第1章　电筒密谈

假若你才10岁，你的好朋友与你临街而住，而且你们卧室的窗户面对着面 。每天晚上，当父母像平常一样很早催你上床睡觉时，你可能还想与好朋友交流思想、发现、小秘密、传闻、笑话和梦想，没有人可以责备你，毕竟,渴望交流是大多数人的天性。

当你们卧室还亮着灯时，你和你的好朋友可以临窗舞动手臂、打手势或以身体语言来交流思想，但复杂一些的交流就有些困难了。而且一旦父母宣布“熄灯”，交流也就无法继续进行了。

如何联系呢？用电话吗？10岁的小孩子屋里有电话吗？即使有，你们的谈话可能被偷听。如果家里的电脑通过电话线联了网，它可能会提供无声的帮助，不过很不幸，它也不会在你的房间里。

你和朋友采用的方法是用手电筒。所有的人都知道手电筒是为孩子们藏在被窝里看书而发明的，它也适合在黑暗中用来交流。它无声无息，且光的方向性很好，不会从卧室的门缝中泄露而使家人起疑。

用手电筒的光可以交谈吗？这值得一试。一年级你就学过在纸上写字母和单词，把这种方法运用到手电筒上看起来也合情合理。你所需做的就是临窗而站，用光画出字母。画字母‘O’，就打开电筒，在空中画个圈，然后关上开关；字母‘ I’则是画竖直的一笔。但是你很快发现这种方法行不通，当你注视来去飞舞的光柱时，会发现在脑海中将它们组合起来不是件容易的事，这些光划成的圈圈杠杠太不准确了。

也许你曾经看过一部电影，影片中两个水手隔海用闪烁的光传递消息。在另一部电影中，一个间谍用镜子反射阳光向一间屋子中被俘获的同伙发送讯息。这就给了你启发，你起先设计一种简单的交流方法，使字母表中的每个字母与一定数目的闪烁相对应。 A闪一下，B闪两下，C闪三下，如此递推，Z就闪烁26下。BAD这个词由字母间有间隔的两闪、一闪、四闪组成，这样你不会误以为它是闪七下的字母G了。词间的停顿则比字母间的停顿时间稍长一些。

这看起来很有希望，采用这种方法的优点是你不需要在空中挥舞手电筒，只需对准方向按开关就行了；缺点是你试图发送的第一个消息（“How are you?”）就需要131次闪烁，更糟的是，你忘了定义标点符号，所以无法表示句尾的问号了。

这离问题的解决已经很近了，你想别人以前肯定也遇到过类似的问题，你解决它的思想一定是正确的。为了解决问题，白天的图书馆之行使你发现了神奇的摩尔斯电码（morse code），这正是你想要的，即使你不得不重新学习如何“写”字母表中的字母。

以下就是区别：在你发明的体系中，每个字母是一定数目的闪烁，从闪烁一下的 A到闪烁26的Z；而在摩尔斯电码中，有长短两种闪烁，当然，这会使摩尔斯电码更为复杂，但它在实际应用中却被证实是更有效的。那句“ How are you？”现在仅需32次而不是131次闪烁，而且这还包含了问号。

在讨论摩尔斯电码的工作原理时，人们并不说“长闪烁”、“短闪烁”，他们使用“点（dot）”和“划（dash）”，因为这样易于在印刷品上表示。在摩尔斯电码中，字母表中的每一

个字母与一个点划序列相对应，正如在下表中你所看到的：

A	·−	J	·−−−	S	···
B	−···	K	−·−	T	−
C	−·−·	L	·−··	U	··−
D	−··	M	−−	V	···−
E	·	N	−·	W	·−−
F	··−·	O	−−−	X	−··−
G	−−·	P	·−−·	Y	−·−−
H	····	Q	−−·−	Z	−−··
I	··	R	·−·		

尽管摩尔斯电码与计算机毫不相关，但熟悉它的本质却对深入了解计算机内部语言和软硬件的内部结构有很大的帮助。

在本书中，编码或代码（code）通常指一种在人和机器之间进行信息转换的系统（体系）。换句话说，编码便是交流。有时我们将编码看成是密码 (机密)，其实大多数编码并不是的。大多数的编码都需要被很好地理解，因为它们是人类交流的基础。

在《百年孤独》的一书的开篇，马尔克斯回忆了一个时代，那时“世界一片混沌，许多事物没有名字。为了加以区别才给事物各个命名。”这些名字都是随意的，没有什么原因说明为什么不把猫称为狗或不把狗称为猫。可以说英语词汇就是一种编码。

我们用嘴发出声音组成单词，这些词可以为那些听得到我们声音，理解我们所用语言的人所听懂，我们称这种编码为“口头语言”或“语音”。对写在纸上（或凿在石头上、刻在木头上或通过比划写在空气中）的词，还有一种编码方式，那就是我们在印刷的报刊，杂志和书籍上看到的字符，称之为“书面语言”或“文本”。在许多语言中，语音和文本间有很强的联系。例如在英语中，字母或一组字母与一定的读音相对应。

手势语言的发明帮助了聋哑人进行面对面的交流。这是一种用手和胳膊的动作组合来表达词语中的单个字母、整个词及其基本概念的语言。对盲人来说，他们可以使用布莱叶盲文（Braille）。这种文字使用凸起的点代表字母，字母串和单词。当谈话内容要被迅速地记录下来时，缩写和速记是很有用的。

人们在相互沟通时使用了各种不同的编码，因为在不同的应用场合，其中的一些较其他的更为简便。例如，语言不能在纸上存储，所以使用了文字；语言、文字不适合用来在黑夜中安静地传递消息，故摩尔斯电码是一个方便的替代品。只要一种编码可以适用于其他编码所不能适用的场合，它就是一种有用的编码。

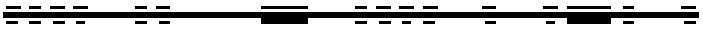
以后将看到，计算机中使用了不同的编码来传递和存储数字、声音、音乐、图像和视频（电影）。计算机不能直接处理人类世界的编码，因为它不能模拟人类的眼睛、鼻子、嘴和手指来接收信息。尽管这些年来计算机的发展趋势使我们的桌上电脑具有捕获、存储、处理和提供人类交流中所使用的各种信息的能，而且不论这些信息是视觉的（文字和图片）、听觉的（语言、声音及音乐）还是两者的混合（动画和电影）。所有这些信息都要求使用它们自己的编码方式，正如交谈需要使用人的某些器官（嘴和耳朵），而书写和阅读则需要使用另外一些

器官（手和眼睛）一样。

用手电筒发送摩尔斯电码时，电筒的开关快速地合开代表一个点，让电筒照亮稍长的时间则代表一个划。举例来说，发送字母A，要先快速地合开开关，然后再稍慢些合开。在发送下一个字母前要有短暂的停顿。约定划的时间大约是点的 3倍。例如，如果点的照亮时间为 1秒，那么划就是3秒。（实际上，摩尔斯电码的传递速度要快得多。）接收者看到了短闪和长闪就知道是A。

摩尔斯电码中点划之间的间隔是极为关键的。例如，发送字母 A时，点划之间的间隔应与一个点的时间大致相同（如果点的时间是 1秒，那么间隔的时间也是1秒）。同一个词中字母间间隔稍长，约为划的持续时间（或者 3秒，如果那是划的持续时间的话）。下面是单词“ hello ”对应的摩尔斯电码，图中示意了字母间的间隔（隙）：

单词之间相隔大约2倍于划的时间（如果划是3秒，那么间隔即为6秒）。下面是“ hi there ”对应的编码（码字）：



手电筒开和关的时间长度并没有限定，这取决于点的时间长度，点长又由手电筒开关触发的速度和摩尔斯电码发送者记忆电码的熟练程度来决定，熟练发送者的划也许与生手的点等长。这个小问题会使接收电码有些困难，但在一两个字母之后，接收者通常就可以辨认出哪个是点，哪个是划了。

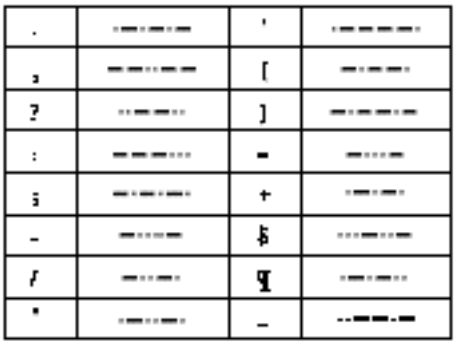
粗看起来，摩尔斯电码的定义 —— 这里所谓的定义是指与字母表中的字母相对应的各种点划序列 —— 与打字机字母的排列一样是随意的。但仔细观察后你会发现不完全如此，简短的码字分配给了使用频率较高的字母，例如 E和T，爱赌博的人和“ 财富之轮 ”爱好者可能一下就注意到了这个问题；不常用的字母如 Q和Z（它们在赌局中是 10点）则分配以较长的码字。

几乎所有人都知道一点儿摩尔斯电码，国际遇险信号 SOS的摩尔斯电码为“ 三点三划三点 ”。SOS并非缩写，选择它仅仅因为它有一个易记的摩尔斯电码序列。第二次世界大战中，英国广播公司选用贝多芬第五交响曲中的片段作为节目前奏 —— BAH、BAH、BAH、BAHMMMMM，听起来颇像摩尔斯电码中V（代表Victory）的码字。

摩尔斯电码的一个缺点是它没有对大小写字母进行区分。除表示字母外，摩尔斯电码还用5位长的码字来表示数字：

1	·—————	6	—····
2	··————	7	——···
3	···———	8	———··
4	····—	9	————·
5	·····	0	—————

这些数字的码字看起来还有些规律 (相对于字母对应的码字而言)。大多数标点符号的码字采用5位、6位或7位的码长：



对欧洲一些语言中的重音字母以及一些有特殊用途的缩写定义了特别的码字， SOS就是这样一个缩写：发送时每个字母的码字之间仅有一点的时间间隔。

如果有特制的用于发送摩尔斯电码的手电筒，你和朋友之间的交流就方便多了。这种手电筒除了常有的开关，还有一个按钮，按压按钮就可以控制电筒的亮灭。经过练习后，你们每分钟可以发送和接收5～10个单词。虽然仍比交谈慢（大概每分钟100个词左右）但已足够用了。

当你和朋友最终熟记了摩尔斯电码时（这是唯一精通发送接收的方法），你也可以用它代替日常用的语言。为了达到最高的速度，可以发“滴（dih）”音代表点、“嗒(dah)”音代表划。摩尔斯电码同样也可将文字简化为用点和划两个符号表示。

以上的关键在于“两”这个词 —— “滴、嗒”两个声音，“点、划”两种方式。实际上任何两种不同的东西经过一定的组合都可以代表任何种类的信息。

第2章　编码与组合

摩尔斯电码由萨缪尔·摩尔斯（1791—1872）发明，本书后面会在多处提到他。摩尔斯电码是随着电报机的发明而产生的，电报机我们以后也还要做详尽的说明。正如摩尔斯电码很好地说明了编码的本质一样，电报机也提供了理解计算机硬件的良好途径。

大多数人认为摩尔斯电码的发送易于接收，即使你没有记住摩尔斯电码，也可以方便地借助下面这张按字母顺序排列的表发送：

A	·−	J	·−−−	S	···
B	−···	K	−·−	T	−
C	−·−·	L	·−··	U	··−
D	−··	M	−−	V	···−
E	·	N	−·	W	·−−
F	··−·	O	−−−	X	−··−
G	−−·	P	·−−·	Y	−·−−
H	····	Q	−−·−	Z	−−··
I	··	R	·−·		

接收摩尔斯电码并将其翻译回单词比发送费时费力多了，因为译码者必须反向地将已编码的“滴-嗒”序列与字母对应。例如，在确定接收到的字母是“ Y”之前，必须按字母逐个地对照编码表。

问题是我们仅有一张提供“字母→摩尔斯电码”的编码表，而没有一张可供逆向查找的“摩尔斯电码→字母”译码表。在学习摩尔斯电码的初级阶段，这张译码表肯定会提供很大的便利。然而，如何构造译码表却毫无头绪，因为我们似乎无法找出这些按字母顺序排列的“滴-嗒”序列的规律。

那么忘记那些字母序列吧，也许按照码字中“滴”“嗒”的个数来排列会是个更好的尝试。例如，仅含一个“滴”或“嗒”的摩尔斯电码序列只可能代表E或T这两个字母之一：

·	E
−	T

两个“滴”或“嗒”的组合则代表了4个字母I、A、N、M：

··	I	−·	N
·−	A	−−	M

三个“滴”或“嗒”的序列代表了8个字母：

···	S	−··	D
··−	U	−·−	K
·−·	R	−−·	G
·−−	W	−−−	O

最后（如果不考虑数字和标点符号的摩尔斯电码），四个“滴”或“嗒”的序列则共代表了16个字母：

····	H	—···	B
···—	V	—··—	X
··—·	F	—·—·	C
··——	Ü	—·——	Y
·—··	L	——··	Z
·—·—	Ä	——·—	Q
·——·	P	———·	Ö
·———	J	————	Ch

四张表共包括2 + 4 + 8 + 16 ＝ 30个编码，可与30个字母相对应，比拉丁字母所需的26个字母还多了4个。出于这个原因，在最后一张表中，你可能注意到有4个编码与重音字母相对应。

在翻译别人发送的摩尔斯电码时，上面 4张表提供了极大的便利。当你接收到一个代表特定字母的码字时，按其中含有的“滴”“嗒”个数，至少可以跳到其对应的那张表中去查找。每张表中，全“滴”的字母排在左上角，全“嗒”的字母排在右下角。

你注意到4张表大小的规律了吗？每张表都恰好是其前一张表的两倍大小。这其中包含的意义是：前一张表的码字后加一个“滴”或加一个“嗒“，即构成了后一张表。

可以按下面的方式总结这个有趣的规律：

点划数	码字数
1	2
2	4
3.	8
4	16

四张表中每张码字数都是前一张的两倍，那么如果第一张表含2个码字，第二张表则含2×2个码字，第三张表2×2×2个码字。以下是另一种表达方式：

点划数	码字数
1	2
2	2×2
3	2×2×2
4	2×2×2×2

当然，如果遇到数的自乘，可以用幂表示，例如 2×2×2×2可以写成2^4。数字2、4、8、16分别是2的1、2、3、4次幂，因为可以用依次乘 2的方法将它们计算出来。由此我们的总结还可以写成下面的方式：

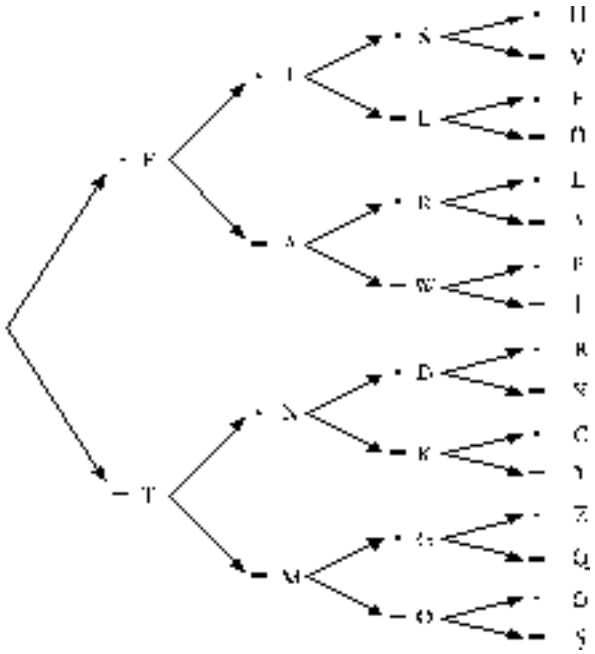
点划数	码字数
1	2^1
2	2^2
3	2^3
4	2^4

这张表简单明了，码字数是 2的次方，次方数目与码字中含有的“ 滴 ”“ 嗒 ”数目相同。我们可以把表总结为一个简单的公式：

码字数 = 2“滴”与“嗒”的数目

很多编码中都用到2的幂，在下一章中我们会看到另一个例子。

为了使译码的过程更为简便，可以画出如下一张树形图：



这张表表示出了由“ 滴 ”与“ 嗒 ”的连续序列得出的字母。译码时，按箭头所指从左到右进行。例如，你想知道电码“ 滴-嗒-滴 ”代表的字母，那么从最左边开始选择点，沿箭头向右选择划，接着又是点，得出对应的字母是 R，它写在最后一个点的旁边。

