到总的数值，就要用每个位置上的符号乘以位值权重，然后再把结果加起来。所以，从右边开始，就有（1×1）+（0×2）+（1×4）=5。

现在，你可以自己尝试把二进制数转换成十进制数了。

练习1-2：将二进制转换成十进制

把下列用二进制表示的数转换成等价的十进制数。

10（二进制）=____（十进制）

111（二进制）=____（十进制）

1010（二进制）=____（十进制）

你可以通过附录A中的答案检查结果。你做对了吗？最后一小题可能有点棘手，因为它在左边多引入了一位，即8的位置。现在，尝试一下从十进制转换成二进制。

练习1-3：将十进制转换成二进制

把下列用十进制表示的数转换成等价的二进制数。

3（十进制）=____（二进制）

8（十进制）=____（二进制）

14（十进制）=____（二进制）

我希望这些题你也答对了！很快你就会发现同时处理十进制和二进制会造成混淆，因为一个像10这样的数，在十进制中表示的是10，在二进制中表示的是2。本书从现在开始，如果有混淆的苗头，那么二进制数将用0b开头。之所以选择0b作为前缀，是因为有几种编程语言采用了这种方法。前导字符0表示数字值，b是二进制（binary）的缩写。例如，0b10代表二进制的2，而10没有前缀，则代表十进制的10。

1.4 位和字节

十进制数中的单个位或符号称为数字（digit）。像1247这样的十进制数就是一个四位数。类似地，二进制数中的单个位或符号称为位（bit）。每个位都可以是0或1。像0b110这样的二进制数就是一个3位数。

单个位能传递的信息很少：要么关，要么开；要么0，要么1。我们需要用位序列来表示更复杂的东西。为了让这些位序列更易于管理，计算机把8位分成一组——称为字节。下面是一些位和字节的例子（因为都是二进制，所以省略了前缀0b）：

1 这是一个位

0 这还是一个位

11001110 这是一个字节或8位

00111000 这也是一个字节

10100101 这还是一个字节

0011100010100101 这是两个字节或16位

注意

4位数（即半个字节）有时被称为半字节（写为nybble或nyble）。

一个字节能存储多少数据呢？考虑这个问题的另一个思路是：用8位可以得到多少个不同的0/1组合？在回答这个问题之前，我们用4位来说明，这样更容易看明白。

在表1-1中，我列出了4位数中所有可能的0/1组合，还给出了与该数字对应的十进制表示。

如表1-1所示，我们可以用4位数表示16个不同的0/1组合，范围从十进制的0到15。查看位的组合列表有助于说明这一点，但是我们还有几种方法可以解释这个问题，而不用枚举每种可能的组合。

表1-1 4位数的所有可能值

二进制	十进制
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

通过把所有位都设置为1，我们可以确定4位能表示的最大数字，即0b1111。这就是十进制的15。如果把表示的0也算上，那么得到的总数就是16。另一种简便的方法是取2的位数次幂，这里位数为4，那么总共有$2^4=2\times2\times2\times2=16$个0/1组合。

研究4位数是一个好的开始，不过我们前面谈论的是字节，它有8位。按照上面的方法，我们可以列出全部的0/1组合，但是我们跳过这一步，直接采用简便方法，计算2的8次方，可以得到256，这就是一个字节所含的不同组合的数量。

现在，我们知道了一个4位数可以有16种0/1组合，一个字节可以有256种组合。这和计算机有什么关系呢？假设一个计算机游戏有12个关卡，那么这个游戏只需要4位就可以轻松地保存当前关卡号。如果游戏有99个关卡，那么4位就不够用了，4位最多只能表示16个关卡！但是，一个字节就能很好地处理99个关卡的需求。计算机工程师有时需要考虑用多少位或多少字节来保存数据。

1.5 前缀

表示复杂的数据类型需要大量的位。像数字99这样简单的对象只需要一个字节，而数字格式的视频则可能需要数十亿位。为了更易于表示数据的大小，我们使用了诸如G和M之类的前缀。国际单位制（SI）也被称为公制，定义了一组标准前缀。这些前缀用于描述可以被量化的对象，并不仅限于位。在后面关于电路的章节中，我们还会看到它们。表1-2列出了一些常见的SI前缀及其含义。

