**Chapter 3:**

问题解答：pp. 45, Summary中：

A floating-point number is a whole number and a fraction. Conversion of the fraction to binary requires the denominator of the fraction to be expressed as a power of 2. The Excess\_X system is used to store this power of 2.

翻译：一个浮点数是由一个整数和一个分数（或称小数）组成。分数部分转换成二进制，需要将分数部分的分母表示成2的幂。Excess\_X系统是用来存储2的幂。

解释：小数其实也是分数的一种表示方法，举例，一个小数是0.75, 即75/100，分母是由10的2次幂表示。推广至二进制数，一个二进制的小数，实际上也是分子和分母组成，只不过分母不是10的x次幂，而是2的x次幂。例如二进制数中的0.11011， 其实是25 分之11011。

验证： （25分之11011）2= （27/32）10 = (0.84375)10 = (0.11011)2.

部分习题答案：

20：c

21-25：addbc

26-30：ddddd

31-35：cbdda

36-40：cbcbc

41-45：bddcc

46：b

47：a. 00010111 b.0 1111001 c. 00100010 d. overflow溢出

48: a.0000000000101001 b. 0000000110011011 c. 000001001101001

d. 0000000101010110

49.a. 32 00100000 b.-101 11100101, c. 56 00111000 d. 129 overflow

50. a. 142 00000000010001110 b.-180 1000000010011100

52：a. 0000 0000 1010 0010

b. -110: (1101110)2, 1111 1111 1001 0001

c.0000101000000000（9个0）

d. 没溢出, 16位反码的范围:正负2^15-1, 即正负32767之间, 0010111101011011

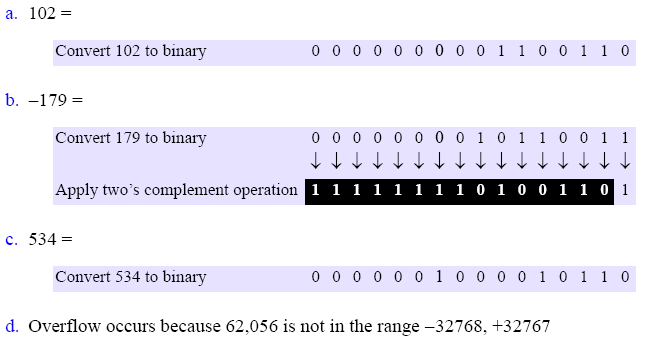
53：a. -12 (1100)2 1111 0100

b. 1001 1011

c. 00111000

d. 溢出

54.



