2.6 浮点数

2.6 浮点数

01 浮点数的表示与IEEE754

02 浮点数的加减

03 程序员如何看浮点运算

04 浮点运算中的误差传播

2.6.1、二进制表示浮点数

十进制科学计数法：

1.2345×10^{20}，12.345×10^{19}，0.497×10^{-50} ，4.97×10^{-51}

二进制小数和十进制数转换

$0.1_2 = 1/2 = 2^{-1} = 0.5_{10}$

$0.01_2 = 1/2^2 = 2^{-2} = 0.25_{10}$

$0.001_2 = 1/2^3 = 2^{-3} = 0.125_{10}$

$0.11_2 = 2^{-1} + 2^{-2} = 0.75_{10}$

$0.011_2 = 2^{-2} + 2^{-3} = 0.375_{10}$

2.6.1、二进制与十进制表示的常见浮点数

0.1	2^{-1}	0.5
0.01	2^{-2}	0.25
0.001	2^{-3}	0.125
0.0001	2^{-4}	0.0625
0.00001	2^{-5}	0.03125
0.000001	2^{-6}	0.015625
0.0000001	2^{-7}	0.0078125

2.6.1、十进制转二进制

例1：0.8125_{10} 转二进制是多少？

答：0.1101_2

例2：0.8_{10} 转二进制是多少？

答：$0.11001100110011\cdots_2$

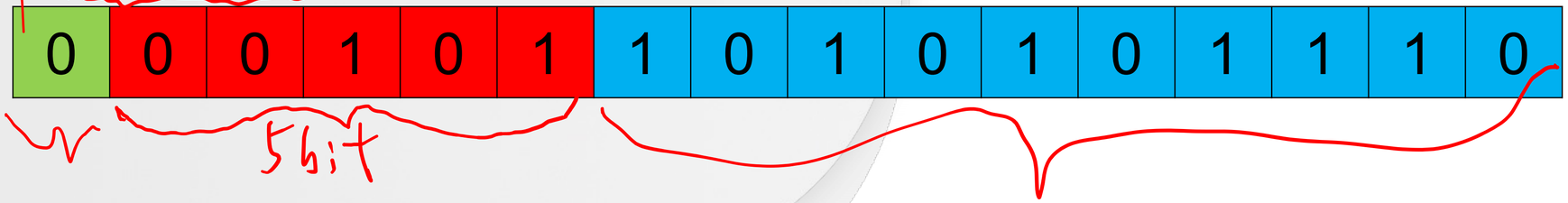
2.6.1、二进制表示浮点数

二进制科学计数法：
101010.11101
可以表示为：
1.0101011101×2^5

浮点数其实是两个值的乘积，而且表示并不唯一。

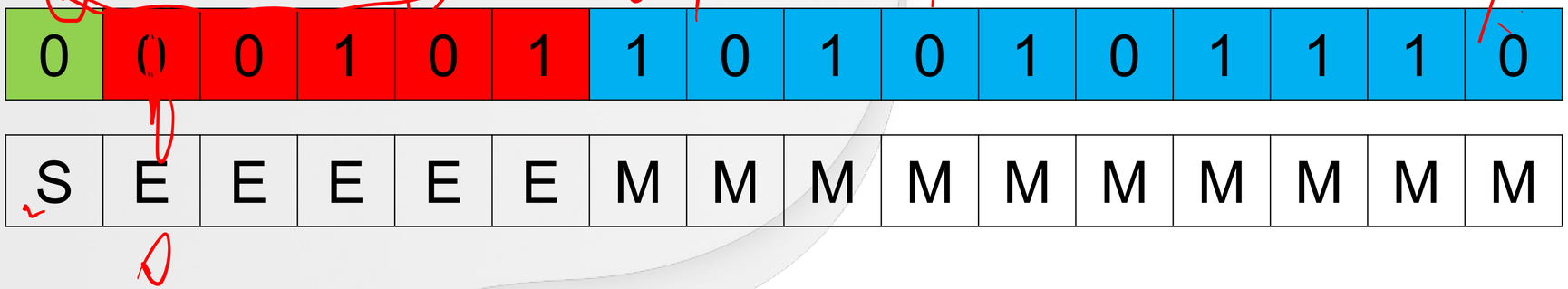