表1-2 常见SI前缀

前缀名	前缀符号	值	以 10 为基数
太	T	1 000 000 000 000	10^{12}
吉	G	1 000 000 000	10^{9}

(续)

前缀名	前缀符号	值	以 10 为基数
兆	M	1 000 000	10^6
千	k	1000	10^3
厘	c	0.01	10^{-2}
毫	m	0.001	10^{-3}
微	μ	0.000 001	10^{-6}
纳	n	0.000 000 001	10^{-9}
皮	p	0.000 000 000 001	10^{-12}

有了这些前缀，如果我们想说“30亿字节”，就可以用缩写3GB表示。或者，如果我们想表示4000位，我们就可以说4kb。注意，B代表字节，b代表位。

你会发现这种约定一般用于表示位和字节的数量，可惜的是，这在技术上通常也是不正确的。其原因在于：在处理字节时，大多数软件实际是以2为基数的，而不是以10为基数，如果计算机告诉你文件的大小为1MB，那么它实际有1048576字节！这约等于10^6，但不完全等于。看起来像是个奇怪的数，不是吗？这是因为我们是按十进制来看的。在二进制中，同样的数字表示为0b10000000000000000000，它是2^{20}。表1-3展示了在处理字节时如何解释SI前缀。

表1-3 SI前缀应用于字节时的含义

前缀名	前缀符号	值	以 2 为基数
太	T	1 099 511 627 776	2^{40}
吉	G	1 073 741 824	2^{30}
兆	M	1 048 576	2^{20}
千	K	1024	2^{10}

位和字节另一个会混淆的地方和网络传输速率有关。互联网服务提供商通常以每秒位数作为单位，基数为10。因此，如果互联网连接传输速率为50Mb/s，那么就意味着每秒只能传送大约6MB数据。也就是说，每秒50000000位，除以每字节8位，得到每秒6250000字节，用6250000除以2^{20}，得到每秒大约6MB。

二进制数据的SI前缀

行业内已习惯用KB表示1024字节。——编辑注

为了解决由前缀多义性导致的混乱，2002年引入了一组新的前缀（在IEEE 1541标准中）用于二进制场景。当处理2的幂时，Ki用于代替K，Mi用于代替M，以此类推。这些新的前缀对应于以2为基数的值，用于之前不正确使用旧前缀的场合。例如，由于KB可能被解释为1000或1024个字节，因此这个标准就建议使用KiB来表示1024个字节，而KB则保留其原始意义，即KB等于1000个字节[注]。

这看上去是个好主意，但是直到撰写本书的时候，这些符号还未得到广泛使用。表1-4列出了新的前缀及其含义。

表1-4 IEEE 1541-2002二进制数据的前缀

前缀名	前缀符号	值	以 2 为基数
二进制太	Ti	1 099 511 627 776	2^{40}
二进制吉	Gi	1 073 741 824	2^{30}
二进制兆	Mi	1 048 576	2^{20}
二进制千	Ki	1024	2^{10}

这个差异很重要，因为在实践中，大多数软件在显示文件大小的时候使用的是旧的SI前缀，但在计算大小的时候却是以2为基数的。换句话说，如果你的设备说一个文件的大小是1KB，那么它的意思是说有1024字节。此外，存储设备制造商在为它们的设备容量打广告的时候，则倾向于以10为基数来表示容量大小。这就意味着，广告宣称为1TB的硬盘可能包含10^{12}字节，但是当你把它连接到计算机时，则会显示其容量大约为931GB（10^{12}除以2^{30}）。由于新前缀使用率不高，因此本书将继续使用旧的SI前缀。

1.6 十六进制

在结束用二进制思考这个话题之前，我还要提一下另一个数字系统：十六进制。我们"正常的"数字系统是十进制系统，或者说以10为基数的系统。计算机使用的是二进制系统，即以2为基数的系统。十六进制系统以16为基数！十六进制（缩写为hex）系统也是一种位值系统，其中的每个位置都表示16的幂，每个位置的符号都可以是16个符号中的一个。

