0 0
f
3a. 无穷大
s
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 0 0
3b. NaN
s
1 1 1 1 1 1 1 1
≠0
−∞
−规格化
−非规格化
+非规格化
+规格化
+∞
NaN
−0
+0
NaN

规格化数

- $(-1)^s \times 1.frac \times 2^{\exp-b}$
  - e（实际指数）= exp（指数位）- b（偏移量）
  - b=2^{k-1}-1=127/1023，k为指数位数
  - 尾数最高位始终为1，省略
- 单精度浮点数：0 10000010 11000000000000000000000
  - 符号位 s=0，正数
  - 指数位 exp=10000010=130 D，e=130-(2^7-1)=3
  - 尾数 frac=11000000000000000000000
  - $(-1)^s \times 1.frac \times 2^{\exp-b}$=$(-1)^0 \times 1.110\ldots0 \times 2^3$=1 × 1.75 × 8=14.0
  - float x=14.0;

规格化数：练习

- float f=15213.0;

已知15213 D=11101101101101 B。写出f的二进制和十六进制机器数。

解：

15213=11101101101101=1.1101101101101*2^{13}

s=0

exp=e+b=13+2^7-1=13+127=140=10001100

frac=1101101101101

f的二进制机器数为：0 100,0110,0 110,1101,1011,0100,0000,0000

十六进制为：　　0x466DB400

非规格数

- 解决规格化数无法表示非常接近0的浮点数的问题
- $(-1)^s \times 0.frac \times 2^{1-b}$
  - exp（指数位）全0
  - e（实际偏移）=1-b=1-(2^{k-1}-1)=2-2^{k-1}=-126/-1022
  - 尾数最高位始终为0，省略
- 单精度浮点数：0 00000000 00000000000000000000001
  - 符号位 s=0，正数
  - 指数位 exp=00000000，e=1-(2^7-1)=-126
  - 尾数 frac=00000000000000000000001
  - $(-1)^s \times 0.frac \times 2^{1-b}=(-1)^0 \times 0.00000000000000000000001 \times 2^{-126}$
    $=1 \times 2^{-23} \times 2^{-126}=2^{-149} \approx 1.4 \times 10^{-45}$

特殊值

- 整数除0：异常
- 浮点数除0：+/- ∞
  - x/0 > y 是有效比较
- exp全1
  - frac全0
    - 1.0/0.0=-1.0/-0.0=+∞
    - -1.0/0.0=1.0/-0.0=-∞
  - frac不是全0：NaN（Not a Number）
    - sqrt(-1)，∞-∞，∞*0

总结：浮点数表示

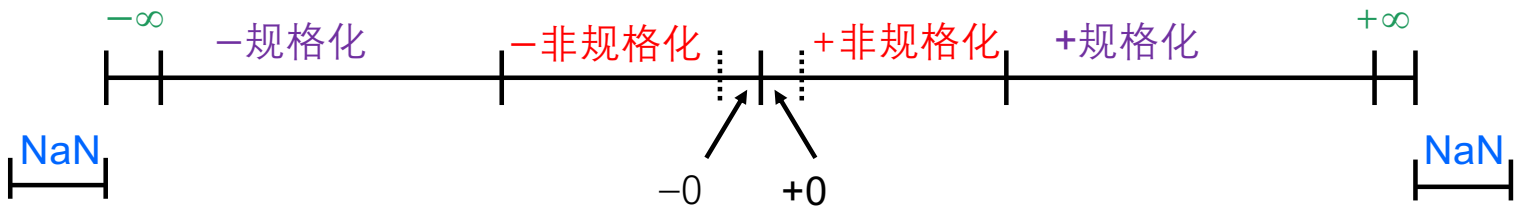
- 机器码：s exp frac
  - 单精度：1 8 23
  - 双精度：1 11 52
- 真值
  - 规格化的：$(-1)^s \times 1.frac \times 2^{\exp-b}$
    - exp不是全0或全1
  - 非规格化的：$(-1)^s \times 0.frac \times 2^{1-b}$
    - exp全0
  - b=2^{k-1}-1，k=8/11，b=127/1023，1-b=-126/-1022
  - ∞
    - exp全1，frac全0
  - NaN
    - exp全1，frac不是全0