如果认真考虑，会发现事先建立这样一张表是定义摩尔斯电码所必需的。首先，它保证了你不会犯给不同的字母相同码字的错误！其次，它保证你使用了全部的可用码字，而没有使“ 滴 ”与“ 嗒 ”的序列毫无必要的冗长。

我们可以加长码字至 5位或更长，5位长的码字又提供了额外的 32（2×2×2×2×2或2^5）个码字。一般而言，这就足够 10个数字和16个标点符号使用。实际上，摩尔斯电码中的数字确实是5位的，但在许多其他编码方式中，5位码字常用于重音字母而不是标点符号。

为了包含所有的标点符号，系统必须扩充至 6位表示，提供64个附加编码，此时系统可表示2+4+8+16+32+64共126个字符。这对摩尔斯电码而言太多了，以至于留下许多“ 未定义 ”的码字。此处“ 未定义 ”指不代表任何意义的码字，如果在你接收的摩尔斯电码中有未定义的码字，就可以肯定发送方出了差错。

由于推出了下面这条公式：

码字数 = 2“滴”与“嗒”的数目

我们就可以继续导出更长的码字位数所代表的码字数目。很幸运，我们不必为确定码字数目而写出所有可能的码字，我们所要做的不过是不断地乘 2而已：

点划数	码字数
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

摩尔斯电码被称为二元码（binary code），因为编码中仅含“滴”和“嗒”。这与一个硬币很相似，硬币着地时只可能是正面或反面。二元事物（例如硬币）、二元编码（例如摩尔斯电码）常常用2的乘方来描述。

上面所做的对二元编码的分析在数学上的一个分支——组合学或组合分析里只能算是一个简单的练习。传统上，由于组合分析能够用来确定事件出现的几率，例如硬币或骰子组合的数目，所以它常用于概率统计，但它也同样有助于我们理解编码的合成与分解。

第3章　布莱叶盲文与二元编码

摩尔斯不是第一个成功地将书写语言中的字母翻译成可解释代码的人，他也不是第一个因为其编码而受到人们纪念的人，享有这个荣誉的是一个晚摩尔斯 18年出生的早慧的法国失明少年。虽然人们对他的生平所知甚少，但就是所知的这一些却足以给后人留下深刻印象。

路易斯·布莱叶1809年出生于法国的Coupvray，他的家乡在巴黎以东25英里，父亲以打造马具为生。 3岁时，在这个本不该在父亲作坊里玩耍的年龄，小布莱叶意外地被尖头的工具戳中了眼睛。由于伤口发炎，感染了另一只眼，他从此双目失明。布莱叶原本注定在贫困潦倒中度过一生（正如那时大多数盲人一样），但他的聪明才智和求知欲不久即显露了出来。在本地牧师和一位学校老师的帮助下，布莱叶和其他孩子一道上了学， 10岁那年又前往巴黎的皇家盲人青年学院学习。

盲人教育的一大障碍就是他们无法阅读印刷书籍。Valentin Haüy(1745—1822)，巴黎学校的创始人，发明了一种将字母凸印以供触摸阅读的方法。但这种方法使用起来较为困难，并且只有很少的书籍用这种方法“制造”。

视力正常的Haüy陷入了一种误区。对他而言，字母A就是A，它看起来（或感觉起来）也必须像是个A。（如果给他手电筒作为交流工具，他也会试图在空气中画出字母的形状，而我们已经知道这种方法并不有效。）Haüy也许没有意识到一种与印刷字母完全不同的编码会更适于盲人使用。

另一种可选的编码有一个出人意料的起源。法国陆军上尉 Charles Barbier在1819年发明了一种他自称为écriture nocturne的书写体系，这种体系也被称为“夜间文字。他使用厚纸板上有规律凸起的点划来供士兵们在夜间无声地传递口信（便条），士兵们使用尖锥状的铁笔在纸的背面刺点和划，凸起的点可以用手指感觉阅读。

Barbier体系的问题是其过于复杂。 Barbier没有用凸起的点来代表字母表中的字母，而是用其代表声音。这样的系统中一个单词通常需要许多码字表达。这种方法在野外传递短小消息还算有效，但对长一些的文章而言则有明显不足，更不要说是整本的书籍了。

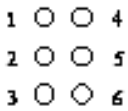
布莱叶在12岁时就熟悉Barbier方法了，他喜欢使用这些凸点，不仅因为它们易于用手指阅读，更因为它们易于书写。教室里拿着铁笔和纸板的学生可以记笔记供课后阅读。布莱叶勤奋地工作试图改进这种编码系统。不出 3年（在他15岁时），他创建了自己的系统，其原理直到今天还在使用。布莱叶系统有很长时间仅局限在他所在的学校使用，后来它逐渐扩散到世界各地。1835年，布莱叶染上了结核病。 1852年，在他43岁生日过后不久，他便去世了。

时至今日，布莱叶系统的改进版本甚至可以与有声录音带竞争，它为盲人提供了与书写世界联系的途径。布莱叶方法仍是适于既聋又盲的人阅读的唯一方法。近来年，随着电梯和

自动语言机的普及，布莱叶系统更加广为人知。

本章将剖析布莱叶编码的编码方法及其工作原理，不过不必真正学习布莱叶编码或记住任何东西，我们只要大概了解一下编码的本质就行了。

布莱叶编码中，普通书写语言的每个字符 —— 具体而言如数字、字母和标点符号 —— 都被编码成局限在2×3小格中一个或多个凸起的点。这些小格一般被标记为 1～6：

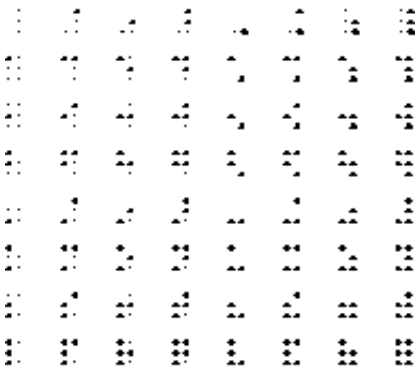


在当今实际使用中，特殊的打字机或刻印机可以在纸上打出布莱叶编码中的小点。

由于在书中夹印几页布莱叶编码极其昂贵，我们使用了在通常印刷品中常用的布莱叶码的表示方法。在这种表示方法中，小格中的 6个点全部印刷出来，大点代表小格中的凸起点，小点则代表平滑的点。例如下图中的布莱叶字母中，点 1、3、5是凸起的，点2、4、6则没有：

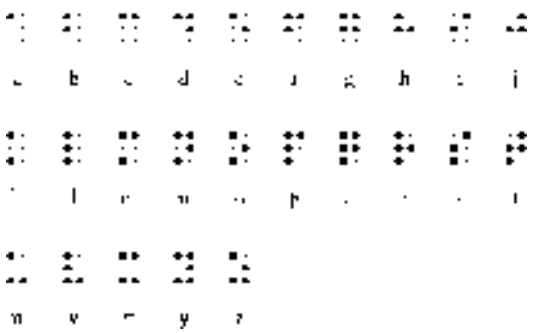
在这里吸引我们的问题是：点是二元的。一个特定的点不是凸起的就是平滑的，那么 6个点的组合数目就是2×2×2×2×2×2，或64(2^6)。

因此，布莱叶编码系统可以代表 64个不同的码字。以下就是所有的 64个码字：



如果我们发现布莱叶编码只用了 64个码字中的一部分，我们会疑问为什么 64个码字中有一些不被使用；如果发现布莱叶编码使用了多于 64个的码字，则又会让人怀疑我们是否神志清醒或数字计算的真实性，2乘2是等于4吗？

分析布莱叶编码，还是从基本的小写字母开始：



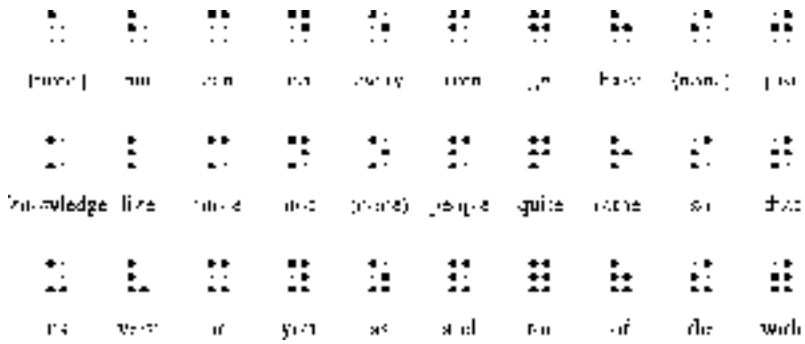
举例来说，短语“ you and me ”在布莱叶编码中看起来是这样的：

注意，代表同一个单词中的字母的小格用一个小距离分隔，大一些的距离（一般是没有凸点的小格）用来分隔不同的单词。

这就是布莱叶发明的布莱叶编码的基础，布莱叶还为法文中出现的重音字母设计了码字。注意，W没有对应的码字，这时由于在古法语中没有 W（不必担心，这个字母最终还是会露面的）。这样算来，我们仅使用了64个码字中的25个。

通过仔细的检查，会发现上面的布莱叶编码存在特定的规律。第 1行（从字母a～j）只用了小格的上面4个点 —— 点1、2、4、5；第2行除了点3凸起外其余都与第1行相同，第3行则除了点3、6凸起外其余都与第1行相同。

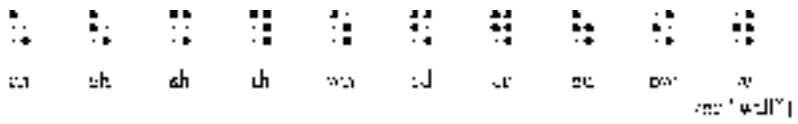
在布莱叶之后，布莱叶编码在许多方面有了扩展，现在大多数英语出版物所使用的系统是二级布莱叶码。二级布莱叶码采用了许多缩写来简化编码树以提高阅读速度。以下的三行（包括“ 完整的 ”第3行）显示了下面这些词的码字：



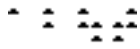
因此，在二级布莱叶码中，短语“ you and me ”被写成如下形式：

到现在为止，已描述了 31个码字 —— 词间没有凸起点的空格和三行每行 10个用于字母和单词的码字。这离理论上可用的 64个码字还相距甚远。不过我们将要看到，在二级布莱叶码中，没有任何浪费的码字。

首先，我们使用a～j的编码加上凸起的6号点。它们代表词中的缩写，这其中包括 W和另一个词的缩写：

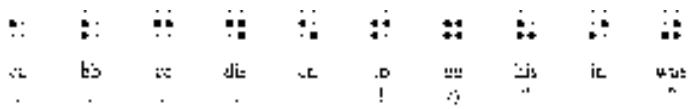


举例来说，“ about ”可以用二级布莱叶码写成如下形式：



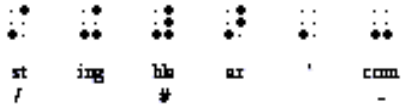
其次，可以把代表字母a～j的码字中的点下移一行，即仅使用点 2、3、5和6。这些码字根

据上下文代表标点符号或缩写：

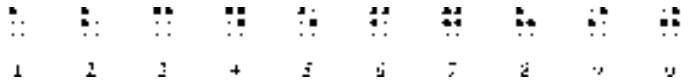


头4个码字代表逗号、分号、冒号和句号。注意左括号和右括号用同一个码字代表，但左引号和右引号则使用了不同的码字。

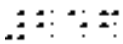
已经有51个码字了。接下来的6个码字使用点3、4、5、6尚未使用的组合来表示缩写和几个额外的标点符号：



“ble”的码字非常重要，因为当它不是单词的一部分时，它表明其后跟随的码字要被翻译成数字，这些数字的编码与a～j的编码相同：



由此，如下码字的序列代表数字256：



如果你一直在计数的话，我们还需要7个码字才能达到总计的64个码字。下面就是剩余的7个码字：

第一个（点4凸起）是重音字母标识符，其余的作为一些缩写的前缀，也用于其他用途：点4、6凸起时（本行的第5个码字），该码字代表数字中的小数点或强调标识符，这由上下文决定。点5、6凸起时，码字则是与数字标识对应的字母标识。

最后（也许你正在疑惑布莱叶编码如何表示大写字母），我们用6号点来作为大写标识，它表明其后跟随的字母是大写的。例如，可用如下的码字写出该编码创始人的名字：

这包含大写字母标识、字母l、缩写ou、字母i和s，空格，另一个大写字母标识，字母b、r、a、i、l、l和e（在实际应用中，该名字还可以再删掉最后两个不发音的字母）。

总结一下，我们已经看到了6个元素（凸点）如何恰好形成64个码字。这64个码字根据上下文大多有双重含义，其中有数字标识以及取消数字标识作用的字母标识。这些标识改变了跟随其后的码字的含义——从字母变数字或从数字变字母。起这种作用的码字常被称为“先行码/前置码”或“转义码”，它们更改其后字符的含义直至更改作用被取消。

大写标识表示其后的字母（也仅有字母）应写成大写，这种码字被称为“换码代码”。“换码代码”使你“避免”那种单调的、常规的码字解释，而转入一种新的解释方法。在以后几章中可以看到，当把书面语言转换为二元码字时，“换码代码”和“转义码”的使用是很普遍的。

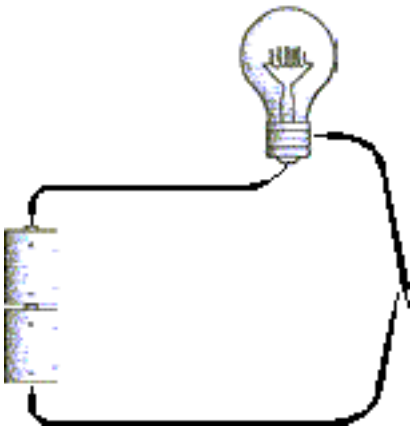
第4章　手电筒剖析

手电筒的用途极为广泛，用于在黑暗的遮盖物里阅读和用于发送编码消息只是两个用途最明显的方面。最普通的家用手电筒也能在教学演示中说明神秘物质电（ electricity）时扮演中心角色。

电是一种令人称奇的现象，尽管它已得到普遍应用，但依然还保持着很大的神秘性，即使对那些自称已经弄清楚它的工作原理的人而言也是这样。但恐怕不管怎么样，我们都必须好好努力钻研一下电学。幸运的是，我们只需要明白一小部分基本概念就可以理解它在计算机中是怎样应用的。

