Bits, Bytes, Integers, and Floating Point

杨斯琪 刘昕垚

Bits and Bytes

- bit：计算机内存中的最小单位。用0或1表示
- Byte：最小的寻址内存单位

- Byte=8bits（B=8b）

Word Size

- to计算机：虚拟地址空间的最大大小
- to程序：如何编译
  - 64位机器可以运行32位机器编译的程序

TIPS:常见：short：2字节

int：4字节

float：4字节

double：8字节

word（字）：标识ISA处理数据单元的单位，32位为4Byte，64位为8Byte。

Byte Ordering

- 字节序列前面的是低地址
- 大端：最高有效字节在低地址
  - eg： Sun, PPC Mac, Internet
- 小端：最低有效字节在低地址
  - eg：x86, ARM processors running Android, iOS, and Windows

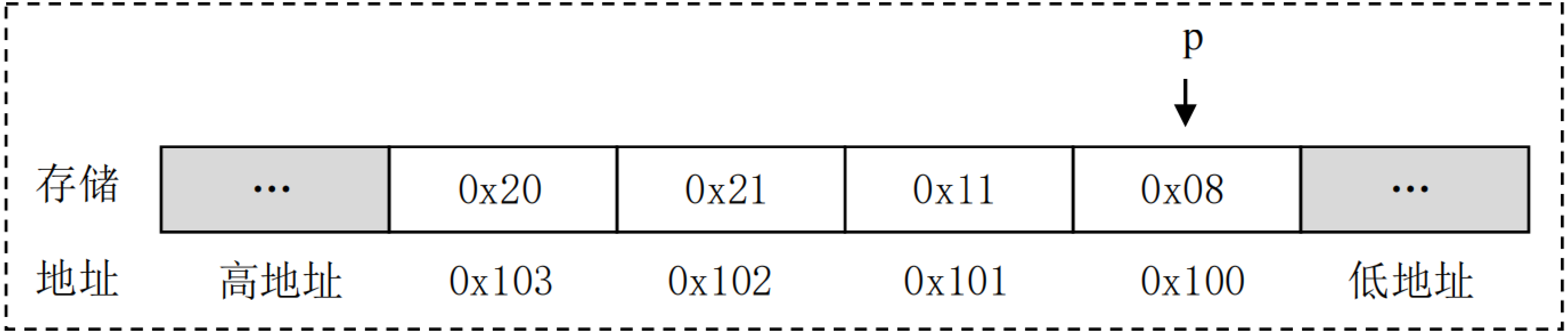
TIPS：

- 1、只是存储形式的字节顺序，具体运算及操作仍然按照语义合理性展开；
  遇到困惑时可以从“小端机器也需要合理运行”角度思考。
- 2、特殊情况下强制要求大小端机器的字节顺序保持同一性，后面例题中可以看到某个特例。

字节顺序例题

(　　)11. 已知在x86-64架构下, 0x100到0x103的字节存储如下图所示. 假设指针p一开始指向地址0x100处, 类型为 “short *” . 则当执行 “p++;” 的指令后, p所指向的短整型(short)的值变为 (　　　　).

A. 0x2021　　B. 0x2120　　C. 0x2111　　D. 0x1121



- 答案： A
- 解析：
    - 指针++就是按照指针类型的大小，前进一个类型的大小
    - 由于x86是小端，所以读取顺序是0x103，0x102、因此为0x2021.

Bit-level Manipulations

- & | ~ ^
- 异或性质的范例：void XOR(int& x, int& y) {

```
    y = x ^ y;
    x = x ^ y;
    y = x ^ y;
    printf(x, y);
}
```

XOR(a,b)的输出结果为？

A) a, b B) b, a C) b,0 D) b, a^b

- 答案：B

TIPS:

