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：$0 \sim 2^{16}-1$ 即 0~65535

小数：$0 \sim 1-2^{-16}$ 即 0~0.9999847412109375

（2）　原码表示的定点小数：

$-1+2^{-15} \sim 1-2^{-15}$ ，即 -0.999969482421875~0.999969482421875

(3)　补码表示的定点小数：

$-1 \sim 1-2^{-15}$ ，即 -1~0.999969482421875

(4)　补码表示的定点整数：

$-2^{15} \sim 2^{15}-1$ ，即 -32,768~32,767

(5)　原码表示的定点整数：

$-2^{15}+1 \sim 2^{15}-1$ ，即 -32,767~32,767

(6)　浮点数，阶码 6 位（含 1 位阶符），尾数 10 位（含 1 位数符）

正数：$2^{-(2^5-1)} \times 2^{-9} \sim 2^{(2^5-1)} \times (1-2^{-9})$ ，

即 0.0000000000090949470177~2143289344

负数：$-2^{(2^5-1)} \times (1-2^{-9}) \sim -2^{-(2^5-1)} \times 2^{-9}$

即 -2143289344 ~ -0.0000000000090949470177

(7)　（6）补码规格化，若不考虑隐藏位：

正数：$2^{-2^5} \times 2^{-1} \sim 2^{(2^5-1)} \times (1-2^{-9})$

即 0.00000000011641532182693~2143289344

负数：$-2^{2^5} \times 1 \sim -2^{-2^5} \times 2^{-1}$

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
1
0
1
0
1
1
1
```

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 2^{64}-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

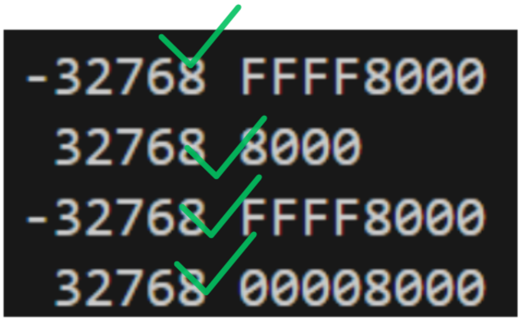
(3) 机器字长 64 位

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