手电筒当然是一种大多数家庭都拥有的较简单的电器。拆开一支有代表性的手电筒，你会发现它包括一对电池，一个灯泡，一个开关，一些金属片和一个把所有零件装在一起的塑料筒。

只用电池和灯泡，就可以自己做一个简单的手电筒。当然，还需要一些短的绝缘线（末端的绝缘皮除掉）和足够多的连接物：



注意上图右边两个松开的线端（头），那就是开关。如果电池有电并且灯泡也没有烧坏的话，接触两个线端，灯就亮了。

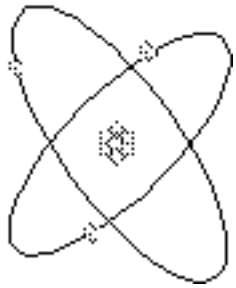
这是我们要分析的头一个简单电路，首先要注意的是电路是一个回路。只有从电池到电线、到灯泡、到开关、再回到电池的路径是连续畅通的，灯泡才会亮。电路中任何一点断开都会引起灯泡的熄灭。开关的目的就是控制电路开闭这个过程。

电路环接的特性提示我们有某种物质在电路中循环移动，可能与水在水管里流动有某些相似。“ 水与水管 ”的类比常用来解释电的工作机理，但最终它也像其他类比一样不可避免地解释不下去了。电在宇宙中是独一无二的，必须用它的术语来解释它。

在对电的工作的理解中，最流行的科学理论是电子理论（electron theory），该理论认为电起源于电子的运动。

众所周知，一切物质 —— 我们能看到、感觉到的东西 —— （通常）是由极其微小的被称为

原子的东西构成。每一个原子是由三种微粒构成的，即中子、质子和电子。你可以把原子画成一个小的太阳系，中子和质子固定在原子核内而电子像行星环绕太阳一样围绕原子核运动：



需要解释一下的是该模型与你在一个放大倍数足够大的显微镜下看到的真正原子不是一模一样的，它只是一个示例模型。

图中原子包含3个电子、3个质子和4个中子，说明这是一个锂原子。锂是已知的112种元素之一，它们的原子序数由1～112。一种元素的原子序数是指元素的原子核中质子的个数，通常也是其电子数。锂的原子序数为3。

原子能够通过化学合成形成分子，分子与组成它的原子的性质通常是不同的。比如水分子包含两个氢原子和一个氧原子（即H_2O）。显然水既不同于氢气，也不同于氧气。同样，食盐分子由一个钠原子和一个氯原子构成，而钠和氯都不可能成为法国馅饼的调味品。

氢、氧、钠、氯都属于元素，水和食盐都属于化合物。但是盐水是一种混合物，而不是化合物，因为其中水和食盐都保持它们各自的性质不变。

一个原子的电子数通常等于其质子数。但在某种特定环境下，电子能从原子中电离出来，这样电就产生了。

单词electron和electricity都源于古希腊词ηλεκτρον(elektron),你可能猜它的意思就是“极其微小而不可见的东西”。但事实并非如此——ηλεκτρον的真正意思是“琥珀”，一种玻璃状的硬质树液。这个看似不相关的词源来自于古希腊人所做的实验，他们用琥珀与木头相摩擦而产生我们今天所说的静电。在琥珀上摩擦木头使木头从琥珀获得电子，结果木头所含的电子数多于质子数而琥珀所含的电子数小于质子数。在更多的现代实验中，地毯能从鞋底获得电子。

质子和电子具有带电荷的特性，质子带正电荷（+）、电子带负电荷（−）。中子是中性的，不带电。即便我们用加减号来标明质子和电子，但符号并不表示算术运算中的加号和减号的意思，也不表示质子拥有某些电子所不具备的东西。使用这些符号仅仅表示质子和电子在某个方面性质相反。这个相反的特性也正表明了质子和电子是如何相互关联的。

当质子数与电子数相等时，它们是最适合和最稳定的。质子数与电子数的不平衡会导致它们趋于平衡。静电火花就是电子运动的结果，是电子从地毯通过你的身体再流回到鞋子的过程引起的。

描述质子和电子关系的另一条途径是注意观察异电性相吸同电性相斥的现象，但光凭看原子结构图我们是不能猜想到的。表面上看原子核中挤在一起的质子是互相吸引的。质子是通过比同性斥力大的某种力聚合在一起的，这种力叫强内力。释放核能的原子核裂变就是由于强内力导致的。本章只讨论通过得失电子获得电（电能）的问题。

静电不只存在于手指触摸门把手时闪出的火花之中。暴风雨时，云层的下层积累电子而

云层的顶层失去电子，闪电的瞬间，电子的不平衡马上消失。闪电正是大量的电子迅速从一端转移到另一端的结果。

手电筒电路中的电能显然比电火花或闪电之中的电能要好利用得多。灯泡能稳定持续地亮是因为电子并不是从一点跳到另一点。当电路中的一个原子把一个电子传给邻接的另一个原子时，它又从另一个邻接的原子获得电子，而这个原子又从它的一个邻接原子获得电子，如此依次循环。可见电路中的电就是从原子到原子的电子通路。

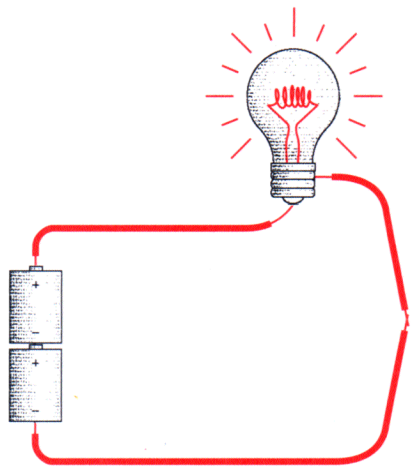
这不可能自发形成。仅仅只把一些破旧的电路材料连接在一起是不可能有电能产生的，需要某种可以激发电子环绕电路移动的物质。再分析一下前面所画的简单手电筒电路图，可以肯定激发电子运动的既不是电线，也不是灯泡，那么最有可能的就是电池了。

几乎每一个人都多少了解手电筒里所用电池的类型方面的一些知识：

- 它们都呈管状，且大小不同。比如有D、C、A、AA和AAA等型号。
- 无论电池大小怎样，它们都被标有“ 1.5伏 ”。
- 电池的一端是平的，标有一个负号（－）；另一端中间有一个小突起，标有一个正号（+）。
- 要想设备正常工作，就要正确安装电池（注意电池极性）。
- 电池的电能最终将用尽。有的电池可以充电，有的不行。
- 由此可以猜测，电池是用某种奇特的方式产生电能。

所有的电池中都发生着化学反应，一些分子裂变成其他分子或者结合形成新的分子。电池中有化学物质，这些化学物质就是用来起反应，从而在标有（－）的电池的一端（称为负极或阴极）产生多余的电子而在电池的另一端（称为正极或阳极）需要得到电子。这样，化学能转化为电能。

只有当某种特别的电子通过某条途径从电池负极出发，然后再传送到正极时，化学反应才能发生。因此假如一节空电池放在那里，那么什么事也不会发生（事实上，化学反应还是在进行的，只是速度极慢）。只有一条电路能将电子运离负极又为正极提供电子时，反应才会发生。电子在下图电路中是沿逆时针方向运动的：

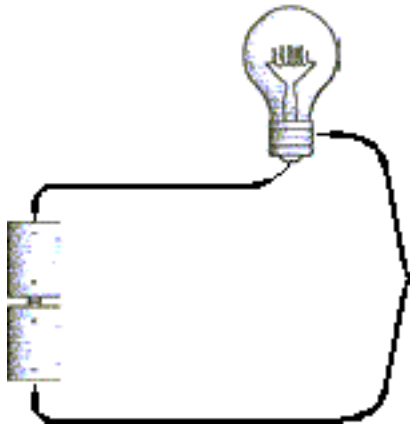


如果不是基于这个简单的事实：所有的电子，不管来自什么地方，都是一模一样的，否则，来自电池的化学物质里的电子就不可能如此随意地与铜导线的电子混合在一起的。铜导

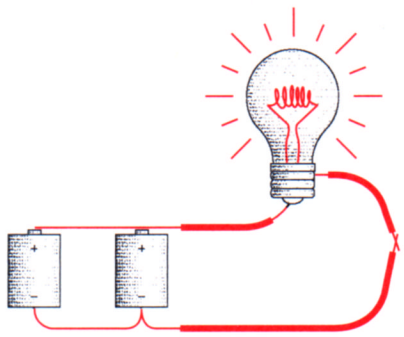
线的电子与任何其他电子是没有区别的。

注意，两个电池都是向着同一个方向。放在下面的电池的正极从上面电池的负极获得电子，这样两个电池就好像结合形成了一个更大的电池，这个大电池一端为正极，另一端为负极，其电压是3伏而不是1.5伏了。

如果把电池中的一个倒置，电路就会连不通，如下图所示：



在化学反应中，两个电池的正极都需要获得电子，但由于它们相互接触，电子无法通过某种途径到达它们。如果两个电池的正极连上了,那么它们的负极也应该连上，如下图所示：



这样的电路还是能连通。电池的这种连接方法称为并联，前一种连接方法称为串联。并联后的电压与单个电池电压同样都是 1.5伏。并联后的灯仍然可能亮，但不如串联时亮度大，不过电池的寿命将会是串联时的两倍。

通常认为电池为电路提供电能，但同样也可以认为电路为电池化学反应的发生创造了条件。电路将电子从负极传送到正极。电路中的化学反应将一直进行到所有的化学物质耗尽，这时你就需要换电池或是给电池充电了。

电子从电池的负极到正极流经了导线和灯泡。为什么需要导线？电不能通过空气传导吗？噢，可以说能，也可以说不能。电能够通过空气导通（尤其是潮湿的空气），否则也观察不到闪电。但电不能很轻易地流经空气。

一些物质的导电能力比其他物质的导电能力明显要好。元素的导电能力取决于它内部的原子结构。电子绕核旋转是在不同的轨道上的，这些轨道称为层。最外层只有一个电子的原子最容易失去那个电子，这正是导电需要具备的性质。这些物质易导电因而被称为导体。铜、

银和金都是良好导体，这三种元素位于元素周期表的同一列不是巧合。铜是最常用的导线材料。

导电物质的对立物质称为绝缘物质。一些物质阻碍电的能力比其他物质阻碍电的能力强，这种阻碍电的能力称为电阻。如果一个物质有很大的电阻——说明它根本不能导电——它就被称为绝缘体。橡胶和塑料都是很好的绝缘体，因而它们常用来做电线的绝缘皮。在干燥空气的情况下，布料和木材也是很好的绝缘体。其实只要电压足够高，任何物质都能导电。

铜的电阻很小，但它仍有电阻。导线越长，电阻越大。如果你用数里长的导线连接手电筒，导线的电阻将会大得令手电筒不亮。导线越粗，电阻越小，这可能有点违反直觉。你也许认为粗的导线需要更多的电来“充满它”。而事实上，导线越粗，电子越容易通过它。我已经提到过电压，只是还没有给出它的定义。一节电池为1.5伏特意味着什么呢？实际上，电压——得名于Count Alessandro Volto(1745—1827)，他于1800年发明了第一节电池——是初等电学中较难理解的概念之一。电压表征电势能的大小，无论一节电池是否被连通，电压总是存在的。

假设有一块砖头。如果把它放在地上，它的势能很小。当你把它举起至离地面4英尺高时，它的势能就增加了。你只要把砖块扔下，就能感觉到势能的存在。当你在一座高楼的顶层举着砖块时，它的势能更大。上面三个例子里，你只是拿着砖块而什么也没做，但砖块的势能却不同。

电学里更早的一个概念是电流。电流取决于电路中飞速流动的电子的数量。电流用安培来度量，它得名于André Marie Ampére(1775—1836)，一般简称安，比如“10安的保险丝”。当6 240 000 000 000 000 000个电子在1秒内流过一个特定的点时，就是1安培电流。

用水和水管作个类比。电流与流经水管的水量很相似，而电压类似于水压，电阻类似于水管的粗细程度——水管越小、阻力越大。因此水压越高，流过水管的水量越大；水管越小，流过它的水量就越少。流过水管的水量（电流）与水压（电压）成正比而与水管的阻力（电阻）成反比。

在电学中，如果知道电压和电阻的大小，就可计算出电流的大小。电阻——物质阻碍电流通过的能力——用欧姆度量，得名于Georg Simon Ohm (1789—1854)，他提出了著名的欧姆定律，定律中表述

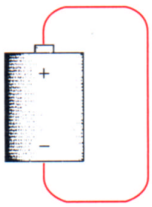
$$I=E/R$$

这里I表示电流，E表示电压，R表示电阻。

举个例子，让我们看一节空置的电池：

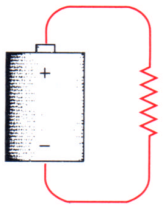
它的电压E为1.5伏，这是电势能。因为电的正负两极只被空气导接，因而电阻（用R表示）非常、非常大，这就意味着电流I等于1.5除以一个巨大的数，电流几乎为0。

现在用一根短铜导线连接电池的正负两极（从现在开始，本书中导线外的绝缘皮不再表示出来）：



我们已经知道这是短路。电压仍是 1.5伏，但电阻很小，这时电流等于 1.5除以一个很小的数，也即意味着电流很大。很多很多的电子将流过导线。实际上，电流将受到电池物理大小的限制。电池不可能导通如此大的电流，且实际电压也将低于 1.5伏。如果电池足够大，导线将会发热，因为电能转化为了热能。如果导线变得很热，它将会发光（辉光放电）甚至可能熔化。

绝大部分电路都介于这两个极端之间。可以把它们统一表述为如下图：



电气（子）工程师用折线来表征电阻。这里它表示电阻不是特别大，也不是特别小。

如果导线的电阻很小，导线将发热发光，这就是白炽灯的工作原理。白炽灯泡是由美国最著名的发明家托马斯·爱迪生（ 1847—1931）发明的。在他致力于发明灯泡的时候（ 1879年），这个思想已被普遍接受并且同时还有不少其他发明家在研究这个问题。

灯泡里的细线叫灯丝，通常用金属钨做成。灯丝的一端连在基座底部的尖端，另一端连在金属基底的一个侧面，用一个绝缘体将它与尖端分开。细线的电阻使它发热。如果暴露在空气中，钨就会由于达到燃烧温度而烧起来。但在灯泡的真空里，钨丝就发亮了。

大多数普通手电筒用两节电池组成一组，总电压是3.0伏。且选用电阻大约为4欧姆的灯泡。这样，电流等于3除以4即0.75安培，也就是750毫安。这就意味着每秒钟有 4 680 000 000 000 000 000个电子通过灯泡。（注意，如果你用欧姆表直接测量手电筒灯泡的电阻，你只会得到一个比4欧姆小得多的结果。这是因为钨的电阻还与它的温度有关系，温度越高，电阻越大。）

你可能已经发现，你买回家的灯泡上标记了特定的瓦特数。瓦特这个名词取自于著名的蒸气机发明家詹姆斯·瓦特（1736—1819）。瓦特是功率P的单位，它用下式计算

$$P=E\times I$$

手电筒是3伏，0.75安培，那么灯泡的功率就要求 2.25瓦特。

家用照明灯大约为100瓦特，这是为家用电压120伏设计的。在这种情况下，电流为 100瓦除以120伏即大约0.83安培。因此，100瓦特灯泡的电阻为120伏除以0.83安培即144欧姆。