2.6.1、二进制保存浮点数

$1.0101011101 \times 2^{5}$



2.6.1、二进制保存浮点数

1.0101011101×2^5



几个问题：

1、指数要能表示正负，用补码？

2、尾数这里能不能优化？

3、这个数给别人，能不能还原？

2.6.1、二进制保存浮点数

$1.0101011101 \times 2^{5}$

0	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0

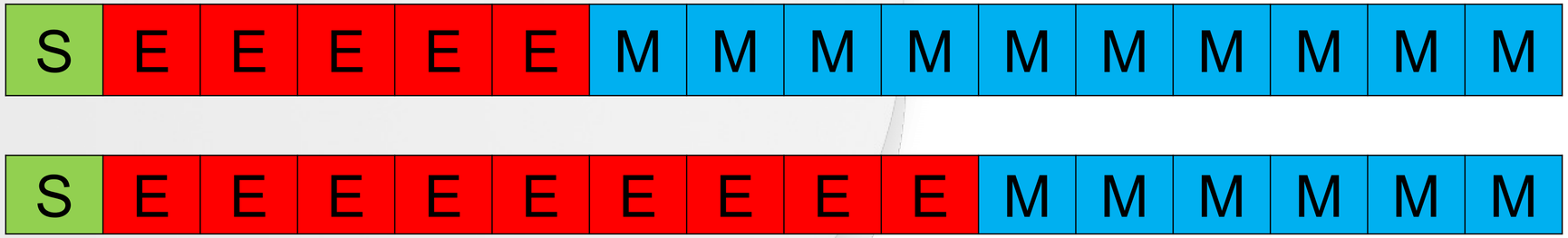
S	E	E	E	E	E	E	E	E	M	M	M	M	M	M	M

8 bit

7 bit

0.101110×2^{45}

2.6.1、二进制保存浮点数



1、指数位越多，尾数位则越少，其表示的范围越大，但精度就会越低

2、指数位越少，尾数位则越多，其表示的范围越小，但精度就会越高

2.6.1、IEEE754浮点数标准

三种浮点数表示：
1、32位单精度浮点数
2、64位双精度浮点数
3、临时实数（不作要求）

float
double

2.6.1、两个关键词

规格化浮点数：小数点前面必须是1，左规(x2)，右规(÷2)

$101010.11101 = 1.0101011101 \times 2^{5}$

$0.00101010111101 = 1.0101011101 \times 2^{-3}$

2.6.1、关于规格化的几个特殊值

1、当尾数为-1/2时，尾数的补码为11.100…0. 虽然这在1~1/2区间，但是不满足补码的规格化形式，因而不是规格化数。

2、当尾数为-1时，尾数的补码是11.00…0, 因为小数补码允许表示-1，所以特别规定-1为规格化数。

3、0的尾数是全0.