- swap程序的基本形式
- 异或的特殊性质

  交换律；结合律
- 注意补码性质

Bit-level Manipulations

- &&　　||　　!
- 提前截止规则　（自己写代码&遇到类似题需注意）

- <<：左移
- >>：右移：逻辑右移
算术右移

Bit-level Manipulations

1.在 64 位机器上，判断下列等式是否恒成立

```
/* random_int()函数返回一个随机的 int 类型值 */
int x = random_int();
int y = random_int();
int z = random_int();
unsigned ux = (unsigned)x;
long lx = (long)x;  /* long 为 64 位 */
long ly = (long)y;
double dx = (double)x;
double dy = (double)y;
double dz = (double)z;
```

Expression	Always True?
(x >= 0) || (3*x < 0)	Y N
(x >= 0) || (x < ux)	Y N
((x >> 1) << 1) <= x	Y N
((x-y)<<3) + (x>>1) - y == 8*x - 9*y + x/2	Y N
(x - y > 0) == ((y+~x+1)>>31 == 1)	Y N
dx + dy == (double) (y+x)	Y N
dx + dy + dz == dz + dy + dx	Y N
(int)((lx+ly)>>1) == ((x&y) + ((x^y)>>1))	Y N

答案

N 考虑 x = (1<<31)>>1

N 考虑 x = -1

Y

N 考虑 x = -1 y = 0

N 考虑 x-y = Tmin

N 考虑 x+y 溢出

Y 恒成立,每一个 int 型都可以由一个 double 型精确表示

Y 恒成立

Integers（总位数为w）

- 无符号数的编码
- UMin = 0 （$000...0_2$）
- UMax = 2^{w-1} （$111...1_2$）

- 补码编码 $B2T(X) = -x_{w-1} \cdot 2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$
- TMin = -2^{w-1} （$100...0_2$）
- TMax = $2^{w-1} - 1$ （$011...1_2$）

- | TMin | = TMax + 1
- UMax = 2 * TMax + 1

- 2^{31}=2147483648
- -1： $111...11_2$

补充考点：
反码和原码

书p47

旁注 有符号数的其他表示方法

有符号数还有两种标准的表示方法：

反码(Ones' Complement)：除了最高有效位的权是$-(2^{w-1}-1)$而不是-2^{w-1}，它和补码是一样的：

$$B2O_w(\vec{x}) \doteq -x_{w-1}(2^{w-1}-1)+\sum_{i=0}^{w-2} x_i 2^i$$

原码(Sign-Magnitude)：最高有效位是符号位，用来确定剩下的位应该取负权还是正权：

$$B2S_w(\vec{x}) \doteq (-1)^{x_{w-1}} \cdot \left(\sum_{i=0}^{w-2} x_i 2^i\right)$$

这两种表示方法都有一个奇怪的属性，那就是对于数字 0 有两种不同的编码方式。这两种表示方法，把[00…0]都解释为+0。而值-0 在原码中表示为[10…0]，在反码中表示为[11…1]。虽然过去生产过基于反码表示的机器，但是几乎所有的现代机器都使用补码。我们将看到在浮点数中有使用原码编码。

请注意补码(Two's complement)和反码(Ones' complement)中撇号的位置是不同的。术语补码来源于这样一个情况，对于非负数 x，我们用 2^w-x(这里只有一个 2)来计算 $-x$ 的 w 位表示。术语反码来源于这样一个属性，我们用[111…1]$-x$(这里有很多个 1)来计算 $-x$ 的反码表示。

整型-补码的类型转换

- 不改变位表示，只改变映射规则：二进制表示—>十进制值

- ＜，＞，＝＝，＜＝，＞＝：

  如果一个运算数有符号，另一个无符号：隐式类型转换发生

  有符号强制转化为无符号（signed→unsigned）

  TIPS:unsigned相减结果一定非负——但这并不意味着不“溢出”！

例题

2．设整型变量采用 32 位补码表示，判断正误（填“√”或“×”）：

i．设 x,y,z 是整型变量，且 x<y<z<0，则 (-y)>(-z)>0．（____(4)____）

ii．表达式“(-5) + sizeof(int) < 0”为真．（____(5)____）

(4) √　(5) ×

1．（6 points）假设 8-bit 整数，请填写以下表格（每空 1 分）

类型	最大的整数+1	39+(-127)	39+(-127) 是否溢出?
Unsigned	二进制:	十进制:	
Two's Complement	二进制:	十进制:	

1.