到此，我们大致分析了手电筒的每一个组成部分 —— 电池、导线和灯泡。但是我们遗漏了一个最重要的部分、对，是它的开关。开关控制电路的开闭。当开关允许电流动时，我们说它是开的或合上的，而关的或断开的开关是不允许电流动的。（这里所表示的开、关的状态正好与门相反，合上的门不允许事物通过的，而合上的开关允许电通过。）开关或开或关，电流或有或无，灯泡或亮或不亮，就像摩尔斯和布莱叶发明的二元码一样，简单的手电筒或亮或不亮，它没有中间状态。二元码与电气电路之间的相似性将在后面的章节中起很大作用。

第5章　绕过拐弯的通信

你12岁了。一天，你最要好的朋友一家要搬到另一个镇上去了。此后，你经常和他在电话里聊天，但电话交谈与那些后半夜的手电筒摩尔斯电码会话完全不一回事。住在你隔壁的另一个好朋友最终成为你新的最要好的朋友。现在到了该教你的新朋友一些摩尔斯电码，让后半夜的手电筒重新亮起来的时候了。

问题是你的新朋友的卧室窗户与你的不是面对面的。房子是挨着的，卧室的窗户都朝着同一个方向。除非你想办法在室外支起一些镜子，否则手电筒现在是不能适用来在黑夜中通信的。

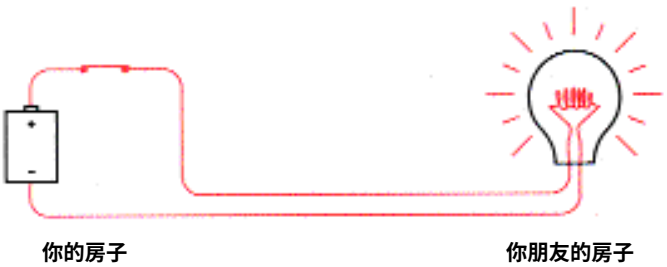
怎么办呢？

现在，你可能已经知道有关电的一些知识了，因此你决定用电池、灯泡、开关和导线来做自己的手电筒。最初的实验中，你在你的卧室里接好电池和开关。两条导线接出你的窗子，跨过篱笆，再接进你朋友的卧室，并在那里再连好灯泡：

尽管图中只示意了一节电池，但实际上你可能得用两个。在下面和以后的图中，用下图表示断开的开关：

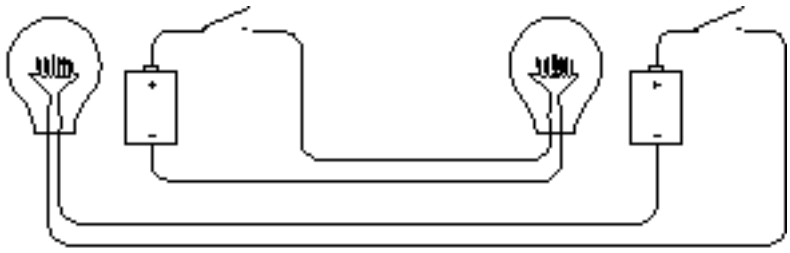
用下图表示闭合的开关：

本章的手电筒与上一章中手电筒的工作原理是相同的，尽管本章的手电筒中连接组件的导线要长得多。当你闭合开关时，你朋友那边的灯泡就亮了：



现在你可以用摩尔斯电码来发送消息了。

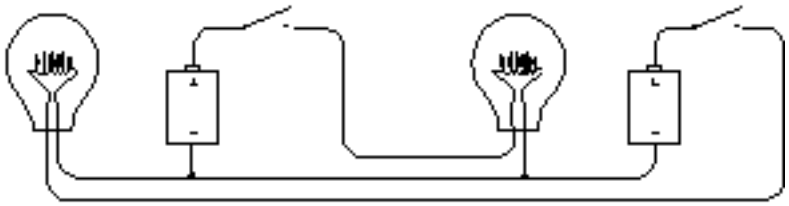
一旦有一个手电筒起作用，你可以做另一个远距离手电筒，好让你的朋友可以发送消息给你:



你的房子　　你朋友的房子

祝贺你! 你已经装上了一个双向电报系统。你可能注意到这两个相似的电路彼此完全独立而没有联系。理论上，你可以给你的朋友发送消息而同时你的朋友也可以给你发送消息（尽管对于你的大脑而言，同时阅读和发送消息可能比较困难）。

聪明的你发现如下改装电路能让你节省 25%的导线：

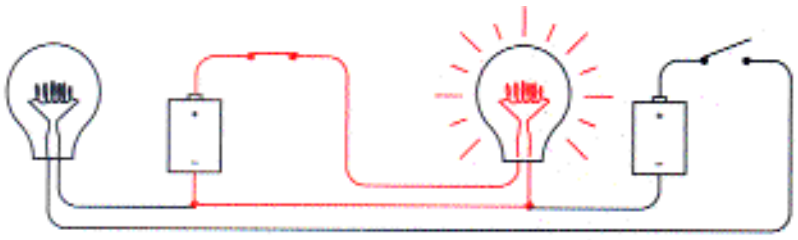


你的房子　　你朋友的房子

注意，现在两个电池的负极接在一起了。两个回路（电池到开关到灯泡再到电池）仍是独立工作，尽管它们连在一起像连体双胞胎。

这种连接叫公用连接。在这个电路中，公用部分从左端灯泡和电池的接合点直到右端灯泡和电池的接合点。图中接合点用黑点标记出来了。

进一步分析一下。首先当你按下开关，你朋友那边的灯就亮了。图中浅色回路中有电流流过：

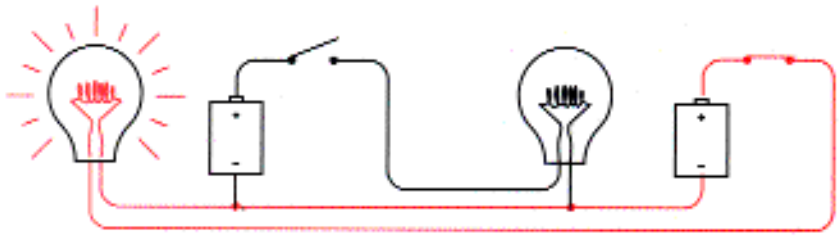


你的房子　　你朋友的房子

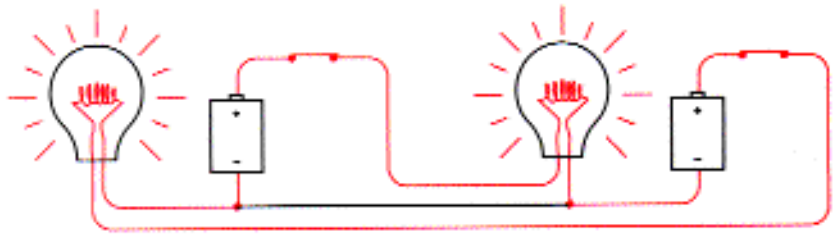
在电路的其余部分里没有电流流过，因为没有了可让电子通过的回路。

当你不发消息而你的朋友发消息时，你朋友房间里的开关控制你房间里灯泡的亮灭。在

下图浅色回路中有电流流过：



当你和你的朋友想要同时发消息时，有时两个开关同时断开，有时一个断开一个闭合，有时两个同时闭合。在最后一种情况下，电路中电的流动如下图所示：



公用部分(两个接合点之间)没有电流流过。

通过公用部分把两个独立电路连接成一个电路，已经把两栋房子之间的四条导线减少到了三条，也即减少了25%的导线开支。

如果不得不接很长距离的线路，我们可能会想到再减少一根导线。但不幸的是对于 1.5伏的D号电池和小灯泡，这是不合适的。如果用的是 100伏的电池和大得多的灯泡时，那就有办法了。

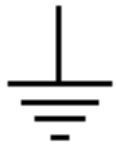
这是个窍门：如果你要搭建电路的公用部分，你不需要任何导线。你可以用另外某种东西取代它。你所用的取代物是一个直径大约为 7900英里，由金属、岩石、有机物等多为无生命的物质组成的巨大球体。它就是地球。

上一章描述的良导体中有银、铜和金。事实上，地球不是一个很好的导体，尽管某些部分（如沼泽）的导电性能比其他部分（如干沙漠）要好得多。但我们知道导体越大越好，一根很粗的导线比一根很细的导线要强得多。这是地球的优势，它的确非常非常大。

要用地球做导体，并不是把一根小细线插到马铃薯旁边的地里就可以了。你还必须使用某种东西以维持和地球的真正接触，这也就是需要一个大面积的导体。一个很好的解决办法是用一根至少8英尺长，1/2英寸粗的粗铜柱，它能提供与地面 150平方英寸的接触。你可以用一个大锤子把它砸进地下，然后再接一根导线。如果你家的水管是铜质的，且从房子外的地下接进来的话，那么你只要把一根导线与水管相连就可以了。

与地球的电性连接（也就是我们常说的接地）在英国叫 earth,在美国叫ground。用ground可能会引起一点点儿误会，因为它也经常用来指电路的公用部分。本章除非特别声明，否则ground都指与地球的物理连接。

画电路图时常用下面这个符号表示接地：



电气工程师们使用这个符号是由于他们不喜欢费时间画一个埋在地下的 8英尺长的铜柱。让我们来看看它是怎么工作的。从分析单回路开始：

如果你使用的是高压电池和大灯泡，你只需要在你和你朋友的房子之间接一根导线，因为你可以用大地来做导体：

当你断开开关，电子的流动如下图所示：

电子从你朋友房子的地下出发，通过灯泡、导线和你房间里的开关，然后进入电池的正极。电子由电池的负极进入地下的。

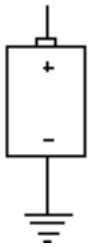
也许你还真的很想看到电子从埋在你家后院的 8英尺长的铜柱进入地下，飞速地通过大地到达埋在你朋友家后院的铜柱。

但是当你考虑到地球在为世界上数以千计的电路完成此功能时，你也许会问：这些电子怎么知道该到哪儿去呢？显然它们不知道。这里要用地球的一个特殊性质来解释。

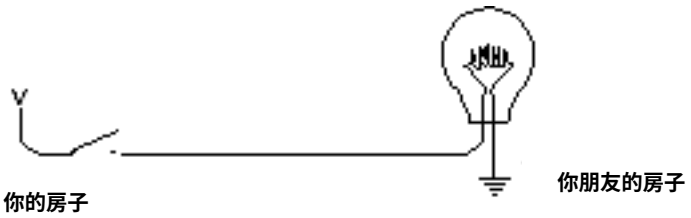
是的，地球是一个巨大的导体，但它同时也是电子的来源和仓库。地球对于电子而言就好像大海对于水滴而言。地球的确是电子无尽的源头，也是电子巨大的存储池。

但是地球也有电阻，这就是为什么如果用 1.5伏的D号电池和手电筒灯泡就不能用接地来减少电路开支的原因。地球对于低电压电池而言电阻实在太大了。

你可能注意到上面两张画了电池的图中，电池的负极接地了：



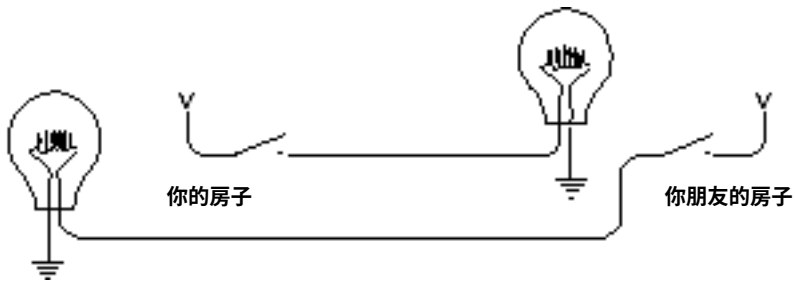
以后将不再画接地的电池，而用代表电压的字母 V来代替它。单回路灯泡电报机现在如下图所示：



V代表电压，但它也可以表示吸取器。把V看成电子吸取器，把大地看成电子的海洋，电子吸取器从地下吸取电子，放入回路，使之工作（比如点亮灯泡）。接地有时也被看成零电势，意味着没有电压存在。电压 —— 像早先解释的 —— 是一种电势能，就像悬浮的砖块具有势能一样。零电势就好像摆在地上的砖块 —— 它不能再往什么地方掉下去了。

在第4章中，我们注意到的一件首要的事情是电路是一个回路。新电路看起来一点儿都不像回路，但它仍然是回路。你可以用负极接地的电池代替 V，然后用一根线把所有有接地符号的地方连起来，你将得到与本章开始时一样的电路图。

因此，通过一对铜柱（或是自来水管）的帮助，可以只用两根跨越你和你朋友房子之间篱笆的导线就建立起了双向摩尔斯电码系统：



这个电路与先前的三线配置电路功能相同。

本章已经迈出了通信改革中的关键性一步。最初，我们只能通过直线视觉和在手电筒的可见范围内进行摩尔斯电码通信。

使用电线，不仅突破了直线视觉的限制，而且通过建立系统来绕过拐弯进行通信，我们还摆脱了距离的限制。只要搭造更长更长的线路，就可以越过成百上千英里进行通信。

对了，这还不太准确。尽管铜是电学上很好的导体，但它不是最完美的。导线越长，电

阻越大；电阻越大，电流越小；电流越小，灯泡越暗。

那么导线可以造多长呢？因情况而定。假设你正在使用原来四根线的双向电路，无接地和公用，并且还用手电筒和灯泡。为了节省开支，你先从电器行买了一些 20号规格的电话线，每100英尺$ 9.99。电话线是用来连接你的扩音器和立体声系统的。它有两根导线，因此它是电报系统的上佳选择。如果你的卧室与你朋友的卧室不到 50英尺远，只用一捆电话线就够了。

美国的导线粗细规格为 AWG。AWG数越小，导线越粗，电阻越小。你所买的 20号规格电话线直径大约 0.032英寸，每1000英尺大约10欧姆电阻，这样对于卧室之间 100英尺长的回路电阻为1欧姆。

这并不坏，但如果要连上英里的线呢？线的总电阻将达到 100欧姆以上。回想一下上一章中，灯泡电阻仅为 4欧姆。利用欧姆定律，可以很容易地计算出电路中的电流不再是以前的 0.75安（3伏除以4欧），而是比0.03安还小（3伏除以100欧以上）。几乎可以肯定，电流的大小不够点亮灯泡。

使用粗线是一个很好的解决方法，但价格太昂贵。 10号规格线（电器行的汽车电路耦合线价格为每35英尺$ 11.99，而且你需要双倍长度因为它只有单线）大约 0.1英寸粗，1000英尺为1欧姆，即1英里5欧姆。

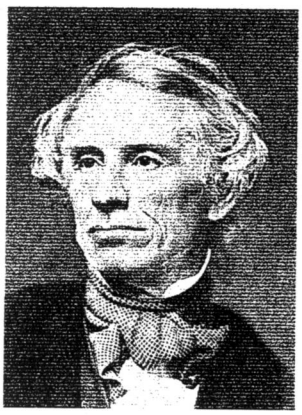
另一个解决办法是增加电压，使用大电阻灯泡。比如使用 120伏电压的100瓦家用照明灯泡的电阻为144欧姆。电线的电阻对于整个电路电流的影响将大大减小。

接下来的是150年前，人们在美洲和欧洲之间搭建第一个电报系统时所面临的问题。不管电线多粗，电压多高，电报线还是不能无限延长。根据计划，工作系统的极限为 200英里。这与纽约和加利福尼亚间的上千英里距离相差太多。

