**6.10**

设机器字长为8位（含一位符号位）

[+0]原=0，0000000 [- 0]原=1，0000000

[+0]补= [- 0]补=0，0000000

[+0]反=0，0000000 [- 0]反=1，1111111

[+0]移= [- 0]移=1，0000000

结论：补码和移码表示的零是唯一的，原码和反码表示的零不唯一。

**6.15**

机器零：当一个浮点数的尾数为0时，不论其阶码为何值，或者当一个浮点数的阶码等于或小于它所能表示的最小数时，不论其尾数为何值，机器都把该浮点数当机器零处理。

全0表示机器零：

尾数：采用**补码**时全0表示尾数为0

阶码：采用**移码**时，全0为其所能表示的最小数

**6.16**

字长：16位

1. 无符号数：

整数：0~ 即0~65535

小数：0~ 即0~0.9999847412109375

1. 原码表示的定点小数：

~ ，即 -0.999969482421875~0.999969482421875

1. 补码表示的定点小数：

~ ，即 -1~0.999969482421875

1. 补码表示的定点整数：

~ ，即 -32,768~32,767

1. 原码表示的定点整数：

~ ，即 -32,767~32,767

1. 浮点数，阶码6位（含1位阶符），尾数10位（含1位数符）

正数：~ ,

即 0.00000000000090949470177~2143289344

负数： ~

即 -2143289344 ~ -0.00000000000090949470177

1. （6）补码规格化，若不考虑隐藏位：

正数：~

即 0.00000000011641532182693~2143289344

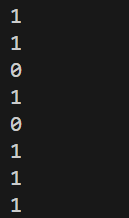
负数： ~ ：

即 -4,294,967,296 ~ -0.00000000011641532182693

**补充题**

1. 常见的数值的简洁编码表示：

可用Huffman编码实现变长编码，高频值用短码，低频值用长码。不同长度编码之间的区分则需要使用​前导字节标记、游程编码等方法

**（2）**机器字长64位

1. 0 == 0U

2. -1 < 0

3. -1 < 0U

4. 2147483647 > -2147483647-1

5. 2147483647U > -2147483647-1

6. 2147483647 > (int) 2147483648U

7. -1 > -2

8. (unsigned) -1 > -2

1、2、4、7显然

3. 是由于C语言中进行符号整数和无符号整数混合运算时，有符号整数会被隐式转换为无符号整数。-1补码表示1111...1111转换为无符号数为264-1>0

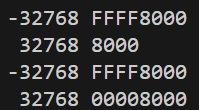
5. 的原因同上

6 . int表示的最大值为2147483647，而2147483648U超过int能表达的范围，所以

2147483648U=1000 0000 0000 0000 0000 0000 0000 0000强制转换为有符号整数int类型的补码，即真值变为-2147483647 < 2147483647

8. -1的补码为1111...1111，-2的补码为1111...1110，而1111...1111U > 1111...1110U

**（3）**机器字长64位

short si = -32768;

unsigned short usi = si;

int i = si;

unsigned ui = usi ;

真值的结果显然如此

机器数的结果，前三个显然，最后ui是对unsigned short类型的usi做了零扩展，所以多出了高位的4个0