类型	最大的整数+1	39+(-127)	39+(-127)是否溢出?
Unsigned	二进制：0000 0000	十进制：168	是
Two's Complement	二进制：1000 0000	十进制：-88	否

TIPS：“溢出”的定义：

完整的算数结果不能放到数据类型的字长限制中去。

人话：原生的计算结果不符合该类型应有的数据范围。

扩展与截断

- 扩展
  - 无符号：在开头添加0（零扩展）
  - 补码：扩展的每一位都复制原数符号位的值

    TIPS：该操作不改变值的大小，证明见书p55，但直观上也可以理解。

- 截断（截断为k位数字）
  - 无符号：$x' = x \bmod 2^k$
  - 补码：$x' = U2T_k(x \bmod 2^k)$

- 应用于不同数据类型-类型转换规则中

不同数据类型-类型转换规则

- short - int：直接扩展
- short - unsigned int：
  - 先改变大小
  - 后改变有无符号
  - 即：

```
short sx;
(unsigned) sx= (unsigned) (int) sx
```

四则运算

- 加：
- 无符号： For x and y such that $0 \le x, y < 2^w$:

$$x +_w^u y = \begin{cases} x + y, & x + y < 2^w \quad \text{Normal} \\ x + y - 2^w, & 2^w \le x + y < 2^{w+1} \quad \text{Overflow} \end{cases}$$

- 补码： For integer values x and y in the range $-2^{w-1} \le x, y \le 2^{w-1} - 1$:

$$x +_w^t y = \begin{cases} x + y - 2^w, & 2^{w-1} \le x + y \quad \text{Positive overflow} \\ x + y, & -2^{w-1} \le x + y < 2^{w-1} \quad \text{Normal} \\ x + y + 2^w, & x + y < -2^{w-1} \quad \text{Negative overflow} \end{cases} \tag{2.13}$$

TIPS：溢出后的处理须注意

- 乘：（截断）
  - 无符号：$x *_w^u y = (x \cdot y) \bmod 2^w$
  - 补码：$x *_w^t y = U2T_w((x \cdot y) \bmod 2^w)$
- 乘以常数：
  - u * 2^k：u << k

书上有关于

“四则运算在两种表示中具有位级等价性，即使出现溢出的意外情况”

的证明，但大概不会考（吧。

- 除：
  - 除以常数——u >> k：$\lfloor u / 2^k \rfloor$
  - 无符号：逻辑右移
  - 补码：算术右移

例题

4.阅读如下一段代码

```
int x = 0x2021;
int y = -x;
int z = (x / 2) - (x >> 1) + (y / 2) - (y >> 1);
printf("%d", z);
```

问运行这段代码后的输出是什么：

- 答案：1
- 右移向下取整，而除以 2 则是向 0 取整

即：正数右移和除以2都是向0取整，取整后x’<=x;

负数除以2向零取整，x’>=x; 右移向下取整，x’<=x

Fractional binary numbers

b_i	b_{i-1}	•••	b_2	b_1	b_0	b_{-1}	b_{-2}	b_{-3}	•••	b_{-j}

- 在十进制下表示： $\sum_{k=-j}^{i} b_k \times 2^k$
- 局限性：
  - 只能表示形如 $x/2^k$的数
  - 定点

IEEE floating point

- 标准形式：$(-1)^s \times M \times 2^E$

- 符号s：决定正负（s=1 or 0）
- 阶码E：对浮点数加权（对应exp编码部分）
- 尾数M：二进制小数。范围：1~2-ε 或 0~1-ε（对应frac编码部分）