这个问题的答案 —— 不是为手电筒，而是为过去的嘀嗒电报 —— 虽说是一个简单易行的设备，但是通过它，整个计算机得以构造。

第6章　发报机与断电器

1791年，萨缪尔·摩尔斯生于马萨诸塞州的查尔斯顿镇，该镇是邦克山之战的地点，也是波士顿东北重镇。摩尔斯出生那年，美国宪法刚实施两年，乔治·华盛顿出任美国第一个任期的总统职务。Catherine大帝统治俄国。路易十六世和Marie Antoinette在两年后的法国大革命中被送上断头台。1791年，莫扎特完成了《魔笛》，他的最后一部作曲，次年于35岁时去世。



摩尔斯在耶鲁受过教育，又在伦敦学过艺术，他是位著名的肖像画家。他的作品《General Lafayette》(1825)珍藏于纽约市政大厅。1836年，他曾参与过竞选纽约市市长且获得了5.7%的选票。他也是早先的摄影术狂热爱好者。他从Louis Daguerre本人那儿学习了银版相片的制作，制造出了美国第一批用银版照相术制成的相片，1840年，他把这个手艺传授给了17岁的Mathew Brady。此人以及他的同事后来为美国内战、亚伯拉罕·林肯和摩尔斯本人留下了一些很有纪念价值的照片。

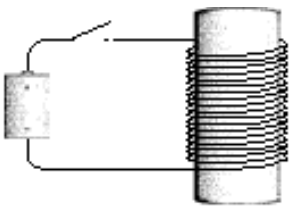
这些只是一个多职业生涯者的足迹。摩尔斯最著名的贡献在于他发明了电报和以他名字命名的编码。

世界范围内的即时通信我们已经很熟悉，但它是当今新技术发展的结果。19世纪早期，你可以即时通信和远距离通信，但不能同时达到两个要求。即时通信只能限制在你的声音能达到（没有扩音器可用）或是你的眼睛能看到（也许得用望远镜）的范围；远距离通信则要花时间用信件通过马车、火车或者轮船的方式来实现。

在早于摩尔斯发明的年代里，人们曾做过许多加速远距离通信的尝试。一种技术上简单的方法是雇佣一批人接力，站在山顶上用旗语信号通信。技术上稍微复杂一点儿的方法是使用巨大的带有可动手臂的装备，原理与旗语相同。

电报思想的正式成形是在19世纪早期。1832年在摩尔斯开始试验之前，已经有其他科学家在做一些试探。原理上讲，电报思想很简单：你在线的一端做某些事引起线的另一端发生了某些事。这正是上一章用远距离手电筒所做的事情。但摩尔斯不可能使用灯泡作为他的信号设备，因为实用性灯泡直到1879年才发明出来。摩尔斯使用的是电磁现象。

如果你取一只铁棒，用细导线将它绕几百圈，然后让电流通过导线，铁棒变成了磁铁，这时它就能吸引其他的铁和钢。（电磁铁上细线的电阻足够大以防止电磁铁形成短路。）移开电流，铁棒的磁性消失：

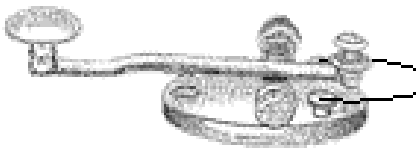


电磁铁是电报的基础。一端上开关的闭合引起另一端上的电磁铁产生一些动作。

摩尔斯最早的电报机比后来改进的要复杂得多。摩尔斯认为电报系统应该在纸上实际写点儿什么(这就像后来的电脑使用者描述的“生成一个硬拷贝”)。这当然不必是文字，因为文字太复杂，但某些字符应该记录下来，或曲线或点或划。注意，摩尔斯坚持要用纸记录下发报内容的这种想法，与Valentin Haüy要求盲人书籍应该使用突起的字母文字一样。

尽管摩尔斯早在1836年就告知专利局他已经成功地发明了电报，但直到1843年，他才说服议会为此设备的示范表演出资赞助。1844年5月24日是有历史意义的一天，Washington和马里兰州巴尔的摩之间的电报线成功地传送了圣经上的一句话“What hath God wrought！”。

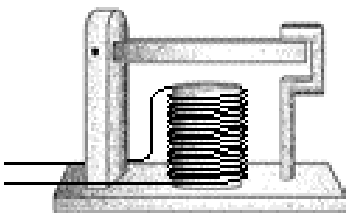
传统电报机发送消息的核心部分如下图所示：



尽管外观比较怪，但它只是一个为高速开合（闭）设计的开关，称为“按键/按钮”。长时间按键最舒适的方式是在手掌的拇指、食指和中指之间握住把手，然后敲击。短时间敲击形成摩尔斯电码的点，长时间敲击形成摩尔斯电码的划。

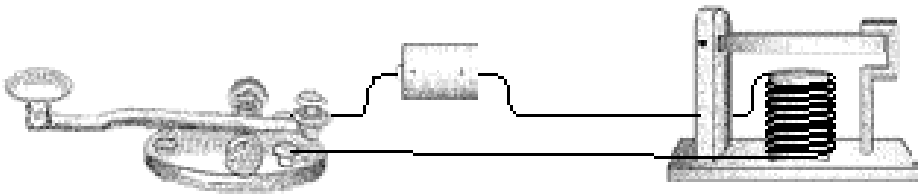
线的另一端是一个接收机，其基本结构是一个电磁铁吸拉一根金属拉杆。起初电磁铁控制的是一支笔，当由小装置控制的机械通过弯曲的弹簧缓慢地拖拉一卷纸时，相连的笔上下蹦弹将点划记录在纸上，懂得摩尔斯电码的人再将点划翻译成字母和文字。

当然，人是会偷懒的。电报机使用者很快发现只要简单地利用笔跳上跳下的声音他们就能翻译编码。笔的装置最终被撤消，代替的是传统电报机的发声装置，称为“发声器/音响器”，结构如下：



当电报机的键按下时，发生器的电磁铁将可动棒拖下发出“滴”的声音；当键放开时，棒弹回初始位置，发出“嗒”的声音。快速的“嘀嗒”为点，慢速的则为划。

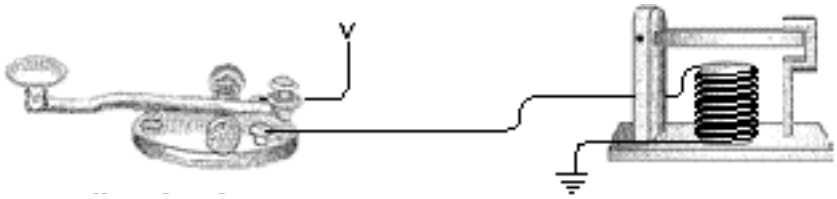
按键、发声装置，电池和一些导线可像上一章所述手电筒电报一样连接起来：



你的电报站　　你朋友的电报站

我们已经知道，两个电报站之间不需要两根线。如果大地作为另一半回路的话，一根线就足够了。

如上一章所做，我们用字母V代替接地的电池，因此最终的单向设置如下图所示：



你的电报站　　你朋友的电报站

双向通信只不过再需要一个按键和发生器。与上章所做相似。

电报的发明真正标志着现代通信的开始。人类首次能够在眼、耳的范围之外以快于马奔跑的速度通信。发明中使用的二元码是其精华所在，但在后来的电子和无线电通信中，包括电话、收音机和电视，二元码都没有用到，只到最近二元码才出现在计算机、CD盘、DVD盘、数字卫星电视广播和高清晰电视中。

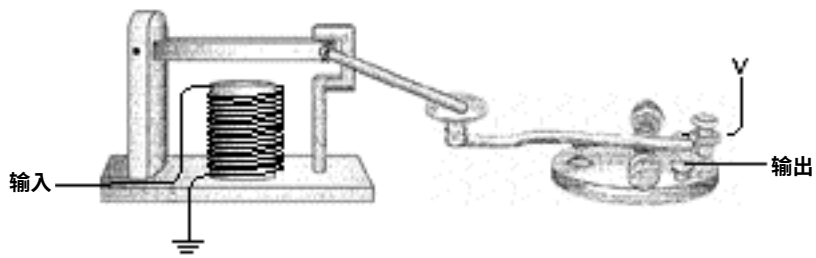
摩尔斯的电报机战胜了其他设计，部分原因是它对不好的电线状态的容忍度比较大。假如你在按键和发声装置之间接一根线，该电报机通常可以工作，但其他电报系统却不具备这样的容忍性。但正如上章所谈及的，最大的问题在于长距离导线的电阻。尽管一些电报线使用高达300伏的电压能在300英里的范围内工作，导线还是不能无限延伸。

一个明显的解决办法是使用转发（中继）系统，也称继电器系统。大约每200英里就让某位发报者通过发声装置接收消息再用按键发送出去。

现在想像一下你已被某电报公司雇佣为转发系统的工作人员。他们把你放在纽约和加利福尼亚之间某个地方的一间简陋得只有一张桌子和一把椅子的小屋里。一根导线从东边的窗户进来连到发声装置上。你的按键连在电池和从西边窗子出去的导线上。你的工作是接收来自于纽约的消息然后把它们发送到加利福尼亚。

起初，你是接收了整条消息后再转发它。你记录下发声器的嘀嗒，到消息接收结束，你再用你的按键将它们发送出去。最终你掌握了边听边发的技巧而不用把整条信息记录下来，这节约了转发时间。

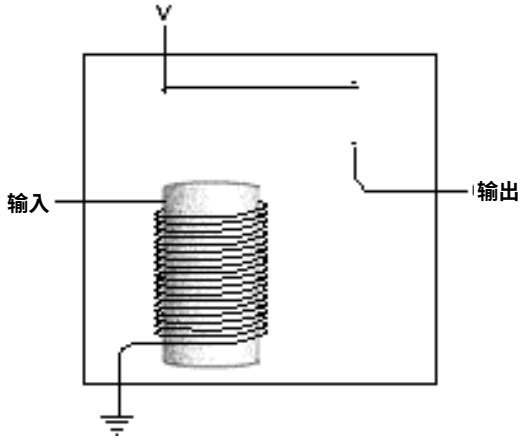
某天你在转发消息时，你注意到铁棒上下跳动又注意到了手指按动键的上下跳动。你看了看发声器又看了看键，然后你意识到棒的上下跳动与按键的上下跳动是一致的，于是你出去取回一根小木条，用这根木条和一些线把发声器和按键连接了起来：



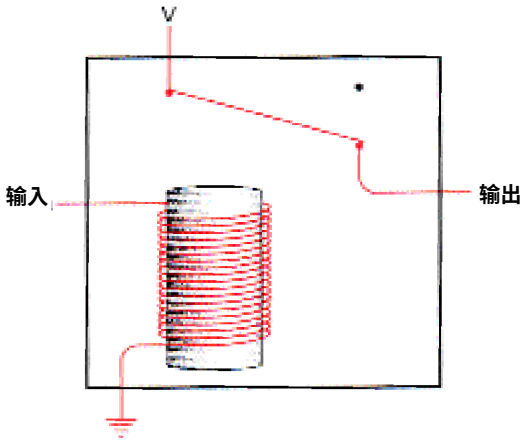
现在它可以自动工作了，你可以去喝下午茶也可以去钓鱼了。

这只是一个趣味情景的想像。但实际上，摩尔斯很早就理解这个装置的思想。我们已经发明的这个装置叫重发器或继电器。一个继电器就像一个发声装置，输入的电流形成电磁用以拖动金属杆，金属杆作为开关的一个部分连接到外接的导线上。这样，微弱的输入电流被扩大形成比较强的输出电流。

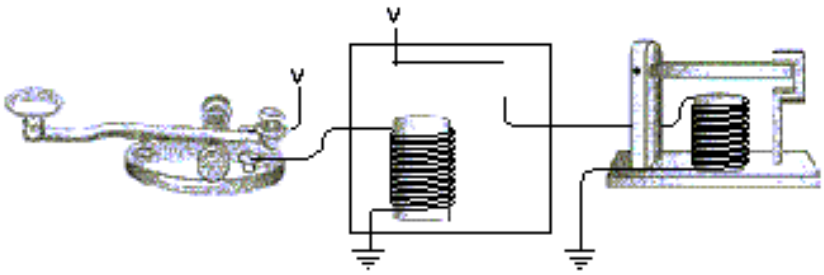
继电器的概要描述如下图所示：



输入电流激发电磁铁，电磁铁吸引一根有弹性的金属条作为开关从而输出电流：



因此电报按键、继电器和发声器大致连接如下：



你的电报站　　继电站　　你朋友的电报站

继电器是一种卓越的设备。它是一个开关，但并不是由人工而是借助于电流进行开关操作的。利用这种设备可以做出令人惊奇的事情。事实上，你可以用继电器装配出一台计算机中的大部分部件。

是的，继电器这种设备是一种很好的发明，足以与电报相提并论。后面还将会用到，且它会变得非常小巧、方便。但是，在能够使用它之前，得先学会数数。

第7章　十进制记数法

语言仅仅是一种编码的想法似乎很容易被人们接受，很多人在学生时代至少学过一种外语，因此，我们知道在英语中“ cat ”（猫）也可以被叫作gato、chat、Katze、KOIIIK或καπα。

然而，数字不那么容易随文化的不同而改变。不论那种语言，也不管怎样读那些数字，地球上我们能够遇到的几乎所有的人都用同样的方式来写数字：

1　2　3　4　5　6　7　8　9　10

数学，从某种意义上来说是不是可以称得上是一种世界语言呢？

毫无疑问，数字是我们平时能够接触到的最抽象的代码。当你看到数字“ 3 ”时并不需要立即将它和任何事情相联系。你可能将它设想为 3个苹果或者3个其他什么东西，但是当你从上下文中得知这个数字是指某个小孩的生日、电视频道、曲棍球比赛的得分或者是制作蛋糕的食谱中提供的需要面粉的杯数时，也能够像认为它代表 3个苹果时一样自然。因为数字一开始产生时就很抽象，所以让我们理解这些苹果：

并不一定要用符号“3 ”来表示就更困难了。本章的很大一部分以及下一章将来讲解这些苹果：

也可以用“ 11 ”的形式来表示。

先不讨论数字10与生俱来的特殊性。大多数人使用的数字系统是基于 10（有时候是5）的，这种情况并不奇怪。最初人们是用手指来数数的。要是人类进化成有 8个或12个手指，人类计数的方式就会有所不同。英语 Digit（数字）这个单词也可以指手指或脚趾，单词 five（五）和单词fist（拳头）有相同的词根，这种情况并不是巧合。

这样看来，人类选择使用以 10为基础的记数方法（或称为十进制记数法）完全是任意的，但我们赋予10的整数次幂重大的意义，并给它们命名：十个一年是一个十年；十个十年是一个世纪；十个世纪是一个千年；千个一千是百万；千个百万是十亿。下面是 10的各次幂：

$10^1 = 10$

$10^2 = 100$

$10^3 = 1000$（千）

$10^4 = 10\ 000$

$10^5 = 100\ 000$

$10^6 = 1\ 000\ 000$（百万）

$10^7 = 10\ 000\ 000$

$10^8 = 100\ 000\ 000$

$10^9 = 1\ 000\ 000\ 000$(十亿)

多数历史学家认为数字最初创造出来是用来数东西的，比如：人数、财产数、商品交易量等。举个例子来说，假定某个人有4只鸭子，他可能画4只鸭子作为记录：

后来，专门负责画鸭子这项工作的人想：“我为什么一定要画4只鸭子呢？为什么不能只画1只鸭子，然后用其他方法（管它用什么方法，哪怕用一条竖线来代表一只鸭子）来表示有4只呢？”