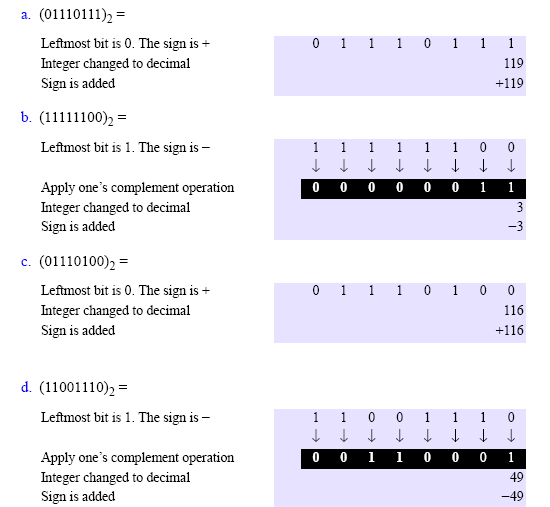
55: a. 107 b. 148 c.6 d. 80

56：a. +123 b.-52 c.99 d. -80

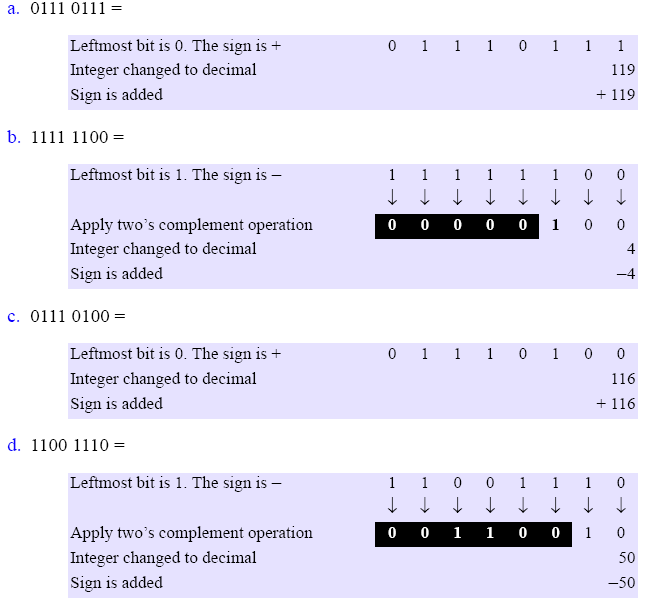
57：a. +99 b. -123 c. +115 d. -63

58. a. +119 b. -4 c. +116 d. -50

60.



61.



62. 事实上是上课时讲的第二种方法。 可应用两种方法，并对比结果。

a. 01110111不变，b. 11111100->10000011+1->10000100,和上课讲的方法相同

c. 01110100不变，d. 11001110->10110001+1->10110010,和上课讲的方法相同

63，64略

65. a. 234 b. 112 c. 874 d. 888

66. 按照二进制反码one’s complement的思想，应该理解为正数时不变；负数时取9的补码。二进制反码即正数不变，负数的反码可以通过使用1减去正数的每一位，假设每一位用x标识，即使用1-x就可以求得该负数的反码。对应到十进制的反码：即用9-x可以求得负数的反码，x为对应正数的每一位数值。正数不变。

也就是3位的数，从000-999， 用000-499表示+0 - -+499; 用 500 - 999表示-499 - -0。

也就是在这个系统中，可表示的数的范围为-499 - +499。当最左边的一位小于5时，为正数；大于等于5时，为负数。有2个0（0 和 999）。所以如果给出一个数+560，让你求其9的反码时，答案应该是该数会超出9的反码的标识范围，会溢出。

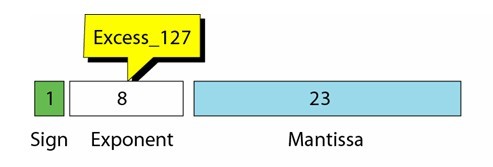
67. a. 20 \* 1.10001

b. 25 \* 1.111111

c. 20 \* 1.01110011

d. 20 \* 1.011010000011000

68. 也就是单精度表示法



a. 1 01111111 10001000000000000000000

b. 0 10000010 11111100000000000000000

c. 0 01111011 011100110..0(15个0)

d. 1 01111010 011010000..0(18个0)

69. 双精度数(64位表示)

a. 1 01111111111 100010..0(47个0)

b. 0 10000000010 1111110..0(46个0)

c. 0 01111111011 011100110..0(44个0)

d. 1 01111111010 011010000..0(44个0)

70. a. 11.101

b. 1100.00011

c. 100.001101

d. 1100.0000101

71.

a. 0.1875 = 0.125+0.0625=1/8 + 1/16=3/16

b. 0.640625=0.5+0.125+0.015625=1/2+1/8+1/64=41/64

c. 0.40625=0.25+0.125+0.03125=1/4+1/8+1/32=13/32

d. 0.375=0.25+0.125=1/4+1/8=3/8

72. a. 7.1875 = 111+1/8 + 1/16=111.0011

b. 12.640625=1100.101001

c. 11.40625=1011.01101

d. 0.375=0.011

73. a. 7.1875 = 111+ 2-3\*1.0 + 2-4\*1.0 = 2-3 \* (111000 + 1 + 0.1) = 22 \* 1.110011

用32位IEEE表示：

0 11111101 11001100000000000000000

a. 7.1875 = (111.0011)2 = 22 × 1.110011

S = 0

E = 2 + 127 = 129 = (10000001)2

M = 110011 (plus 17 zero at the right)

→ **0 10000001 11001100000000000000000**

b. −12.640625 = (−1100.101001)2 = − 23 × 1.100101001

S = 1

E = 3 + 127 = 130 = (10000010)2

M = 100101001 (plus 14 zero at the right)

→ **1 10000010 10010100100000000000000**

c. 11.40625 = (1011.01101)2 = 23 × 1.01101101

S = 0

E = 3 + 127 = 130 = (10000010)2

M = 01101101 (plus 15 zero at the right)