和所有的位值系统一样，最右边的位置仍是1的位置，其左边第一位是16的位置，然后是256（16×16）的位置，之后是4096（16×16×16）的位置，以此类推。非常简单！但是，每个位置可以放置的16个符号是哪些呢？通常，我们有表示数字0～9的10个符号。要表示其他值，还需要另外6个符号。我们可以随机选择一些符号，比如&、@和#，但是这些符号没有明显的顺序。相反，采用A、B、C、D、E和F（大小写均可）才是标准做法。在这个方案中，A代表10，B代表11，以此类推，直到F，它代表的是15。这样是对的，我们需要一些符号来表示从0到基数减1的数字。因此，其他6个符号是A～F。为了清晰起见，标准的做法是用前缀0x来表示十六进制。表1-5列出了16个十六进制符号，以及它们十进制和二进制的值。

表1-5 十六进制符号

十六进制	十进制	二进制（4 位）	十六进制	十进制	二进制（4 位）
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	A	10	1010
3	3	0011	B	11	1011
4	4	0100	C	12	1100
5	5	0101	D	13	1101
6	6	0110	E	14	1110
7	7	0111	F	15	1111

当需要表示的值大于十进制的15或十六进制的0xF时会怎么样呢？和十进制一样，我们增加一位。0xF的后面是0x10，这就是十进制的16。然后是0x11、0x12、0x13，以此类推。现在看一下图1-4，便可以看到更大的十六进制数0x1A5。

图1-4给出了十六进制数0x1A5，它的十进制值是多少呢？最右边位置的值为5，旁边位置的权重为16，位置上有个A，A代表十进制的10，因此，中间位置为16×10=160。最左边位置的权重为256，位置上有个1，所以这个位置的值为256。总数值用十进制表示为5+160+256=421。

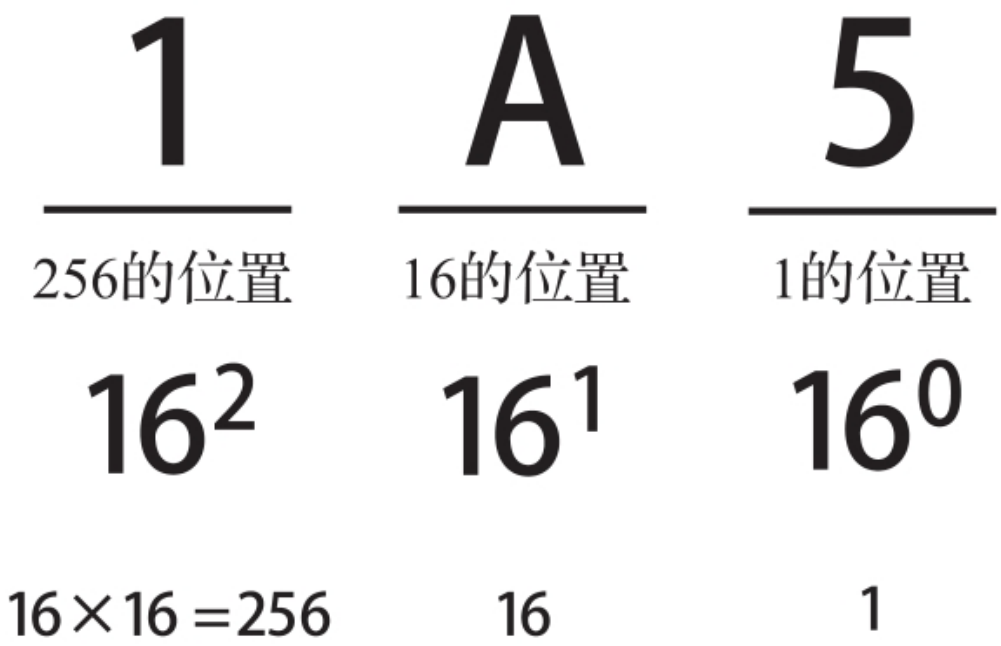

图1-4 按位值分解十六进制数0x1A5

为了强调这一点，这个例子展示了像A这样的新符号是如何根据其出现的位置而有不同的值的。0xA是十进制的10，0xA0则是十进制的160，因为A出现在16的位置上。

现在，你可能会说："太棒了，但是这有什么用呢？"我很高兴你有这样的疑问。计算机不使用十六进制，大多数人也不使用。但是，十六进制对于那些要处理二进制数的人来说却是很有用的。