2.6.1、两个关键词

偏置指数：$2^{e-1}-1$,e为存储指数的比特长度

2.6.1、移码

补码的缺点：在比较大小时，因为有符号位，导致运算器输出错误结果。

01011，00111，10101，11001排序

对所有值+2^n之后再比

01011 + 100000 = 101011,

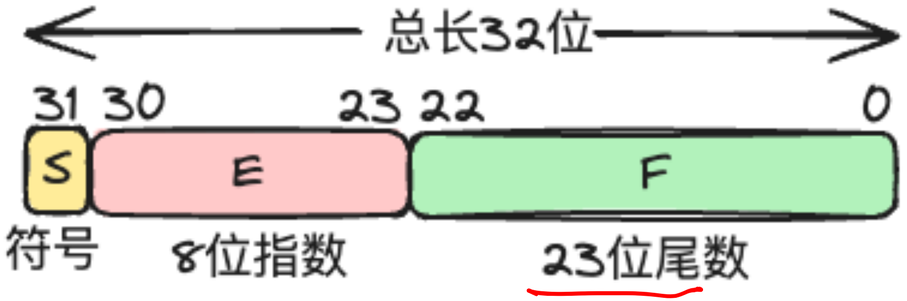
00111 + 100000 = 100111,

-11011 + 100000 = 001101,

-00111 + 100000 = 011001

移码的表示范围：0 ~ （2^{n+1}-1）

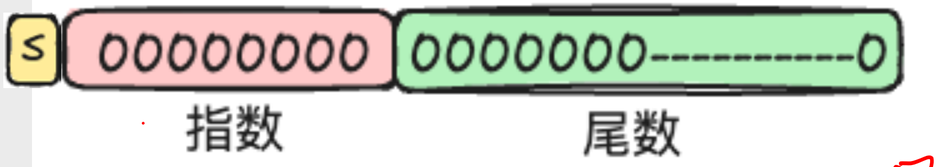
2.6.1、32位IEEE754单精度浮点数



1、尾数（底）：0、0.100…0 到 0.111…1之间

2、指数（阶码、移码表示）：存储值是真实值+偏置值（$2^{n-1}-1$）

3、0的表示：0x00000000, 0x80000000



真实值：$(-1)^S \times 1.F \times 2^e$

2.6.1、转换计算

例1：用IEEE754表示的浮点数x为41360000H，请求x的十进制表示（解压）。

例2：十进制数0.75的IEEE754单精度浮点数的二进制表示。

2.6.1、常用浮点数

	符号位	阶码	尾数	总位数	最大指数	最小指数	指数偏移量
短实数	1	8	23	32	+127	-126	+127
长实数	1	11	52	64	+1023	-1022	+1023
临时实数	1	15	64	80	+16383	-16382	+16383

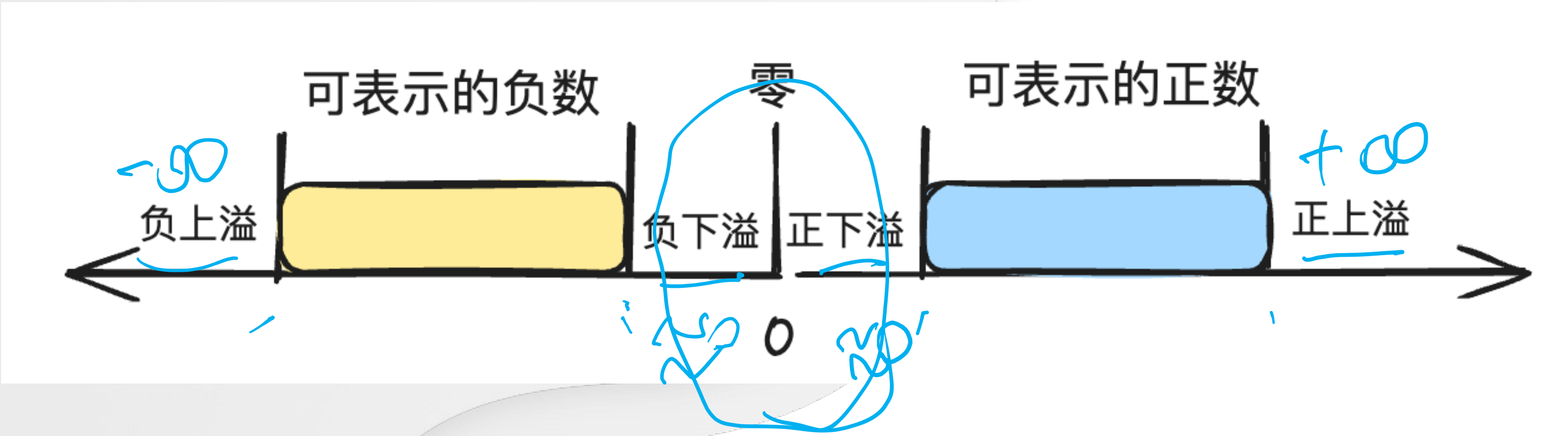
2.6.1、常用浮点数表示范围

	最小值	最大值
短实数（单精度）	E=1，M=0 $1.0\times2^{1-127}=2^{-126}$	E=254,M=.111…1 $1.111\ldots1\times2^{254-127}=2^{127}\times(2-2^{-23})$
长实数（双精度）	E=1，M=0 $1.0\times2^{1-123}=2^{-1022}$	E=2046,M=.111…1 $1.111\ldots1\times2^{2046-1023}=2^{1023}\times(2-2^{-52})$

2.6.1、阶码全0和全1的解释

	单精度	双精度
规格化：正常计算规则	S E不全为0 不全为1 M	
非规格化：表示0和很小的数	S E全为0 M	
正负无穷大	S E全为1 M全为0	
NaN	S E全为1 M不全为0	

2.6.1、常用浮点数表示范围



2.6.1、二进制保存浮点数0.3

0.0100110011001100110011001100110011001100110011001100110011

0	0	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0

默认的舍入规则:
舍入到最接近，一样接近的情况下偶数优先.

	f = 有效数位																						
舍入后 →	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0
舍入前 →	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