浮点数表示范围

描　述	exp	frac	单精度	
			值	十进制
0	00 ··· 00	0 ··· 00	0	0.0
最小非规格化数	00 ··· 00	0 ··· 01	$2^{-23} \times 2^{-126}$	1.4×10^{-45}
最大非规格化数	00 ··· 00	1 ··· 11	$(1-\varepsilon) \times 2^{-126}$	1.2×10^{-38}
最小规格化数	00 ··· 01	0 ··· 00	1×2^{-126}	1.2×10^{-38}
1	01 ··· 11	0 ··· 00	1×2^{0}	1.0
最大规格化数	11 ··· 10	1 ··· 11	$(2-\varepsilon) \times 2^{127}$	3.4×10^{38}

$$S_n = a * \frac{1-r^n}{1-r}$$

$$\varepsilon = 2^{-23}$$



浮点数实例

不是所有小数都可以用浮点数表示，
不可表示数会被转换为最近的可表示数（舍入）。

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

    return 0;
}
```

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:
```

浮点数运算

- 类比：十进制科学计数法
  - 加减 $1.123 \times 10^5 + 2.560 \times 10^2$

    $= 1.123 \times 10^5 + 0.002560 \times 10^5$

    $= (1.123 + 0.00256) \times 10^5$

    $= 1.12556 \times 10^5$
  - 乘除 $1.5 * 10^2 * 2 * 10^1$

    $= (1.5 * 2) * 10^{2+1}$

    $= 3 * 10^3$

  尾数相乘除，阶码相加减

浮点数加减运算

0.5 – 0.4375

$0.5=1.000*2^{-1}$，$0.4375=0.25+0.125+0.0625=0.0111$ $B=1.110*2^{-2}$,

$1.000*2^{-1}-1.110*2^{-2}$

1. 对阶：比较两个浮点数的指数，将指数较小的浮点数的尾数右移，同时增加其指数，直到两个数的指数相等。

$1.110*2^{-2}=0.1110*2^{-1}$

2. 尾数加减：将对阶后的尾数按符号位进行加法或减法运算。

$(1.000-0.1110)*2^{-1}=0.001*2^{-1}$

浮点数加减运算

3. 规格化：如果结果的尾数不是规格化形式，则需要左移或右移尾数，并调整指数。
   - 左规（尾数高位为0）：尾数左移1位，指数减1，检查溢出，重复直到规格化或指数全0（实际指数-126）

   $0.001*2^{-1}=0.01*2^{-2}=0.1*2^{-3}=1.0*2^{-4}$
   - 右规（尾数高位进位）：尾数右移1位，指数加1，检查溢出

   $1.0 + 1.0 = 1.00 * 2^0 + 1.00 * 2^0 = 10.00 * 2^0 = 1.000 * 2^1$
4. 舍入：当尾数比规定位数长时，需要舍入（向0或向最近偶数）。
5. 检查溢出：检查指数，如果超过最大值，结果为∞，如果小于最小值，结果为非格式化数或0。

0.5 – 0.4375=0.0625

浮点数舍入

1. 向0舍入
   - 直接截断多余的有效位
2. 向正无穷舍入
   - 正数进位
   - 负数截断
3. 向负无穷舍入
   - 正数截断
   - 负数进位
4. 向最近偶数舍入（默认）
   - 统计偏差更小

浮点数舍入

- 向最近偶数舍入（默认）

  1.1101，1.110101，1.110110，1.110111，1.1110

  - 更靠近某个值时，舍入到更靠近的值
    - Round(1.110101)=1.1101
    - Round(1.110111)=1.1110
  - 位于两个值中间时，舍入到偶数值（尾数最低有效位为0）
    - Round(1.110110)=1.1110