使用十六进制有助于克服处理二进制数的两个常见困难。首先，大多数人都不擅长阅读较长的0/1序列。之后，这些位要一起运行。对于人类而言，处理16个或更多的位很困难且容易出错。其次，尽管人类善于处理十进制数，但十进制和二进制之间的转换并不容易。对于大多数人来说，很

难在看到一个十进制数后，就能很快说出该数用二进制表示时哪些位是1，哪些位是0。但是，将十六进制转换成二进制要简单得多。表1-6给出了2个16位二进制数及其对应的十六进制与十进制表示。注意，为了清晰起见，我在二进制数值中添加了空格。

表1-6 16位二进制数及其十六进制和十进制表示

	例 1	例 2
二进制	1111 0000 0000 1111	1000 1000 1000 0001
十六进制	F00F	8881
十进制	61 455	34 945

先看表1-6中的例1。它的二进制表示是个明确的序列：前4位全是1，之后的8位全是0，最后的4位全是1。从十进制角度来看，这个序列是模糊的。从61455来看，完全不清楚哪些位是0，哪些位是1。而十六进制序列是二进制序列的镜像。第一个十六进制符号是F（即二进制中的1111），之后的两个十六进制符号是0，最后一个十六进制符号是F。

接着看例2，前3个4位组都是1000，最后一个4位组是0001。这在二进制中看着很简单，但在十进制中看着就很难了。十六进制提供了更清晰的表示，十六进制符号8对应于二进制的1000，十六进制符号1对应于0001！

我希望你能看出一种模式：二进制中每4位对应于十六进制中的1个符号。如果你还记得4位是半个字节的话，那么便可知道一个字节可以表示成两个十六进制符号。一个16位的二进制数可以用4个十六进制符号表示，一个32位的二进制数用8个十六进制的符号表示，以此类推。让我们以图1-5中的32位二进制数为例。

8A52FF00

1000 1010 0101 0010 1111 1111 0000 0000

图1-5 每个十六进制符号映射为二进制中的4位

在图1-5中，我们可以按每次半字节的节奏来消化这个相当长的数字，这是用十进制表示的相同数字（2320695040）无法办到的。

由于二进制与十六进制之间的转换相对容易，因此许多工程师通常会同时使用它们，只在必要时才会转换成十进制。后续在有意义的情况下，本书将会使用十六进制。

尝试不经过转换成十进制的中间步骤，把二进制转换成十六进制。

练习1-4：将二进制转换成十六进制

把下列用二进制表示的数字转换成等价的十六进制数。如果可以的话，不要先转换成十进制。本题的目标是直接从二进制转换成十六进制。

10（二进制）=_____（十六进制）

11110000（二进制）=_____（十六进制）

一旦你掌握了将二进制转换成十六进制的方法，就尝试一下从十六进制转换成二进制。

练习1-5：将十六进制转换成二进制

把下列用十六进制表示的数字转换成等价的二进制数。如果可以的话，不要先转换成十进制。本题的目标是直接从十六进制转换成二进制。

1A（十六进制）=_____（二进制）

C3A0（十六进制）=_____（二进制）

1.7 总结

本章讨论了一些计算机的基本概念。你知道了计算机是可以通过编程来执行一组逻辑指令的电子设备。你还了解了现代计算机是数字设备而不是模拟设备，并明白了二者之间的区别：模拟系统是那些用变化范围较大的值来表示数据的系统，而数字系统则是用符号序列表示数据的系统。之后，我们探究了现代数字计算机是如何只依赖于0和1两个符号的，并了解了仅含有两个符号的数字系统，即以2为基数的系统（或二进制系统）。我们还介绍了位、字节和标准SI前缀，利用它们，你可以更容易地描述数据的大小。最后，你了解到对使用二进制的人来说，十六进制非常有用的原因。