IEEE floating point

- 规格化
  - exp位模式不全为0也不全为1，表示的无符号整数为e
  - Bias=2^{k-1}-1（k表示exp位数）
  - E=e-Bias

  - frac描述小数值f=$0.f_{n-1}f_{n-2}\cdots f_1f_0\ _{(2)}$
  - M=1+f

  - 值=$(-1)^s \times M \times 2^E$

  绝对值范围 00...01 0...00 到 11...10 1...11

IEEE floating point

- 非规格化——exp位模式全为0
  - Bias=2^{k-1}-1 （k表示exp位数）
  - E=1-Bias
  - M=f
  - 值=$(-1)^s \times M \times 2^E$
  - 表示范围：
    - 0，和非常接近0的数;
    - 可能的数值分布均匀接近0.0

TIPS：+0.0和-0.0的区别：

+0.0=0 0...00 0...00

-0.0=1 0...00 0...00

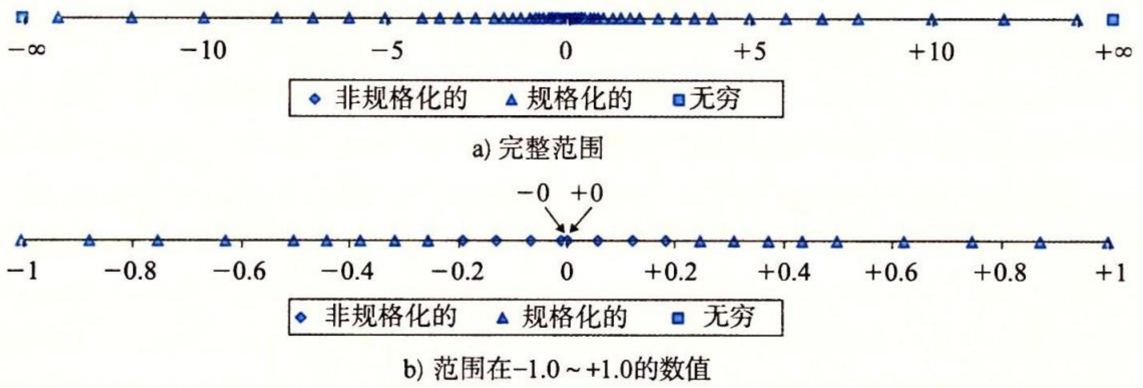


图 2-34 6 位浮点格式可表示的值（$k=3$ 的阶码位和 $n=2$ 的尾数位。偏置量是 3）

IEEE floating point

- 特殊值——exp位模式全为1
  - 小数域全为0：
    - 表示溢出--非常大的数相乘导致溢出 or 除以0
    - s=0： +∞
    - s=1： -∞
  - 小数域为非零：NaN
    - 表示无穷和实数以外的特殊结果

例题-浮点数分布

(2)考虑一种12-bit长的浮点数(符号位(s)：1-bit;阶码字段(exp)：4-bit;小数字段(frac)：7-bit),此浮点数遵循IEEE浮点数格式,则[1,2)区间中包含________个用上面规则精确表示的浮点数.

- 答案：128
- 解析：值=$(-1)^s \times M \times 2^E$

  非规格化不在[1,+∞)，考虑规格化数值;

  当且仅当E=0时成立，此时区间内浮点数个数由小数字段决定;

  2^7=128个。

Rounding

- 向偶数舍入
  - 一般向最接近舍入
  - 如果是两个可能结果中间数值：向最接近的偶数舍入
  - 二进制：最低有效位为0视为偶数

- 向零舍入
- 向下舍入
- 向上舍入

Floating Point Operations

- 加法:
  - 一般可交换，不可结合
  - $(-1)^{s1}\ M1\ 2^{E1} + (-1)^{s2}\ M2\ 2^{E2} = (-1)^{s}\ M\ 2^{E}$ （不妨设E1>E2）
  - M: 按照大的E对齐相加
  - E: E1
  - 再规格化

例题-特殊反例

12. 下列关于教材第二章中整数和浮点数的说法中，正确的是(　　　).
A. 假设a是使用补码表示的整型，则表达式"-a == ~(a + 1)"为真.
B. 假设a和b是两个负整型，则可以通过表达式"a + b > 0"是否为真来判断a + b是否产生了溢出.
C. 假设a是浮点数，且a的阶码域为零，则a一定不是规格化数.
D. 假设a，b是两个浮点数，且它们都不是非数(NaN)，则表达式"a + b == b + a"为真.

