**整数答案：**

**1.** FF FF DF FC

**2.** Int类型：

* + 最大值：0x7FFFFFFF
  + 最小值：0x80000000
  + -1：0xFFFFFFFF
  + 0：0x00000000
* Unsigned short类型：
  + 最大值：1111 1111 1111 1111
  + 最小值：0000 0000 0000 0000

**3.** **直接截断取低16位，因此si的数值表示为0xef 01。**

**4.** 表格

AI 生成的内容可能不正确。

**5. n=4**

**在X86(小端)机器上，各字节存储0x0F,0x00,0x00,0x00内容。**

**在IBM的power8处理器这样的大端机器上，各字节存储0x00,0x00,0x00,0x0F**

**6.** **True // 解释 因为int类型的最小值是-2147483647-1，而题目给的数值超过int类型表示范围，因此 是long类型，1U表示该常数按照unsigned int存储，按照C语言隐式转换规则，此表 达式两边转换成long类型比较。因此答案是true。**

**7.** -8 (1000) **先符号拓展1111 1111 1111 1111 1111 1111 1111 1000**

**再补码转无符号数：10进制数值是4294967288（0Xff ff ff f8）**

**8.** **A**

**浮点数：**

**1.** **bias = 2^3-1 =7 exp=2+4=6 -> E=exp-7=-1 -> V=1.11 \* 2^-1 = 0.111=0.875**

**2.** **非负值最小规格化数：0 0000 0001 000…0**

**最小非规格化数：1 0000 0000 111…1**

* 解析：单精度浮点数的“非负值最小规格化数”的小数表示：
  + 因为是非负值，所以符号位s=0；
  + 规格化数要求阶码exp不为0和255，所以最小规格化数的阶码exp为0x01；
  + 尾数frac最小可为0；
  + 则小数表示为。
* 单精度浮点数的“最小非规格化数”的二进制表示：
  + 因为没有要求为非负值，则符号位s可为1；
  + 非规格化数，所以exp为0x00000000；
  + 由于符号位为1，即此浮点数为负数，因此尾数要最大，才能使得该浮点数最大，则frac有23个1构成；
  + 因此二进制表示为：1 00000000 11111111111111111111111。
* **负的绝对值最小值：1 00000000 00000000000000000000001 （非规格化）**
* **1 00000001 00000000000000000000000（规格化）**
* **1的位模式：0 01111111 00000000000000000000000**