2.6.1、IEEE754中定义的4种舍入

- **舍入到最接近**：舍入到最接近，在一样接近的情况下偶数优先。
- **朝 +∞ 方向舍入**：会将结果朝正无限大的方向舍入。
- **朝 -∞ 方向舍入**：会将结果朝负无限大的方向舍入。
- **朝 0 方向舍入**：会将结果朝 0 的方向舍入。

2.6.2、浮点数的加减

例1： $1.2\times10^5+4.6\times10^3$

$1.2\times10^5+4.6\times10^3=(1.2+4.6)\times10^5=1.246\times10^5$

$1.2\times10^5+0.046\times10^5=(1.2+0.046)\times10^5=1.246\times10^5$

2.6.2、浮点数的加减

步骤：

1、对阶（小数点对齐，阶码相等）：低阶向高阶对齐

2、尾数求和（差）是在对阶后进行

3、规格化处理

4、舍入

5、溢出判断：浮点数的溢出与否是由阶码的符号决定的

-126　　127

2.6.2、浮点数的加减

例1：两个浮点数x = 1.01101x2^5 ，y = 1.10011x2^3，用单精度浮点数求x+y和x-y的值（结果保留3位小数位）。

x阶码5，y阶码3，

$1.01101 \times 2^5 + 0.011\,0011 \times 2^5$

$+\ 0.011\,0011$

$1.101\,000 \times 2^5$

$x+y = 1.11 \times 2^5$

$1.0110\,100$

$-\ 0.011\,0011$

$1.000\,0001 \times 2^5$

$x-y = 1.00 \times 2^5$

2.6.3、程序员如何看浮点数

因为精度损失的存在，会出现如下情况：

1、浮点加法不具有结合性，(x+y)+z 和 x + (y + z)不一定相等

2、$x^2 - y^2$ 和 (x+y)*(x-y)

$x = 5.9995998 \times 10^2$

$y = 1.000201 \times 10^{-1}$

$x^2 = 35.995197760160004 \times 10^4$

$= 3.5995198 \times 10^5$

$y^2 = 1.000402040401 \times 10^{-2}$

$= 1.000402 \times 10^{-2}$

$x^2 - y^2 =$

3.5995198×10^5

$-0.0000001000402 \times 10^5$

$3.5995196999598 \times 10^5$

$x = 5.9995998 \times 10^2$

$y = 1.000201 \times 10^{-1}$

$x + y =$

5.9995998×10^2

$+ 0.0001000201 \times 10^2$

6.0006000201×10^2

$x - y =$

5.9995998×10^2

$- 0.0001000201 \times 10^2$

5.9985995799×10^2

$\times \quad 6.0006000201 \times 10^2$

$3.5995196759976 \times 10^5$

2.6.3、程序员如何看浮点数

因为精度损失的存在，会出现如下情况：

1、浮点加法不具有结合性，(x+y)+x 和 x + (y + z)不一定相等

2、$x^2 - y^2$ 和 (x+y)*(x-y)不一定相等

3、在C语言中，float和double对应单精度和双精度浮点数，而long double类型刚对应IEEE754中的临时（扩展）浮点数。

3.1、类型转换：char → int → long → double, float → double

3.2、int → float 转换有可能会发生精度损失,int → double 不会

3.3、double → float 可能会发生溢出，精度损失。

3.4、float → int, double → int 小数部分全部舍入。

2.6.4、浮点运算中的误差

1、乘法的相对误差比加法大：

2、两个几乎相等的数相减，引起有效位变少：

3、两个浮点数判断相等时，不能直接使用 == 运算符，
要使用abs求出绝对值后判断是否小于一个很小的常数。

2.6. 本节总结

1. 浮点数一般用科学计数法表示
2. 把科学计数法中的变量，填充到固定 bit 中，即是浮点数的结果
3. 浮点数规则不统一，导致同一个数字的浮点数表示各不相同，在计算时还需要先进行转换才能进行计算
4. IEEE 组织统一浮点数标准，规定了单精度浮点数 float 和双精度浮点数 double