- 答案： C

- ——b为什么是错的?
  - a=b=TMin

- ——d为什么是错的?
  - a=+∞，b=-∞，a+b=NaN

- NaN==NaN?
- NaN不等于任何数，恒为假

- TIPS：注意极端情况

Floating Point Operations

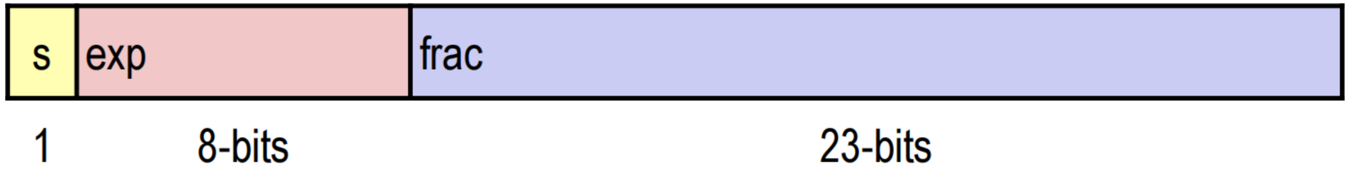
- 乘法:
    - 可交换，不可结合，不可分配

    - $(-1)^{s1}$ M1 2^{E1} x $(-1)^{s2}$ M2 2^{E2} = $(-1)^{s}$ M 2^{E}
    - s: s1 ^ s2
    - M: M1 x M2
    - E: E1 + E2

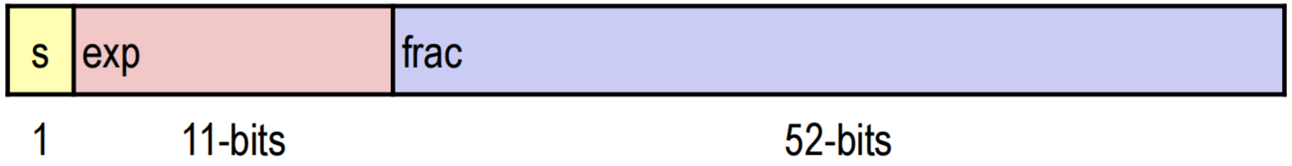
    - 再规格化
        - M右（左）移k位，E增加（减小）k

Floating Point in C

- float：单精度



- double：双精度



- int->float：不溢出，可能舍入
- int/float->double：保留精确数值
- double->float：可能溢出；也可能舍入
- float/double->int：向零舍入。可能溢出。

例题-浮点数误差

```
int i, j, k;
for(i = 0; i <= 2147483647 - 2; i ++){
  j = i + 1;
  k = j + 1;
  float *x = (float *)(&i);
  float *y = (float *)(&j);
  float *z = (float *)(&k);
  if(*y - *x != *z - *y){
    printf("0x%08x\n", i);      //输出 8 位 16 进制数
    break;
  }
}
```

其中 float 类型表示 IEEE-754 规定的浮点数，包括 1 位符号，8 位阶码，23 位尾数．请问该程序是否会有输出？_____(9)_____(填“是”或“否”)．若有输出，请给出输出内容，若没有输出，请说明理由_____(10)_____.

(9)是，**(10)0x00ffffff**

例题-浮点数误差

(　　　)10. 给定一个实数，会因为该实数表示成单精度浮点数(float)而发生误差。不考虑NaN和Inf的情况，该绝对误差的最大值为____.