但若某人有27只鸭子，用画竖线来表示鸭子只数的方法就显得很荒谬了：

于是，有人想到得有一种好的办法才行，数字系统就这样诞生了。

在早期的数字系统中，只有罗马数字系统沿用至今。钟表的表盘上常常使用罗马数字，此外它还用来在纪念碑或雕像上标注日期、标注书的页码，或作为提纲条目的标记。最令人惊奇的是罗马数字常用在电影中做版本说明。（只要你有足够快的速度将字幕结尾处出现的MCMLIII译码，通常情况下就可以回答“这部影片是什么时候拍的”这个问题。）

27只鸭子可以用罗马数字这样表示：

这里用到的概念非常简单：X代表10条竖线，V代表5条竖线。

现在仍在使用的罗马数字有：

I V X L C D M

字母I代表一个一，这可能来自于一条竖线或者伸出的一个手指。字母 V很可能是一只手的符号，代表五；两个字母V组成字母X，代表十；字母L代表五十；字母C来自于拉丁文中表示一百的单词 —— centum；字母D代表五百；最后，字母M来自拉丁文中的单词 —— mille，代表一千。

也许你不一定同意，很长一段时间以来，罗马数字被认为用来做加减运算非常容易，这也是罗马数字能够在欧洲被长期用于记帐的原因。事实上，当对两个罗马数字进行相加运算时，只需将这两个罗马数字的所有符号合并然后用下面的方法将其简化：五个 I是一个V，两个V是一个X，五个X是一个L，等等。

但使用罗马数字做乘除法是很难的。很多其他早期的数字系统（比如古希腊数字系统）和罗马数字系统相似，它们在做复杂运算时存在一定的不足。尽管如此，古希腊人所发明的

非凡的几何学至今仍是中学的一门课程，古希腊人不是以代数享誉世界的。

我们现在使用的数字系统通常称为阿拉伯数字系统，或称为印度—阿拉伯数字系统。它起源于印度，但由阿拉伯数学家传入欧洲。一位著名的波斯数学家 —— Muhammed ibn-Musa al-Khwarizmi（由它的名字得到单词 algorithm（算法））在大约公元 825年写了一本代数书，书中用的就是印度的数字系统（阿拉伯数字）来计数。产生于公元 1120年的拉丁文译本对整个欧洲用现在的阿拉伯数字代替当时使用的罗马数字的过渡过程产生了很大的影响。

印度—阿拉伯数字系统与先前的数字系统相比在以下三个方面不同：

- 印度-阿拉伯数字系统是和位置相关的，也就是说，一个数字依据位置的不同代表不同的数量。数字的位置和数字的大小一样，都是很重要的。（但实际上，数字的位置更重要。）100和1 000 000中都只有一个1，但我们知道一百万比一百要大得多。
- 几乎所有早期的数字系统都有一个阿拉伯数字所没有的东西，那就是用来表示数字 10的一个专门的符号。现在使用的数字系统中是没有代表 10的专门符号的。
- 另一方面，几乎所有早期的数字系统都缺少一个阿拉伯数字中有的，而且事实证明是比代表数字10的符号重要得多的符号，那就是零。

是的，就是零。这个小小的零毫无疑问是数字和数学历史上最重要的发明之一。它支持位置表示法，因为它可以将 205与250区别开来。数字零也使得与位置无关的数字系统中非常复杂的运算变得简单，尤其是乘除法。

印度—阿拉伯数字的整体结构是以读它们的方式展现的。拿 4825作为例子，我们把它读作“四千八百二拾五”，意思是：

四个一千
八个一百
两个十
一个五

或者，可以将它的组成写成这样：

$$4825=4000+800+20+5$$

或者，可以将它进一步分解，写成这样：

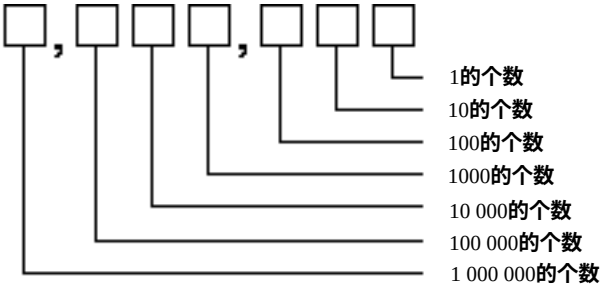
$$\begin{aligned}4825=&4\times1000+\\&8\times100+\\&2\times10+\\&5\times1\end{aligned}$$

另外，也可以使用10的整数次幂的形式，重新写成：

$$\begin{aligned}4825=&4\times10^3+\\&8\times10^2+\\&2\times10^1+\\&5\times10^0\end{aligned}$$

记住，任何数的0次幂都等于1。

多位数中的每位都有特定的意义，如下图所示。这 7个方格可以表示从0～9 999 9999的任何一个数字：



每一个位置（位）与10的一个整数次幂相对应。不需要一个专门的符号来表示数字10，因为可以将1放在不同的位置，用0作为占位符。

分(小)数可以同样的形式作为数字放在十进制数的小数点的右边，这一点非常好。数字42 705.684是：

$$4\times 10\,000+$$
$$2\times 1000+$$
$$7\times 100+$$
$$0\times 10+$$
$$5\times 1+$$
$$6\div 10+$$
$$8\div 100+$$
$$4\div 1000$$

该数也可以写成不带除法的形式，如下：

$$4\times 10\,000+$$
$$2\times 1000+$$
$$7\times 100+$$
$$0\times 10+$$
$$5\times 1+$$
$$6\times 0.1+$$
$$8\times 0.01+$$
$$4\times 0.001$$

或写成10的整数次幂的形式：

$$4\times 10^{4}+$$
$$2\times 10^{3}+$$
$$7\times 10^{2}+$$
$$0\times 10^{1}+$$
$$5\times 10^{0}+$$
$$6\times 10^{-1}+$$
$$8\times 10^{-2}+$$
$$4\times 10^{-3}$$

注意10的指数是怎样变到零再变成负数的。

我们知道，3加上4等于7。同样，30加上40等于70，300加上400等于700，3000加上4000等于7000。这正是阿拉伯数字系统的“魅力”所在，无论你进行多长的十进制的加法，只要根据一种方法将问题分成几步即可。每一步最多只是将两个一位数字相加，这也是很久以前有人强迫你记加法表的原因：

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

从最上边的一行和最左边的一列找到要相加的两个数字，在行与列的交叉点上找到它们相加的结果。例如，4加上6等于10。

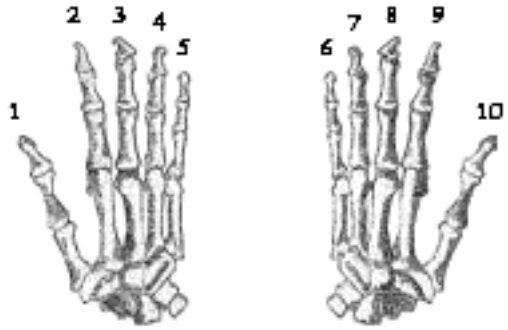
同样，做两个十进制数相乘的运算时，方法可能稍稍复杂一点儿，但仍然只需将问题分成几步，这样就不会比做加法和一位数的乘法更复杂了。你在小学时可能也必须记住下面的乘法表：

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

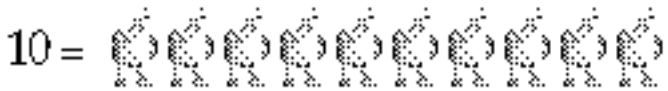
与位置相关的记数系统的优点不在于它多么好用，而在于当它用在不是十进制的系统中时，也一样的好用。我们现在用的数字系统不一定适合所有的人。十进制数字系统的一个很大问题就在于它和卡通人物没有任何关系。大多数的卡通人物每只手上只有 4个手指，因此它们喜欢基于8的数字系统（八进制）。有趣的是，我们所知的大部分关于十进制数的知识同样可以用于卡通朋友所喜爱的八进制数字系统中。

第8章　其他进位制记数法

10对我们来说是一个非常重要的数字。 10是我们大多数人拥有的手指或脚趾的数目，我们当然希望所有人的手指脚趾都是 10个。因为我们的手非常适合数数，因而我们人类已经适应了以10为基础的数字系统：

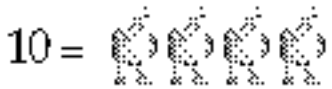


前面数章已经提到过，通常使用的数字系统称为以 10为基础的数字系统或十进制。这个数字系统对我们来说非常自然，因而我们很难想像出还有其他的数字系统。事实上，当我们看到数字10的时候，不由自主地就会认为这个数是指下面这么多只鸭子：

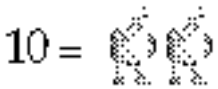


但是，数字10是指这么多只鸭子的唯一理由是因为这么多只鸭子与我们的手指数目相同。如果人类不是有那么多只手指，我们数数的方式就会有所不同，数字 10就可能代表别的东西了。同样是数字10，可以指这么多只鸭子：

或这么多只鸭子：



甚至可以是这么多只鸭子：



当我们明白了10可以指只有两只鸭子的时候，也就可以解释开关、电线、灯泡、继电器（或干脆就叫计算机）是怎样表示数字的了。

如果人类像卡通人物那样，每只手上只有 4个手指会怎样呢？我们可能永远都不会想到要发明一种以10为基础的数字系统的问题，取而代之的是我们可能会认为数字系统基于 8是正常、自然、合理、必然的，是毫无疑问的，是非常合适的。这时，就不能称之为十进制了，得将它称作为以8为基础的数字系统或八进制。

如果数字系统是以8为基础组织起来的，就不需要这样的一个符号：

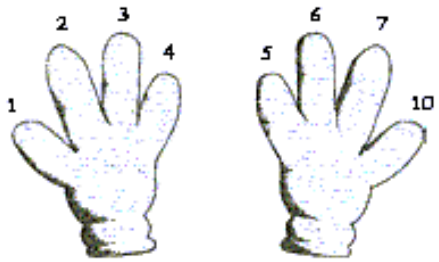
9

把这个符号拿给任何一个卡通人物看，都会有同样的反应：“ 那是什么？它是干什么用的？ ”如果再仔细想一会儿的话，你会发现连这样的一个字符也不需要：

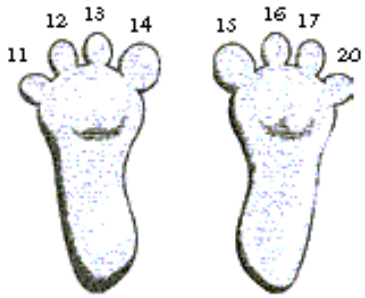
8

在十进制数字系统中，没有专门用来表示 10的符号，所在在八进制数字系统中，也没有专门用来表示10的符号。

在十进制数字系统中数数的方式是 0、1、2、3、4、5、6、7、8、9，然后是10。在八进制数字系统中数数的方式是 0、1、2、3、4、5、6、7，然后是什么呢？我们已经没有符号可用了，唯一的一个有意义的可用符号是 10，的确是那样。在八进制数中， 7之后紧接着的数字是10，但是10并不是指人类的手指那么多的数目。在八进制数中， 10指的是卡通人物手指的数目：



继续数脚趾头：



使用非十进制的数字系统时，将数字“ 10 ”读作“ 么零 ”可以避免一些混淆。同样，“ 13 ”可以读作“ 么三 ”，“ 20 ”可以读作“ 二零 ”。要想真正避免混淆，可以将“ 20 ”读作“ 八进制二零 ”或“ 基于8的数二零 ”。

即使没有手指和脚趾帮忙，我们仍能够将八进制数继续数下去。除了要跳过那些含有 8或9的数字以外，它基本上和数十进制的数是一样的。当然，相同的数字代表的数量是不同的：

0、1、2、3、4、5、6、7、10、11、12、13、14、15、16、17、
20、21、22、23、24、25、26、27、30、31、32、33、34、35、36、37、40、
41、42、43、44、45、46、47、50、51、52、53、54、55、56、57、60、61、62、
63、64、65、66、67、70、71、72、73、74、75、76、77、100...

最后一个数字读作“么零零”，是卡通人物拥有的手指数自乘的结果（即平方）。

在写十进制或八进制数时，为避免混淆，可以借助使用特定的标记以区别表示数字系统。下面用标记“TEN”表示十进制数，标记“EIGHT”表示八进制数。

这样，白雪公主遇到的小矮人的数目是 7_{TEN} 或 7_{EIGHT}

卡通人手的手指数是 8_{TEN} 或 10_{EIGHT}

贝多芬写的交响乐的首数是 9_{TEN} 或 11_{EIGHT}

人的手指的数目是 10_{TEN} 或 12_{EIGHT}

一年中的月份数是 12_{TEN} 或 14_{EIGHT}

两个星期所包含的天数是 14_{TEN} 或 16_{EIGHT}

“情人”的生日庆祝会是 16_{TEN} 或 20_{EIGHT}

一天中所包含的小时数是 24_{TEN} 或 30_{EIGHT}

拉丁字母表中的字符数是 26_{TEH} 或 32_{EIGHT}

与一夸脱液体相当的盎司数为 32_{TEN} 或 40_{EIGHT}

一副牌中含有的牌数是 52_{TEN} 或 64_{EIGHT}

国际象棋棋盘的方格数是 64_{TEN} 或 100_{EIGHT}

Sunset Strip 最著名的17牌号是 77_{TEN} or 115_{EIGHT}

美式足球场的面积是 100_{TEN} 或 144_{EIGHT}

参加温布尔登网球公开赛女单初赛的人数是 128_{TEN} 或 200_{EIGHT}

古埃及孟斐斯城市面积的平方英里数是 256_{TEN} or 400_{EIGHT}

注意，在上面一系列的八进制数中，有一些好整数，像 100_{EIGHT}、200_{EIGHT}、400_{EIGHT}。好整数通常是指结尾有一些零的数。在结尾处有两个零的十进制数意味着它是 100_{TEN} 即 10_{TEN} 乘以 10_{TEN}；在八进制数中，结尾处有两个零表示它是 100_{EIGHT} 即 10_{EIGHT} 乘以 10_{EIGHT}（或 8_{TEN} 乘以 8_{TEN}，等于 64_{TEN}）。

你可能已经注意到了，好的八进制整数 100_{EIGHT}、200_{EIGHT} 和 400_{EIGHT} 与十进制数 64_{TEN}、128_{TEN}、256_{TEN} 相等，它们都是2的整数次幂。例如，400_{EIGHT} 等于 4_{EIGHT} 乘以 10_{EIGHT} 乘以 10_{EIGHT}，所有这些数都是2的整数次幂。任何时候，将2的整数次幂和另一个2的整数次幂相乘，得到的仍是2的整数次幂。

下表给出了一些2的整数次幂的十进制及其对应的八进制的表示形式：

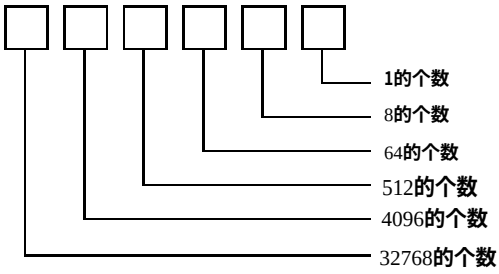
2的整数次幂	十进制数	八进制数
2^0	1	1
2^1	2	2
2^2	4	4
2^3	8	10
2^4	16	20
2^5	32	40
2^6	64	100

(续)

2^7	128	200
2^8	256	400
2^9	512	1000
2^{10}	1024	2000
2^{11}	2048	4000
2^{12}	4096	10000