→ **0 10000010 01101101000000000000000**

d. −0.375 = −0.011 = − 2−2 × 1.1

S = 1

E = −2 + 127 =125 = (01111101)2

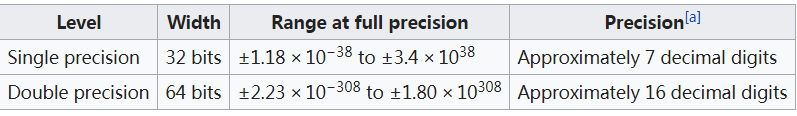
M = 1 (plus 22 zero at the right)

→ **1 01111101 10000000000000000000000**

74 略

**补充：关于浮点数的标识范围：**

<https://en.wikipedia.org/wiki/IEEE_754-1985>



单精度浮点数各种极值情况：下面只考虑规范化数，即规约数。注意特殊数的表示。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **类别** | **正负号** | **实际指数** | **有偏移指数** | **指数域** | **尾数域** | **数值** |
| 零 | 0 | -127 | 0 | 0000 0000 | 000 0000 0000 0000 0000 0000 | 0.0 |
| **类别** | **正负号** | **实际指数** | **有偏移指数** | **指数域** | **尾数域** | **数值** |
| 负零 | 1 | -127 | 0 | 0000 0000 | 000 0000 0000 0000 0000 0000 | −0.0 |
| 1 | 0 | 0 | 127 | 0111 1111 | 000 0000 0000 0000 0000 0000 | 1.0 |
| -1 | 1 | 0 | 127 | 0111 1111 | 000 0000 0000 0000 0000 0000 | −1.0 |
| 最小的非规约数 | \* | -126 | 0 | 0000 0000 | 000 0000 0000 0000 0000 0001 | ±2−23 × 2−126 = ±2−149 ≈ ±1.4×10-45 |
| 中间大小的非规约数 | \* | -126 | 0 | 0000 0000 | 100 0000 0000 0000 0000 0000 | ±2−1 × 2−126 = ±2−127 ≈ ±5.88×10-39 |
| 最大的非规约数 | \* | -126 | 0 | 0000 0000 | 111 1111 1111 1111 1111 1111 | ±(1−2−23) × 2−126 ≈ ±1.18×10-38 |
| 最小的规约数 | \* | -126 | 1 | 0000 0001 | 000 0000 0000 0000 0000 0000 | ±2−126 ≈ ±1.18×10-38 |
| 最大的规约数 | \* | 127 | 254 | 1111 1110 | 111 1111 1111 1111 1111 1111 | ±(2−2−23) × 2127 ≈ ±3.4×1038 |
| 正无穷 | 0 | 128 | 255 | 1111 1111 | 000 0000 0000 0000 0000 0000 | +∞ |
| 负无穷 | 1 | 128 | 255 | 1111 1111 | 000 0000 0000 0000 0000 0000 | −∞ |
| [NaN](http://zh.wikipedia.org/wiki/NaN) | \* | 128 | 255 | 1111 1111 | non zero | NaN（not a number） |
| \* 符号位可以为0或1 . | | | | | | |

注：1. S(1位)    E(8位)        M(23位)    N(32位)    
符            0                0            (-1)S \*2E-127·(1.M) 为规格化数                                       
               0               不等于0      (-1)S\*2-126\*(0.M) 为非规格化数  
号       1到254之间     不等于0    (-1)S\*2E-127\*(1.M) 为规格化数  
              255            不等于0      NaN(非数值)   
位           255                0           无穷大

其中红色字0、1表示隐含位，注意当数字N为非规格化数或是0时，隐含位是0。

2. 当阶码E 为全0且尾数M 也为全0时，表示的真值x 为零，结合符号位S 为0或1，有正零和负零之分。当阶码E 为全1且尾数M 为全0时，表示的真值x 为无穷大，结合符号位S 为0或1，也有+∞和-∞之分。这样在32位浮点数表示中，要除去E 用全0和全1(255)10表示零和无穷大的特殊情况，指数的偏移值不选128(10000000)，而选127(01111111)。对于规格化浮点数，E 的范围变为1到254，真正的指数值e 则为-126到+127。因此32位浮点数表示的绝对值的范围是10-38~1038（以10的幂表示）。"——引自 白中英<<计算机组成原理>>