A. 2^{103}　　B. 2^{104}　　C. 2^{230}　　D. 2^{231}

- 答案： A
- 回忆： float结构1-8-23
- 最大误差一定发生在整数，有效数字不足；阶码为8，最大为128位二进制数。
- 当且仅当舍入部分为中间值&采取偶数舍入时造成最大误差数量级，
- 舍入位数104位，为10000…00，即约2^{103}数量级。

有趣的例题讲解部分

- 抽取幸运观众答题（小心trap！）
- 知识点提炼

例题-字节顺序

1. 一个 IPv4 地址就是一个 32 位无符号整数。例如，10.2.155.253 对应的地址是 0x0a029bfd。协议规定，无论主机字节顺序如何，IP 地址在内存中总是以大端法来存放的。下面代码要实现的功能是检验 IP 是否符合 192.168.56.xx（xx 表示任意 0~255 的数）的模式，如果满足，则执行 if 语句内部指令。那么，在小端法的机器上，应该分别补充的数字是：

```
unsigned ip, mask;
// set ip
mask = ___________;
if(ip & mask == ___________)
{
    // do something
}
```

答案：A

A. 0x00ffffff　　0x0038a8c0

B. 0x00ffffff　　0x00838a0c

C. 0xffffff00　　0xc0a83800

D. 0xffffff00　　0x0c8a8300

例题-字节顺序

2. 在采用小端法存储机器上运行下面的代码，输出的结果将会是？

（int,unsigned 为 32 位长，short 为 16 位长，0~9 的 ASCII 码分别是 0x30~0x39）

```
char *s = "2018";
int *p1 = (int *)s;
short s1 = (*p1)>>12;
unsigned u1 = (unsigned) s1;
printf("0x%x\n",u1);
```

A） 0x00002303 B） 0x00032303 C） 0xffff8313 D） 0x00008313

答案：C

本题考查大小端存储，以及整数类型转化。字符串在存储时不区分大小端，前面的字符存储在低地址，而在被转化成整型的时候低地址被视为低位，因此*p1 为 0x38313032，s1 为 0x8313。在由 short 转化为 unsigned 的时候，我们要先改变大小，之后完成有符号到无符号的转换，因此 u1 为 0xffff8313。

例题-类型转化

2. 在 x86-64 机器上，有下列 C 代码

```
int main() {
    unsigned int A = 0x11112222;
    unsigned int B = 0x33336666;
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", P + Q);
    return 0;
}
```

运行该代码，结果为：0x____________。

- 答案：0x5555

```cpp
#include<iostream>
using namespace std;
int main() {
    unsigned int A=0x11112222;
    unsigned int B=0x33336666;
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%08x\n", *(int *)x);
    printf("0x%08x\n", *(int *)y);
    printf("0x%04x\n", P);
    printf("0x%04x\n", Q);
    printf("0x%04x", Q + P);
    return 0;
}
```

问题　输出　调试控制台　终端　端口

```
0x11112222
0x22223333
0x2222
0x3333
0x5555
```

例题-类型转化

3. 在 x86-64 机器上，有下列 C 代码

```c
int main() {
    char A[12] = "11224455";
    char B[12] = "11445577";
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", Q - P);
    return 0;
}
```

运行该代码，结果为：0x____________。

- 答案：0x0303

```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <queue>
using namespace std;

int main(){
    char A[12] ="11224455";
    char B[12] ="11445577";
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%08x\n",*(int *)x);
    printf("0x%08x\n",*(int *)y);
    printf("0x%04x\n", P);
    printf("0x%04x\n", Q);
    printf("0x%04x", Q - P);
    return 0;
}
```

```
0x32323131
0x35353434
0x3131
0x3434
0x0303
```

例题-精度与范围表示

(　　)8. 对于IEEE浮点数，如果减少1位指数位，将其用于小数部分，将会有怎样的效果？
A. 能表示更多数量的实数值，但实数值取值范围比原来小了.
B. 能表示的实数数量没有变化，但数值的精度更高了.
C. 能表示的最大实数变小，最小的实数变大，但数值的精度更高.