最右边一列的好整数给我们一个暗示：十进制以外的数字系统可能对使用二元码有所帮助。

八进制数字系统和十进制数字系统在结构上没有什么差别，只是在细节上有一些差异。例如，八进制数的每一个位置代表的值是该位数字乘以 8的整数次幂的结果：



这样，八进制数3725_{EIGHT}可以拆分成这样：

$$3725_{EIGHT} = 3000_{EIGHT} + 700_{EIGHT} + 20_{EIGHT} + 5_{EIGHT}$$

还可以写成另外几种不同的形式。下面就是其中的一种，采用十进制形式的 8的整数次幂：

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 512_{TEN} + \\ & 7 \times 64_{TEN} + \\ & 2 \times 8_{TEN} + \\ & 5 \times 1 \end{aligned}$$

采用八进制形式的8的整数次幂的情况：

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 1000_{EIGHT} + \\ & 7 \times 100_{EIGHT} + \\ & 2 \times 10_{EIGHT} + \\ & 5 \times 1 \end{aligned}$$

还有另外的一种拆分形式：

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 8^3 + \\ & 7 \times 8^2 + \\ & 2 \times 8^1 + \\ & 5 \times 8^0 \end{aligned}$$

如果算出其十进制的结果，会得到2005_{TEN}。这就是将八进制数转换成十进制数的方法。

可以采用与做十进制加法和乘法相同的办法来做八进制数的加法和乘法。唯一真正的区别在于要采用不同的表格来对各个数字进行乘法或加法运算。下面是八进制数的加法表：

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

例如，$5_{EIGHT} + 7_{EIGHT} = 14_{EIGHT}$。可以采用与做十进制加法相同的方法将两个稍长一点儿的八进制数相加：

$$\begin{array}{r} 135 \\ +\ 643 \\ \hline 1000 \end{array}$$

先从最右边的一列做起，5加上3等于10，该位写下0，向前进1；1加3加4等于10,该位写下0，向前进1；1加1加6等于10。

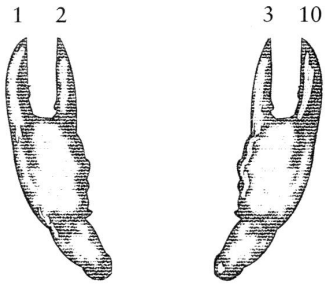
同样，在八进制中，2乘以2仍然等于4。但是3乘以3却不等于9，那是多少呢？3乘以3等于11_{EIGHT}，此数与9_{TEN}所代表的数量相等。下图是完整的八进制数的乘法表：

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

这里，4×6等于30_{EIGHT}，也即表明30_{EIGHT}和4×6的十进制结果24_{TEN}是等值的。

八进制数字系统与十进制数字系统一样，都是有效的，但八进制数字系统在理解上更深了一层。既然我们已为卡通人物开发出了一套数字系统，就再给龙虾开发一套适合它们用的数字系统吧。龙虾根本没有手指，但它两只前爪的末端都有螯。适合于龙虾的数字系统是四

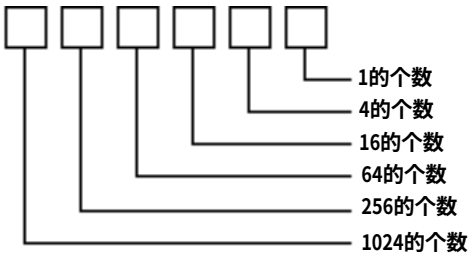
进制数字系统或称为基于4的数字系统：



四进制数可以这样来数：

0、1、2、3、10、11、12、13、20、21、22、23、30、31、32、33、100、101、102、103、110，等等。

这里不打算在四进制数上花太多的时间，因为还有更重要的事情要做。但我们还是要看一下四进制中的每一位是怎样和4的某个整数次幂相对应的：



四进制数31232可以写成：

$$31232_{FOUR} = 3\times 256_{TEN} + 1\times 64_{TEN} + 2\times 16_{TEN} + 3\times 4_{TEN} + 2\times 1_{TEN}$$

也可以写成：

$$31232_{FOUR} = 3\times 10000_{FOUR} + 1\times 1000_{FOUR} + 2\times 100_{FOUR} + 3\times 10_{FOUR} + 2\times 1_{FOUR}$$

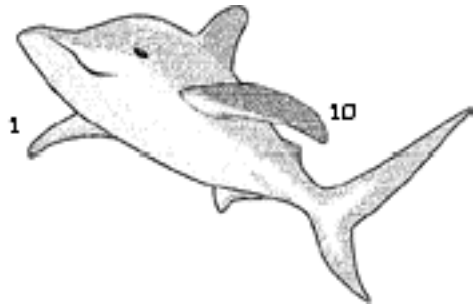
还可以写成：

$$31232_{FOUR} = 3\times 4^4 + 1\times 4^3 + 2\times 4^2 + 3\times 4^1 + 2\times 4^0$$

如果以十进制数的形式计算其结果，就会发现 31232_{FOUR} 等于 878_{TEN}。

现在，我们要做一个跳跃并且是最远的一跳。假定我们是海豚，并且必须用两鳍来数数。则这个数字系统就是基于2的数字系统或二进制的。这样似乎只需要两个数字，即0和1。

现在，0和1已是你要处理的全部问题，需要练习一下才能习惯使用二进制数。二进制数最大的问题是数字用完得很快。例如，下图是海豚怎样用它的鳍数数的例子：



是的，在二进制中，1后面的数字是10。这是令人惊讶的，但也并不奇怪。无论使用哪种数字系统，当单个位的数字用完时，第一个两位数字都是10。在二进制系统中，可以这样来数数：

0、1、10、11、100、101、110、111、1000、1001、1010、
1011、1100、1101、1110、1111、10000、10001、……

这些数看起来好像很大，实际上并不是这样。更准确地说二进制数长度增长的速度要快过二进制数增大的速度：

每个人的头的个数为 1_{TEN} 或 1_{TWO}

海豚身上的鳍的个数为 2_{TEN} 或 10_{TWO}

一个大汤匙中包括的小茶匙的数目为 3_{TEN} 或 11_{TWO}

正方形的边数为 4_{TEN} 或 100_{TWO}

每个人一只手的手指数为 5_{TEN} 或 101_{TWO}

一种昆虫的腿数为 6_{TEN} 或 110_{TWO}

一星期的天数为 7_{TEN} 或 111_{TWO}

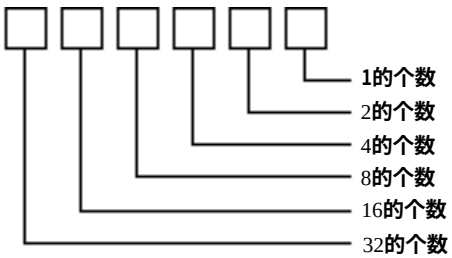
八重奏中音乐家的个数为 8_{TEN} 或 1000_{TWO}

太阳系中的行星（包括冥王星在内）总数为 9_{TEN} 或 1001_{TWO}

牛仔帽重量以加仑计算为 10_{TEN} 或 1010_{TWO}

等等。

在多位二进制数中，数字的位置和2的整数次幂的对应关系为：



因此，任何时候由一个1后跟几个零构成的二进制数一定是2的整数次幂。2的幂与二进制

数中零的个数相等。下面是扩充的2的各次幂的表，可用来说明这条规则：

2的幂	十进制数	八进制数	四进制数	二进制数
2^0	1	1	1	1
2^1	2	2	2	10
2^2	4	4	10	100
2^3	8	10	20	1000
2^4	16	20	100	10000
2^5	32	40	200	100000
2^6	64	100	1000	1000000
2^7	128	200	2000	10000000
2^8	256	400	10000	100000000
2^9	512	1000	20000	1000000000
2^{10}	1024	2000	100000	10000000000
2^{11}	2048	4000	200000	100000000000
2^{12}	4096	10000	1000000	1000000000000

假定有一个二进制数101101011010，它可以写成：

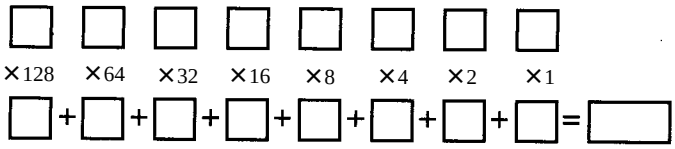
$$\begin{aligned}101101011010_{TWO} = {} & 1\times 2048_{TEN}+\\ & 0\times 1024_{TEN}+\\ & 1\times 512_{TEN}+\\ & 1\times 256_{TEN}+\\ & 0\times 128_{TEN}+\\ & 1\times 64_{TEN}+\\ & 0\times 32_{TEN}+\\ & 1\times 16_{TEN}+\\ & 1\times 8_{TEN}+\\ & 0\times 4_{TEN}+\\ & 1\times 2_{TEN}+\\ & 0\times 1_{TEN}\end{aligned}$$

也可以这样写：

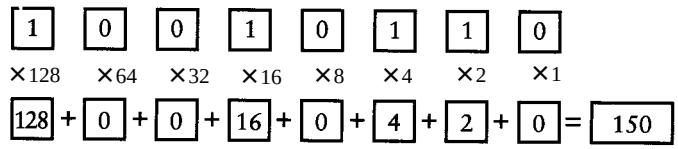
$$\begin{aligned}101101011010_{TWO} = {} & 1\times 2^{11}+\\ & 0\times 2^{10}+\\ & 1\times 2^{9}+\\ & 1\times 2^{8}+\\ & 0\times 2^{7}+\\ & 1\times 2^{6}+\\ & 0\times 2^{5}+\\ & 1\times 2^{4}+\\ & 1\times 2^{3}+\\ & 0\times 2^{2}+\\ & 1\times 2^{1}+\\ & 0\times 2^{0}\end{aligned}$$

如果将各个部分以十进制数的形式相加，得到 $2048+512+256+64+16+8+2=2906_{TEN}$。

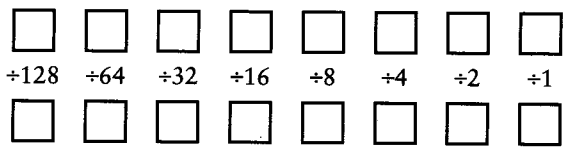
将二进制数转换成十进制数非常简单，你可能更喜欢借助已准备好的模板进行转换：



这个模板允许你转换最大长度为 8的二进制数，但它扩充起来非常容易。使用时，将 8个二进制数字放到上部的 8个小盒子中，一个盒子放一个数字。做 8个乘法运算，将结果分别放到底部的8个小盒子中。将8个盒子中的数字相加就得到最终结果。下面是将 10010110转化成十进制数的例子：

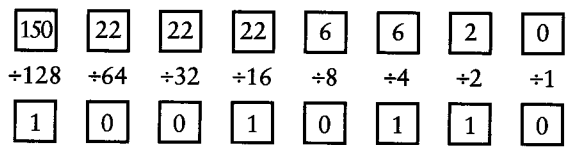


将十进制数转换成二进制数就没那么直接了。但这里也有一个帮助你将 0～225范围内的十进制数转换成二进制数的模板：



实际转化过程要表面上看的麻烦得多，所以一定要仔细按照下面的指导来做。将整个十进制数（应小于等于 225）放在左上角的方格中。用除数（ 128）去除那个数（被除数），如下图所示。将商写在正下方的盒子中（即左下角的盒子中），余数写在右边的盒子中（即上面一行左数第二个盒子中）。用第一个余数再除以下一个算子 64。依照模板的顺序用同样的方法继续做下去。

记住，每次求得的商只能是 0或者1。如果被除数小于除数，商为 0，余数和被除数相等；如果被除数大于除数，商为 1，余数为被除数与除数之差。下面是将 150转换成二进制数的过程：



如果要做两个二进制数的加法或乘法，也许直接采用二进制来做比转化成十进制再做还要简单。这将是你真正喜欢二进制数的地方。如果只需记住下面的二进制加法表就可以做加法运算，也就不难想象掌握加法运算该有多快：

+	0	1
0	0	1
1	1	10

用二进制加法表将两个二进制数相加：

```
  1 1 0 0 1 0 1
+ 0 1 1 0 1 1 0
---------------
1 0 0 1 1 0 1 1
```

从最右边的一列开始做起：1加上0等于1；右数第2列：0加上1等于1；第3列：1加上1等于0，进位为1；第4列：1（进位值）加上0再加上0等于1；第5列：0加上1等于1；第6列：1加1等于0，进位为1；第7列：1（进位值）加上1再加上0等于10。

乘法表比加法表更简单，因为该表可以由两个基本的乘法规则推导出来：零乘以任何数都等于0，1与任何数相乘仍是那个数本身：

×	0	1
0	0	0
1	0	1

下面是13_{TEN}与11_{TEN}以二进制数的形式做乘法的过程：

```
    1 1 0 1
  × 1 0 1 1
-------------
    1 1 0 1
  1 1 0 1
0 0 0 0
1 1 0 1
-------------
1 0 0 0 1 1 1 1
```

最后结果是143_{TEN}。

人们在使用二进制数的时候通常将它们写成带有前导零的形式（即第一个1的左边有零）。例如，0011而不写成11。这些零不会改变数字的值，只是起到一些装饰作用。例如，下面是二进制的前16个数以及和它们等值的十进制数：

二进制数	十进制数
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

让我们再仔细看看这些二进制数字。考虑一下这4个垂直列中每一列的0和1，注意它们在一列中自上而下是以怎样的规律变化的：

- 最右边一列一直在0和1之间相互替换。
- 右数第2列在两个0和两个1之间相互替换。
- 右数第3列在四个0和四个1之间相互替换。
- 右数第4列在八个0和八个1之间相互替换。

这是很有规律的，难道不是吗？事实上，只要再重复这 16个数字并且在每个数字的前面放一个1就可以很容易地写出后面的16个数字：

二进制数	十进制数
10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31

下面是看待这些数字的另一种方式：在数二进制数的时候，最右边的数字（也称最低位数字）是在0和1之间变化的。当它每次从 1变到0时，右数第二位数字（也称次低位数字）也要发生变化，或者从 0变到1，或者从1变到0。每次只要有一个二进制数位的值由 1变到0，紧挨着的高位数字也会发生变化，要么从 0变到1，要么从1变到0。

我们在写十进制中比较大的数字时，通常每三个数字之间留一点儿空隙，这样，我们一看就知道这个数的大概数值。例如，当你看到数字 12000000时，你可能不得不去数其中0的个数，但如果看到的是12 000 000，则马上就能知道是一亿两千万。

二进制数的位长度增加得特别快。例如，一亿两千万的二进制表示为：1011011100011011000000000。为了让它更易读，通常是每四个数字之间用连字符或空格来分开。例如；1011-0111-0001-1011-0000-0000或1011011100011011000000000。本书的后面会讲到更简单的二进制数的表示方法。

通过将数字系统减少至只有 0和1两个数字的二进制数字系统，我们已经在能够接受的范围内做了深入的讨论。不可能找到比二进制数字系统更简单的数字系统了。二进制数字系统架起了算术与电之间的桥梁。前面各章中，我们所看到的开关、电线、灯泡、继电器等物体都可以表示二进制数0和1：

