个 1 组成。这些位模式有助于掩码运算。这种模式能够通过 C 表达式 (1<<k)-1 生成，利用的是这样一个属性，即我们想要的位模式的数值为 2^k-1。例如，表达式 (1<<8)-1 将产生位模式 0xFF。

浮点表示通过将数字编码为 $x\times 2^y$ 的形式来近似地表示实数。最常见的浮点表示方式是由 IEEE 标准 754 定义的。它提供了几种不同的精度，最常见的是单精度(32 位)和双精度(64 位)。IEEE 浮点也能够表示特殊值 $+\infty$、$-\infty$和 NaN。

必须非常小心地使用浮点运算，因为浮点运算只有有限的范围和精度，而且并不遵守普遍的算术属性，比如结合性。

参考文献说明

关于 C 语言的参考书[45，61]讨论了不同的数据类型和运算的属性。(这两本书中，只有 Steele 和 Harbison 的书[45]涵盖了 ISO C99 中的新特性。目前还没有看到任何涉及 ISO C11 新特性的书籍。)对于精确的字长或者数字编码 C 语言标准没有详细的定义。这些细节是故意省去的，这样可以在更大范围的不同机器上实现 C 语言。已经有几本书[59，74]给了 C 语言程序员一些建议，警告他们关于溢出、隐式强制类型转换到无符号数，以及其他一些已经在这一章中谈及的陷阱。这些书还提供了对变量命名、编码风格和代码测试的有益建议。Seacord 的书[97]是关于 C 和 C++ 程序中的安全问题的，本书结合了 C 程序的有关信息，介绍了如何编译和执行程序，以及漏洞是如何造成的。关于 Java 的书(我们推荐 Java 语言的创始人 James Gosling 参与编写的一本书[5])描述了 Java 支持的数据格式和算术运算。

关于逻辑设计的书[58，116]都有关于编码和算术运算的章节，描述了实现算术电路的不同方式。Overton 的关于 IEEE 浮点数的书[82]，从数字应用程序员的角度，详细描述了格式和属性。

家庭作业

- ◆2.55　在你能够访问的不同机器上，使用 show_bytes(文件 show-bytes.c)编译并运行示例代码。确定这些机器使用的字节顺序。
- ◆2.56　试着用不同的示例值来运行 show_bytes 的代码。
- ◆2.57　编写程序 show_short、show_long 和 show_double，它们分别打印类型为 short、long 和 double 的 C 语言对象的字节表示。请试着在几种机器上运行。
- ◆◆2.58　编写过程 is_little_endian，当在小端法机器上编译和运行时返回 1，在大端法机器上编译运行时则返回 0。这个程序应该可以运行在任何机器上，无论机器的字长是多少。
- ◆◆2.59　编写一个 C 表达式，它生成一个字，由 x 的最低有效字节和 y 中剩下的字节组成。对于运算数 x=0x89ABCDEF 和 y=0x76543210，就得到 0x765432EF。
- ◆◆2.60　假设我们将一个 w 位的字中的字节从 0(最低位)到 $w/8-1$(最高位)编号。写出下面 C 函数的代码，它会返回一个无符号值，其中参数 x 的字节 i 被替换成字节 b：

```
unsigned replace_byte (unsigned x, int i, unsigned char b);
```

以下示例，说明了这个函数该如何工作：

```
replace_byte(0x12345678, 2, 0xAB) --> 0x12AB5678
replace_byte(0x12345678, 0, 0xAB) --> 0x123456AB
```

位级整数编码规则

在接下来的作业中，我们特意限制了你能使用的编程结构，来帮你更好地理解 C 语言的位级、逻辑和算术运算。在回答这些问题时，你的代码必须遵守以下规则：

- 假设
  - 整数用补码形式表示。
  - 有符号数的右移是算术右移。
  - 数据类型 int 是 w 位长的。对于某些题目，会给定 w 的值，但是在其他情况下，只要 w 是 8 的整数倍，你的代码就应该能工作。你可以用表达式 sizeof(int)<<3 来计算 w。