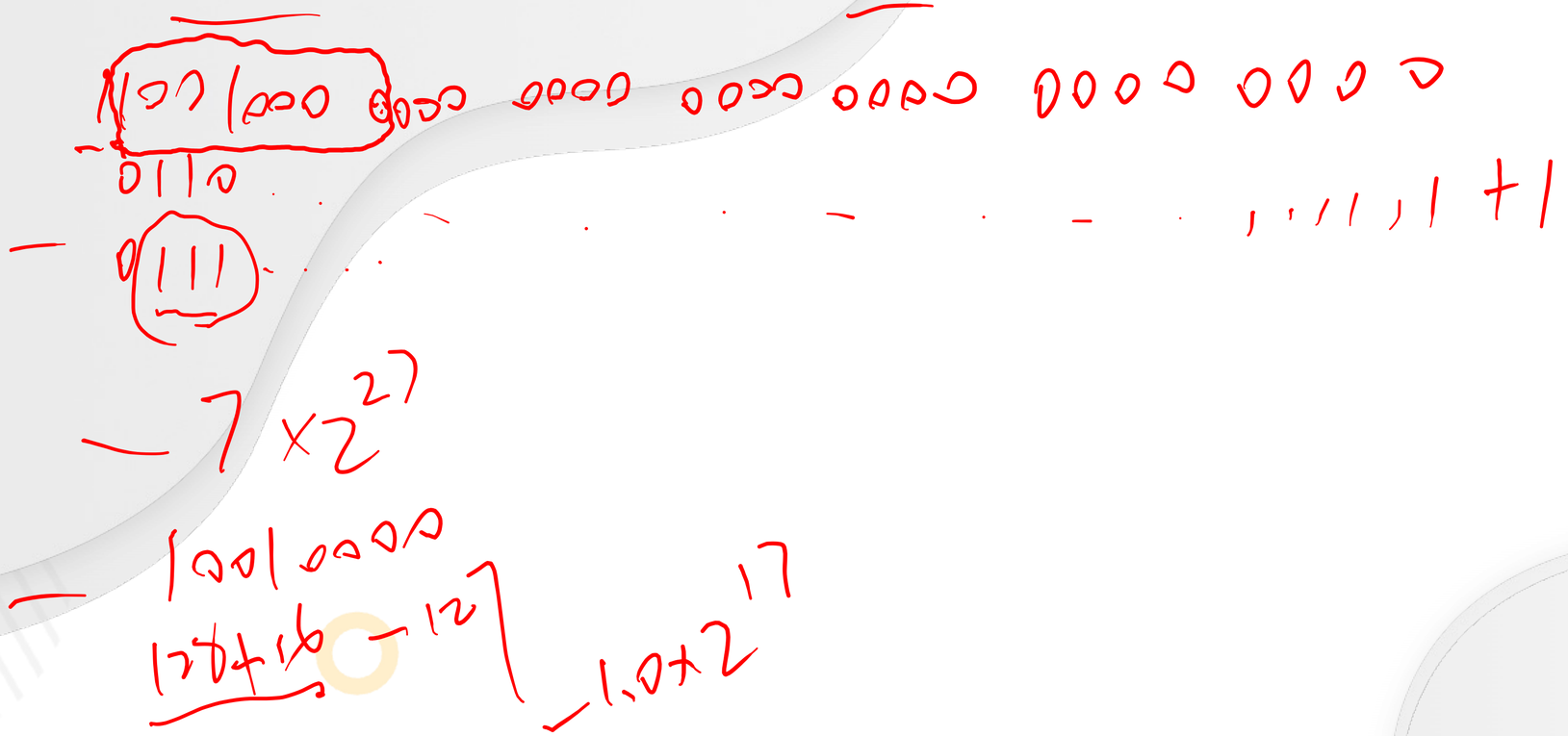
2.6. 本节总结

5. 浮点数在表示小数时，由于十进制小数在转换为二进制时，存在无法精确转换的情况，而在固定 bit 的计算机中存储时会被截断，所以浮点数表示小数可能存在精度损失
6. 浮点数在表示一个数字时，其范围和精度非常大，所以我们平时使用的小数，在计算机中通常用浮点数来存储
7. 浮点数加减时的5个步骤，乘除法参考十进制运算，更好处理。
8. 浮点数的尾数不够表示会进行舍入，这不是溢出，如果指数超过范围才会发生溢出。
9. 因为精度有限导致存储的浮点数和真实值之间有误差，要尽可能减小这种误差
10. 在程序处理中，发生类型转换时也会出现精度损失和误差

2.6.4、注意抓住关键信息

【2020统考真题】已知带符号整数用补码表示，float型数用IEEE754标准表示，假定变量x的类型只可能是int或float，当x的机器数为C800 0000H时，x的值可能是（）

A. -7×2^{27}　　B. -2^{16}　　C. 2^{17}　　D. 25×2^{27}



海贼宝藏
专注IT教育在线学习平台

欢迎参与学习

WELCOME FOR YOUR JOINING

船说：计算机基础