第2章将更仔细地研究二进制是如何用于数字系统的。我们将介绍如何使用二进制来表示各种类型的数据，以及二进制逻辑是如何工作的。

第2章

二进制

在第1章中，我们把计算机定义为一种电子设备，通过编程它可以执行一组逻辑指令。然后，我们大致了解了计算机中所有的内容（从它使用的数据到它执行的指令）是如何以二进制0和1的形式存储的。本章将阐述究竟要如何使用0和1来表示各种类型的数据。我们还将介绍如何把二进制应用于逻辑运算。

2.1 数字化表示数据

前面讲的是用二进制存储数字，具体来说，前面涉及的是如何存储正整数和零。但是，计算机是以“位”的形式来存储所有数据的，包括负整数、小数、文本、颜色、图像、音频和视频等。现在考虑一下如何用二进制来表示各种类型的数据。

2.1.1 数字文本

下面从文本开始说明如何用位0/1表示除数字以外的数据。在计算机中，文本是指字母、数字和相关符号的集合，也被称为字符。文本一般用于表示词、句子、段落等。文本不包括格式（粗体、斜体）。为便于讨论，我们把字符集限制为英文字母和相关字符。在计算机程序设计中，术语“字符串”通常也被用于指代文本字符序列。

记住文本的定义，我们需要表示的究竟是什么？我们需要表示A到Z，区分字母大小写意味着A和a是不同的符号。我们还需要表示像逗号和句号这样的标点符号，以及空格。我们也需要表示数字0～9。这里的数字可能会令人困惑，其实它指的是表示数字0～9的符号或字符，这与存储数字0～9是不同的。

像刚才描述的那样，如果把所有需要表示的不同符号加在一起，大约有100个字符。如果要用唯一的位序列来表示每个字符，每个字符需要多少位呢？6个位能给出64种不同的组合，这不够。7个位能给出128种不同的组合，足够用来表示100个左右的字符。不过，由于计算机通常是以字节为单位处理的，因此只有取整并使用完整的8位（即一个字节）来表示每个字符才有意义。用一个字节可以表示256个不同的字符。

那么，怎么用8位来表示每个字符呢？如你所想，已经存在用二进制表示文本的标准方法，我们马上就会讲到。但在这之前，有一点很重要，那就是：我们可以造出任何想要表示每个字符的方案，只要运行在计算机上的软件知道我们的方案即可。也就是说，在表示特定类型的数据时，有些方案优于其他方案。软件设计人员更喜欢常用操作易于执行的方案。

假设你要创造自己的方案，这个方案用位组来表示字符。你可能决定用0b00000000来表示字符A，用0b00000001来表示字符B，以此类推。这种把数据转换成数字格式的过程称为编码；解释这个数字数据的过程称为解码。

练习2-1：创建自己的方案来表示文本

定义一种用8位数字来表示大写字母A到D的方案，然后用你的方案把单词DAD编码成24位数字。这里的正确答案不止一个，示例答案请参阅附录A。额外小练习：用十六进制表示编码的24位数字。

2.1.2 ASCII

幸运的是，我们已经有几种标准方法来数字化表示文本，所以我们不必再自己发明了！美国信息交换标准码（American Standard Code for Information Interchange，ASCII）便是一种格式，每个字符7位，可以表示128个字符，不过每个字符通常用一个完整字节（8位）存储。使用8位而不是7位意味着有一个额外的前导位，该位为0。ASCII处理英文字符，另一个被称为Unicode的标准可以处理几乎所有语言所使用的字符，其中也包括英文字符。下面重点介绍ASCII。表2-1给

出了ASCII字符子集的二进制和十六进制值。前32个字符没有显示，它们是控制符，比如回车符和换行符，它们最初用于控制设备而不是存储文本。

练习2-2：ASCII编码和解码

（1）根据表2-1把下列单词编码为ASCII二进制和十六进制值，每个字符一个字节。记住，大写字母和小写字母的值不同。

❑Hello

❑5 cats

（2）根据表2-1解码如下单词。每个字符用一个8位ASCII值表示，为了清晰起见增加了空格。

❑01000011 01101111 01100110 01100110 01100101 01100101

❑01010011 01101000 01101111 01110000

（3）根据表2-1解码如下单词。每个字符用一个8位的十六进制值表示，为了清晰起见增加了空格。

❑43 6C 61 72 69 6E 65 74

表2-1 ASCII字符0x20到0x7F