浮点数舍入实例

```
float a;
double b;
a = 123456.789e4;
b = 123456.789e4;
printf(“%f/n%f/n”, a, b);
```

输出:

1234567936.000000

1234567890.000000

float可以精确表示7个十进制有效数位,

后面的数位是舍入后的结果,

舍入后的值可能更大，也可能更小。

C语言中的浮点数类型

- float：IEEE 754中的单精度浮点数
- double：IEEE 754中的双精度浮点数

- int转float：不会溢出，可能会舍入
- int/float转double：保留精确值
- double转int/float：可能溢出，可能舍入
- float/double转int：可能截断，向0截断
  - (int)3.14=3, (int)-3.14=-3
  - 超出范围int范围时是未定义行为

浮点数比较运算

`int x; float f; double d;` f和d都不是NaN。判断以下关系表达式是否永真。

- `x == (int)(float) x`
  `false`
- `x == (int)(double) x`
  `true`
- `f == (float)(double) f`
  `true`
- `d == (float) d`
  `false`
- `f == -(-f)`
  `true`
- `2/3 == 2/3.0`
  `false`
- `d < 0.0 → ((d*2) < 0.0)`
  `true`
- `d > f → -f > -d`
  `true`
- `d * d >= 0.0`
  `true`
- `x*x >= 0`
  `false`
- `(d+f)-d == f`
  `false`

A→B 等价于 !A || B

浮点数加减的注意事项

例：$x = -1.5x10^{38}$, $y = 1.5x10^{38}$, $z = 1.0$

$(x+y)+z = (-1.5x10^{38} + 1.5x10^{38}) + 1.0 = 1.0$

$x+(y+z) = -1.5x10^{38} + (1.5x10^{38} + 1.0) = 0.0$

- 结合律不成立
- 大数吃小数：指数相差很大时，较小的数在对阶过程中被忽略

实例：Ariana 5火箭爆炸

- 1996年6月4日，欧洲空间局的运载火箭Ariana 5在发射37秒钟后偏离飞行路线，随后解体爆炸，火箭上载有价值5亿美元的通信卫星。
- 火箭的水平速率：64位浮点数
  - Ariana 4火箭软件：确认安全后保存为16位有符号整数进行运算
  - Ariana 5火箭：最高速度提升5倍，在将64位浮点数转换为16位有符号整数时产生溢出

数据的机器级表示与处理

- 在机器内部编码后的数称为机器数，其值称为真值
- 整数的表示（补码）
  - 无符号整数：正整数，用来表示地址等，补码等于原码
  - 有符号整数：负数的补码表示
- C语言中的整数
  - 无符号数：unsigned int (short / long)
  - 有符号数：int (short / long)

数据的机器级表示与处理

- 浮点数的表示
  - 符号、尾数、指数（阶）
- 浮点数的精度：与尾数的位数和是否规格化有关
- 浮点数的表示（IEEE 754标准）：单精度SP（float）和双精度DP（double）
    - 规格化数：阶非全0非全1，尾数最高位隐含为1
    - 非规格化数：阶为全0，尾为非0，尾数最高位隐含为0
    - “零”：阶为全0，尾为全0
    - ∞：阶为全1，尾为全0
    - NaN：阶为全1，尾为非0

数据的机器级表示与处理（运算）

- C语言中涉及的运算
  - 整数算术运算、浮点数算术运算
  - 按位、逻辑、移位、位扩展和位截断
- 整数的加、减运算
  - 有、无符号整数的加、减是模运算（高位丢弃、用标志位表示）
  - 现实与计算机中的运算结果有差异
- 整数的乘、除运算
  - 无符号整数：左移k位等于乘2^k、逻辑右移k位等于除2^k
  - 有符号整数乘：左移k位等于乘2^k
  - 有符号整数除：(x>=0？x：x+2^k-1）算术右移k位，等于x除以2^k
- 浮点数运算
  - 加减：对阶/尾数加减/规格化/舍入(就近舍入到偶数)（大数吃小数）
  - 乘除：尾数相乘除，阶码相加减