2.17　一般而言，研究字长非常小的例子是理解计算机运算的非常好的方法。

无符号值对应于图 2-2 中的值。对于补码值，十六进制数字 0～7 的最高有效位为 0，得到非负值，然而十六进制数字 8～F 的最高有效位为 1，得到一个为负的值。

$\vec{x}$		$B2U_4(\vec{x})$	$B2T_4(\vec{x})$
十六进制	二进制		
0xE	[1110]	$2^3+2^2+2^1=14$	$-2^3+2^2+2^1=-2$
0x0	[0000]	0	0
0x5	[0101]	$2^2+2^0=5$	$2^2+2^0=5$
0x8	[1000]	$2^3=8$	$-2^3=-8$
0xD	[1101]	$2^3+2^2+2^0=13$	$-2^3+2^2+2^0=-3$
0xF	[1111]	$2^3+2^2+2^1+2^0=15$	$-2^3+2^2+2^1+2^0=-1$

2.18　对于 32 位的机器，由 8 个十六进制数字组成的，且开始的那个数字在 8～f 之间的任何值，都是一个负数。数字以串 *f* 开头是很普遍的事情，因为负数的起始位全为 1。不过，你必须看仔细了。例如，数 0x8048337 仅仅有 7 个数字。把起始位填入 0，从而得到 0x08048337，这是一个正数。

```
4004d0:  48 81 ec e0 02 00 00    sub   $0x2e0,%rsp            A. 736
4004d7:  48 8b 44 24 a8          mov   -0x58(%rsp),%rax       B. -88
4004dc:  48 03 47 28             add   0x28(%rdi),%rax        C.  40
4004e0:  48 89 44 24 d0          mov   %rax,-0x30(%rsp)       D. -48
4004e5:  48 8b 44 24 78          mov   0x78(%rsp),%rax        E. 120
4004ea:  48 89 87 88 00 00 00    mov   %rax,0x88(%rdi)        F. 136
4004f1:  48 8b 84 24 f8 01 00    mov   0x1f8(%rsp),%rax       G. 504
4004f8:  00
4004f9:  48 03 44 24 08          add   0x8(%rsp),%rax
4004fe:  48 89 84 24 c0 00 00    mov   %rax,0xc0(%rsp)        H. 192
400505:  00
400506:  48 8b 44 d4 b8          mov   -0x48(%rsp,%rdx,8),%rax I. -72
```

2.19　从数学的视角来看，函数 *T2U* 和 *U2T* 是非常奇特的。理解它们的行为非常重要。

我们根据补码的值解答这个问题，重新排列练习题 2.17 的解答中的行，然后列出无符号值作为函数应用的结果。我们展示十六进制值，以使这个过程更加具体。

$\vec{x}$（十六进制）	x	$T2U_4(x)$
0x8	−8	8
0xD	−3	13
0xE	−2	14
0xF	−1	15
0x0	0	0
0x5	5	5

2.20　这个练习题测试你对等式(2.5)的理解。

对于前 4 个条目，x 的值是负的，并且 $T2U_4(x)=x+2^4$。对于剩下的两个条目，x 的值是非负的，并且 $T2U_4(x)=x$。

2.21　这个问题加强你对补码和无符号表示之间关系的理解，以及对 C 语言升级规则(promotion rule)的影响的理解。回想一下，$TMin_{32}$ 是 −2 147 483 648，并且将它强制类型转换为无符号数后，变成了 2 147 483 648。另外，如果有任何一个运算数是无符号的，那么在比较之前，另一个运算数会被强制类型转换为无符号数。

表达式	类型	求值
-2147483647-1 == 2147483648U	无符号数	1
-2147483647-1 < 2147483647	有符号数	1
-2147483647-1U < 2147483647	无符号数	0
-2147483647-1 < -2147483647	有符号数	1
-2147483647-1U < -2147483647	无符号数	1