D. 以上说法都不正确.

- 答案：C

(　　)9. 下面关于IEEE浮点数标准说法正确的是____.
A. 在位数一定的情况下，不论怎么分配阶码位和小数部分，所能表示的数的个数不变
B. 如果甲类浮点数有10位，乙类浮点数有11位，那么甲所能表示的最大数一定比乙小
C. 如果甲类浮点数有10位，乙类浮点数有11位，那么甲所能表示的最小正数一定比乙小
D. ″0111000″可能是7位浮点数的NAN表示

- 答案：D

例题-综合
*简单题

第二题（15 分）

考虑有一种基于 IEEE 浮点格式的 9 位浮点表示格式 A。格式 A 有 1 个符号位，k 个阶码位，n 个小数位。现在已知$\frac{-9}{16}$的位模式可以表示为"101100010"，请回答以下问题：（注：阶码偏移量为2^{k-1}-1）

1. 求 k 和 n 的值。

2. 基于格式 A，请填写下表。值的表示可以写成整数（如 16），或者写成分数（如 17/64）。

描述	二进制位表示	值
最大的非规格化数		
最小的正规格化数		
最大的规格化数		

3. 假设格式 A 变为 1 个符号位，k+1 个阶码位，n-1 个小数位，那么能表示的实数数量会怎样变化，数值的精度会怎样变化？（回答增加、降低或不变即可）

考虑有一种基于 IEEE 浮点格式的 9 位浮点表示格式 A。格式 A 有 1 个符号位，k 个阶码位，n 个小数位。现在已知$\frac{-9}{16}$的位模式可以表示为"101100010"，请回答以下问题：（注：阶码偏移量为$2^{k-1}-1$）

1． 求 k 和 n 的值。（1 分） 答案：k = 4，n = 4

2． 基于格式 A，请填写下表。值的表示可以写成整数（如 16），或者写成分数（如 17/64）。（注：每格 2 分）

描述	二进制位表示	值
最大的非规格化数	0 0000 1111	15/1024
最小的正规格化数	0 0001 0000	1/64
最大的规格化数	0 1110 1111	248

3． 假设格式 A 变为 1 个符号位，k+1 个阶码位，n-1 个小数位，那么能表示的实数数量会怎样变化，数值的精度会怎样变化？（回答增加、降低或不变即可）（2 分）

答案：小数位变少后，NaN 的数量减少了，所以实数数量增加（1 分），数值精度降低（1 分）。

例题-精度与范围表示*复杂

(iii) 假设k+n=11,则该浮点数最多能精确表示____(8)____个连续的整数(用含k的代数式表示).

(8) 本问考察对浮点数表示的深入理解, 属难题.

当k较小时, $[-2^{2^{k-1}-1}\times(2-2^{-n}),2^{2^{k-1}-1}\times(2-2^{-n})]$之间的整数(包括零)都可以被连续精确表示, 这样的数的个数为

$$2\times(2^{2^{k-1}-1}\times 2-1)+1=2^{2^{k-1}+1}-1 \quad \cdots\cdots\text{①}$$

当k较大时, 一旦指数值超过n, 则该整数系统相邻整数之间会产生空隙, 此时只有$[-2^{n+1},2^{n+1}]$之间的整数可以被连续精确表示, 这样的数的个数为

$$2^{n+1}\times 2+1=2^{13-k}+1 \quad \cdots\cdots\text{②}$$

因此, 该浮点数系统最多能够精确表示

$$\min\{2^{2^{k-1}+1}-1,2^{13-k}+1\} \quad \cdots\cdots\text{③}$$

个连续的整数. 由于①式是增函数, ②式是减函数, k=4时①=511<513=②, 而k=5时①=$2^{17}-1$>257=②, 故最终答案为

$$\begin{cases}2^{2^{k-1}+1}-1(1<k\le 4);\\ 2^{13-k}+1(4<k\le 10).\end{cases} \quad \cdots\cdots\text{④}$$

- 书上好题:
  - p83 2.49
  - p87 2.54

*我们只处理了19-21年期中部分相关题and“。”中《2021秋ics小班练习题1》and 书上知识点内嵌习题，其余习题留给大家自行练习。