**3.** **0 111 0000； 0 111 0001(不为0) 0 000 1111;**

**表示的数值范围变小了，靠近0周围的数表示精度变低，越大的数表示精度越大。**

**4. 1）0 01111111 00101100000000000000000**

**2）0 10000000 01100000000000000100100**

**5.** **1) 3**

**2) 1/32~15**

**3) 当小数位数不足时，浮点数舍入**表格

AI 生成的内容可能不正确。

**6.** **（1）-1 的二进制111…1被翻译为一个无符号数，类似进入一个死循环。为有符号数时不会出现死循环。**

**（2）**文本

AI 生成的内容可能不正确。 **= 1.111... × 2^n**

**126：如果是127，超过浮点数位数的部分会大于规格化最大值，所以是126**

**23：注意规格化隐含了一个1，实际上Float最多24位**

**7.**  **符号位为0，则该浮点数为非负值；**

**阶码为0110，则阶数为0110 – 0111 = 1111,即-1**

**尾数为110, 则“0 0110 110”所表示的数值为1.110×2^(-1)=0.1110(二进制表示)，其10进制表示为0.875**

**8.**

**CF400000 = 1100 1111 0100 0000 0000 0000 0000 0000 （补码）**

**转换为原码: 1011 0000 1011 1111 1111 1111 1111 1111(反码)+1**

**1 011 0000 1100 0000 0000 0000 0000 0000**

**定点小数：第一位为符号位 1，小数点在第一位后面 0.011 0000 11**

文本, 信件

AI 生成的内容可能不正确。

**(2) 1 10011110 10000000000000000000000**

**符号位为0——负数**

**E = 158 – 127 = 31**

**M= (1.1)2 = 1.5**

**D = - 231 x 1.5 = -3 x 230**

**9. AI：此等式并非恒成立，因左表达式的双重舍入​（独立转换 + 加法）可能产生与右表达式（精确求和后单次转换）不同的结果，尤其在整数超出 ±2^24 范围时**

**举例： x= 0 11111111 0000.. (无穷大) y=0 00000001 0000.. (非常小)**

**x+y = 1 00000000 000...(零)**

文本, 信件

AI 生成的内容可能不正确。

**10.** **C: 因为NaN更多，所以能表示的实数值更少。指数小了，所以最大实数变小，最小实数变大。 小数部分增加了，所以精度更高。**

**11.**

表格

AI 生成的内容可能不正确。

表格

AI 生成的内容可能不正确。

**有限数是指一个有界的、非无穷大也非非数（NaN）的数值 --> 规格化的最小值（负数最大）**

**注意偶数舍入**

**12. D**

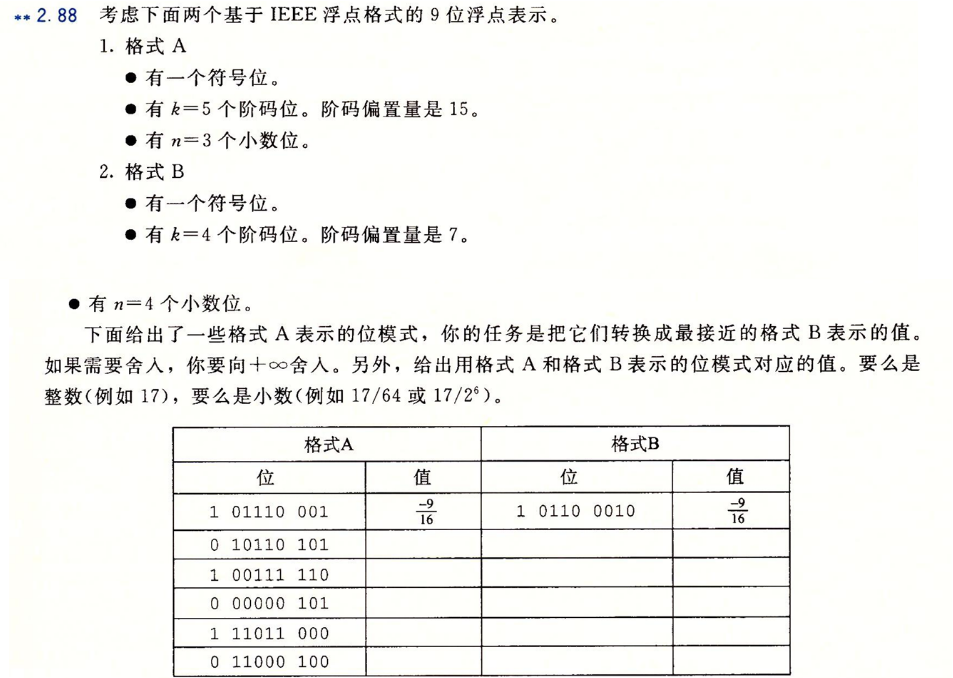
A:如果位数固定，从 0000 ~ 1111，如果理解为不同的比特组合个数，那说法是正确的；理解为能表示的不同“实数值”的个数，那说法是错误的

B: 浮点数最大数：规格化最大数 -> 1.frac \* 2^E ，结论：exp分配位数越多，最大数越大，所以取决于 甲和乙怎么分配

C. 最小正数： 非规格化 00000000…1 ,E=1-bias,所以尾数位多或偏置大（指数位多）可能使最小正数更小，但取决于具体分配

**D: 0 11 1000 exp全1，frac!=0**

**补充题目：**



**答案：**

| **格式A-位** | **格式A-值** | **格式B-位** | **格式B-值** |
| --- | --- | --- | --- |
| **1 01110 001** | **-9/16** | **1 0110 0010** | **-9/16** |
| **0 10110 101** | **13\*2^4** | **0 1110 1010** | **13\*2^4** |
| **1 00111 110** | **-7/2^10** | **1 0000 0111** | **-7/2^10** |
| **0 00000 101** | **5/2^-17** | **0 0000 0000** | **0** |
| **1 11011 000** | **-2^12** | **1 1110 1111** | **-31\*2^3** |
| **0 11000 100** | **3\*2^8** | **0 1111 0000** | **+oo** |