电线可以表示二进制数字。有电流流过电线代表二进制数字 1；如果没有，则代表二进制数字0。

开关可以表示二进制数字。如果开关闭合，代表二进制数字 1；如果开关断开，代表二进制数字0。

灯泡可以表示二进制数字。如果灯泡亮着，代表二进制数字 1；如果没亮，代表二进制数

字0。

电报继电器可以表示二进制数字。继电器闭合，代表二进制数字 1；继电器断开，代表二进制数字0。

二进制数与计算机密切相关！

大约在1948年，美国数学家John Wilder Tukey （生于1915年）提前认识到二进制数将在未来几年中随着计算机的流行而发挥更大的作用。他决定创造一个新的、更短的词来代替使用起来很不灵活的五音节词 —— binary digit。他曾经考虑用bigit或binit，但最后还是选用了短小、简单、精巧且非常可爱的单词bit(比特)来代替binary digit这个词。

第9章　二进制数

1973年，当安东尼·奥兰多在他写的一首歌中要求他挚爱的人“系一条黄色的绸带在橡树上”时，他并没有要求他的爱人进行繁琐的解释或冗长的讨论，只要求她给他一个简单的结果。他不去关心其中的因果，即使歌中复杂的感情和动情的历史在现实生活中重演，所有的人真正想知道的仅仅是一个简单的是或不是。他希望在树上系一条黄色的绸带来表示：“是的，即使你犯了很大的错，并且被判了入狱三年，我仍希望你回来和我一起共渡时光。”他希望用树上没有黄色的绸带来表示：“你连停在这里都别想。”

这是两个界线分明、相互排斥的答案。奥兰多没有这样唱：“如果你想再考虑一下的话，就系半条黄色的绸带”或者“如果你不爱我但仍希望我们是朋友，就系一条蓝色的绸带吧”。相反，他让答案非常的简单。

和黄色绸带的有无具有同样效果的另外几个例子（但可能无法用在诗里）是可以选择一种交通标记放在院外，可能是“请进”或“此路不通”。

或者在门上挂一个牌子，上写“关”或“开”。

或者用从窗口能够看到的一盏灯的亮灭来表示。

如果你只需说“是”或“不是”的话，可以有很多种方式来表达。你不必用一个句子来表达是或不是，也不需要一个单词，甚至连一个字母都不要。你只要用一个比特，即只要一个0或1即可。

正如我们在前面的章节中所了解到的，通常用来计数的十进制数事实上并没有什么与众不同的地方。非常清楚，我们的数字系统之所以是基于 10的（十进制数）是因为我们有 10个手指头。我们同样有理由使用八进制数字系统（如果我们是卡通人物）或四进制数字系统（如果我们是龙虾），甚至是二进制数字系统（如果我们是海豚）。

但是，二进制数字系统有一点儿特别：它可能是最简单的数字系统。二进制数字系统中只有两种二进制数字——0和1。要是我们想寻求更简单的数字系统，只好把数字 1去掉，这样，就只剩下0一个数字了。只有一个数字0的数字系统是什么都做不成的。

“bit(比特)”这个词被创造出来代表“binary digit”，它的确是新造的和计算机相关的最可爱的词之一。当然，“bit”有其通常的意义：“一小部分，程度很低或数量很少”。这个意义用来表示比特是非常精确的，因为1比特——一个二进制数字位——确实是一个非常小的量。

有时候当一个新词诞生时，它还包含了一种新的意思。 bit这个词也是这样。 1比特的意思超过了被海豚用来数数的二进制数字位所包含的意义。在计算机时代，比特已经被看作是组成信息块的基本单位。

当然，上述说法不一定完全正确，比特并不是传送信息的唯一的方式。字母、单词、摩尔斯码、布莱叶盲文，十进制数字都可以用来传递信息。比特传递的信息量很小。 1比特只具备最少的信息量，更复杂的信息需要多位比特来传递。（我们说比特传递的信息量小，并不是说它传送的信息不重要。事实上黄绸带对于与它相关的两个人来说是一个非常重要的信息。）

“听，孩子们，你们很快就能听到 Paul Revere 午夜的马蹄声。”亨利·朗费罗写道。尽管

他在描述Paul Revere是怎样通知美国人英国殖民者入侵的消息时不一定与史实完全一致，但他的确提供了一个利用比特传递信息的令人茅塞顿开的例子：

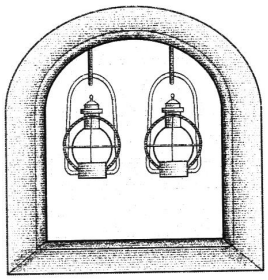
He said to his friend "If the British march
By land or sea from the town to-night,
Hang a lantern aloft in the belfry arch
Of the North Church tower as a special light,—
One, if by land, and two, if by sea..."

(他告诉他的朋友："如果英军今晚入侵，
你就在北教堂的钟楼拱门上悬挂点亮的提灯
作为信号。一盏提灯代表英军由陆路入侵，
两盏提灯代表英军由海路入侵。……)

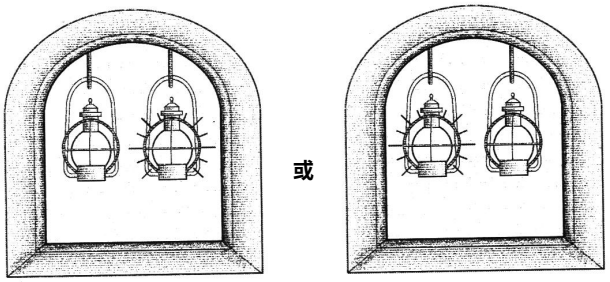
也就是说，Paul Revere 的朋友有两盏灯。如果英军由陆路入侵，他就挂一盏灯在教堂的钟楼上；如果英军由海路入侵，他就挂两盏灯在教堂的钟楼上。

然而，朗费罗并没有将所有的可能都涉及到。他留下第三种情况没有说，那就是英军根本就没有入侵的情况。朗费罗已经暗示第三种可能的信息可以由不挂提灯的方式来传递。

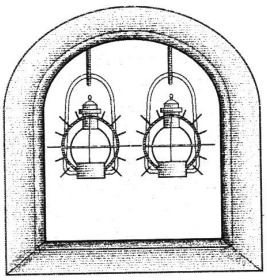
让我们假设那两盏灯是永久固定在教堂钟楼上的。在正常情况下，它们都不亮：



这就是指英军还没有入侵。如果一盏提灯亮：



表示英军正由陆路入侵。如果两盏提灯都亮：



表示英军正由海路入侵。

每一盏提灯都代表一个比特。亮着的灯表示比持值为 1，未亮的灯表示比特值为 0。前面奥兰多已经说明了传送只有两种可能性的信息只需要一个比特。如果 Paul Revere只需被告知英军正在入侵（不管是从何处入侵）的消息，一盏提灯就足够了。点亮提灯代表英军入侵，未点亮提灯代表又是一个和平之夜。

传递三种可能性的消息还需要再有一盏提灯。一旦再有一盏提灯，两个比特就可以通知有四种可能的信息：

00=英军今晚不会入侵
01=英军正由陆路入侵
10=英军正由陆路入侵
11=英军正由海路入侵

Paul Revere将三种可能性用两盏提灯来传送的做法事实上是相当富有经验的。用通信理论的术语说，他采用了冗余的办法来降低噪声的影响。通信理论中的噪声是指影响通信效果的任何事物。电话线路中的静电流显然是影响电话通信的一种噪声。然而，即使是在有噪声的情况下，电话通信仍能够成功，因为口语中存在大量的冗余。你同样可以听懂对方的话而无需将每个音节、每个字都听得很清楚。

在上述例子中，噪声是指晚上光线黯淡以及 Paul Revere距钟楼有一定的距离，它们都阻碍了Paul Revere声将钟楼上的两盏灯区分清楚。下面是朗费罗的诗中很重要的一段：

And lo! As he looks, on the belfry's height
A glimmer, and then a gleam of light!
He springs to the saddle, the bridle he turns,
But lingers and gazes, till full on his sight
A second lamp in the belfry burns!

（哦！他站在与钟楼等高的位置观察，
一丝微光，然后，有一盏灯亮了！
他跳上马鞍，调转马头，
徘徊，凝视，直到看清所有的灯
另一盏灯也亮了！）

那当然不是说Paul Revere正在辨清到底是哪盏灯先亮的问题。

这里最本质的概念是信息可能代表两种或多种可能性的一种。例如，当你和别人谈话时，说的每个字都是字典中所有字中的一个。如果给字典中所有的字从 1开始编号，我们就可能精确地使用数字进行交谈，而不使用单词。（当然，对话的两个人都需要一本已经给每个字编过号的字典以及足够的耐心。）

换句话说，任何可以转换成两种或多种可能的信息都可以用比特来表示。不用说，人类使用的很多信息都无法用离散的可能性来表示，但这些信息对我们人类的生存又是至关重要的。这就是人类无法和计算机建立起浪漫关系的原因所在（无论怎样，都希望这种情况不会发生）。如果无法将某些信息以语言、图片或声音的形式表达，那也不可能将这些信息以比特的形式编码。当然，你也不会想将它们编码。

举手或不举手是一个比特的信息。两个人是否举手 —— 就像电影评论家Roger Ebert和刚去

世不久的Gene Siskel对新影片提供他们最终的评价结果那样——传递两个比特的信息。（我们将忽略掉他们实际上对影片做的评语，而只关心他们有没有举手的问题。）这样，我们用两个比特代表四种可能：

00 = 他们都不喜欢这部影片
01 = Siskel 讨厌它，Ebert喜欢它
10 = Siskel 喜欢它，Ebert讨厌它
11 = Siskel和Ebert都喜欢它

第一个比特值代表Siskel的意见，0表示Siskel讨厌这部影片，1表示Siskel喜欢这部影片。同样，第二个比特值代表Ebert的意见。

因此，如果你的朋友问你Siskel和Ebert是怎么评价《Impolite Encounter》这部电影的，你不用回答“Siskel举手了，Ebert没有举手”或者“Siskel喜欢这部电影，Ebert不喜欢这部电影”，你可以简单地回答“么零”。你的朋友只要知道哪一位代表的是Siskel的意见，哪一位代表的是Ebert的意见，并且知道值为1代表举手，值为0代表没有举手，你的回答就是可以被人理解的。当然，你和你的朋友都要知道这种代码的含义。

我们也可以一开始就声明值为1的比特位表示没有举手，值为0的比特位表示举手了，这可能有点违反常规。通常我们会认为值为1的比特位代表正面的事情，而值为0的比特位代表相反的一方面，这的确只是一种很随意的指派。无论怎样，用此种代码的人只要明白0、1分别代表什么就可以了。

某一位或几位比特位的集合所代表的意义通常是和上下文相关的。橡树上的黄绸带可能只有系绸带的人和期望看到绸带的人知道其中的意思，改变绸带的颜色、系绸带的树或系绸带的日期，绸带可能会被认为只是一块毫无意义的破布。同样，要从Siskel和Ebert的手势中得到有用的信息，我们至少要知道正在讨论的是哪部影片。

如果你保存了Siskel和Ebert对一系列影片的评价和投票结果，你就有可能在表示Siskel和Ebert意见的比特信息中再增加一位代表你自己的观点的比特位。增加的第三位使得其代表的信息可能性增加到8种：

000 = Siskel讨厌它，Ebert讨厌它，我讨厌它
001 = Siskel讨厌它，Ebert讨厌它，我喜欢它
010 = Siskel讨厌它，Ebert喜欢它，我讨厌它
011 = Siskel讨厌它，Ebert喜欢它，我喜欢它
100 = Siskel喜欢它，Ebert讨厌它，我讨厌它
101 = Siskel喜欢它，Ebert讨厌它，我喜欢它
110 = Siskel喜欢它，Ebert喜欢它，我讨厌它
111 = Siskel喜欢它，Ebert喜欢它，我喜欢它

使用比特来表示信息的一个额外好处是我们清楚地知道我们解释了所有的可能性。我们知道有且仅有8种可能性，不多也不少。用3个比特，我们只能从0数到7，后面再没有3位二进制数了。

在描述Siskel和Ebert的比特时，你可能一直在考虑一个严重的，并且是令人烦恼的问题——对于Leonard Maltin的Movie &Video Guide怎么办呢？别忘了，Leonard Maltin是不采用举

手表决这种形式的，他对电影的评价用的是更传统的星级系统。

要想知道需多少个Maltin比特，首先要了解一些关于Maltin评分系统的知识。Maltin给电影的评价是1～4颗星，并且中间可以有半颗星。（仅仅是为了好玩，他实际上不会给电影只评一颗星，取而代之的是给一个BOMB［炸弹］。）这里总共有七种可能性，也就是说只需要3个比特位就可以表示一个特定的评价等级了：

000 = BOMB
001 = ★1/2
010 = ★★
011 = ★★1/2
100 = ★★★
101 = ★★★1/2
110 = ★★★★

你可能会问111怎么办呢，111这个代码什么意义都没有，它没有定义。如果二进制代码111被用来表示Maltin等级，那一定是出现错误了。（这可能是计算机出的错误，因为人不会给出这样的评分。）

前面我们曾用两个比特来代表Siskel和Ebert的评价结果，左边的一位代表Siskel的评价意见，右边的一位代表Ebert的评价意见。在上述Maltin评分系统中，各个比特位都有确定的意义吗？是的，当然有。将比特编码的数值加2再除以2，就得到了Maltin评分中对应的星的颗数。这样编码是由于我们在定义代码时遵循了合理性和连贯性，我们也可以下面的这种方式编码：

000=★★★
001=★1/2
010=★★1/2
011=★★★★
101=★★★1/2
110=★★
111=BOMB

只要大家都了解代码的含义，这种表示就和前述代码一样，都是合理的。

如果Maltin遇到了一部连一颗星都不值得给的电影，他就会给它半颗星。他当然有足够的代码来表示半颗星的情况，代码会像下面这样定义：

000=MAJOR BOMB
001=BOMB
010=★1/2
011=★★
100=★★1/2
101=★★★
110=★★★1/2
111=★★★★

但是，如果他再遇到连半颗星的级别都不够的影片并且决定给它没有星的级别（ATOMIC BOMB？），他就得再需要一个比特位了，已经没有3个比特的代码空闲了。

《Entertainment Weekly》杂志常常给事物定级，除了电影之外还有电视节目、CD、书籍、CD-ROM、网络站点等等。等级的范围从A+～F，如果你数一下的话，发现共有13个等级。这样，需要四个比特来代表这些等级：

0000 = F
0001 = D-
0010 = D
0011 = D+
0100 = C-
0101 = C
0110 = C+
0111 = B-
1000 = B
1001 = B+
1010 = A-
1011 = A
1100 = A+

有3个代码没有用到，它们是：1101、1110和1111，加上后总共是16个代码。

只要谈到比特，通常是指特定数目的比特位。拥有的比特位数越多，可以传递的不同可能性就越多。

对十进制数当然也是同样的道理。例如，电话号码的区号有几位呢？区号共有3位数字。如果所有的区号都使用的话（实际上有一部分区号并没有使用，将它们忽略），一共有10^3或1000个代码，从000～999。区号为212的7位数的电话号码有多少种可能呢？10^7或10 000 000个；区号为212并且以260开头的电话号码有多少个呢？104或10 000个。

同样，在二进制数中，可能的代码数等于2的比特位数次幂：

比特位数	代码数
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

每增加一个比特位，二进制代码数翻一番。

如果知道需要多少个代码，那么怎样才能知道需要多少个比特位呢？换句话说，在上述表中，如何才能由代码数反推出比特位数呢？

用到的方法叫作取以2为底的对数，对数运算是幂运算的逆运算。我们知道2的7次幂等于

128，以2为底的128的对数就等于7。用数学记号来表示第一个句子为：

$$2^7 = 128$$

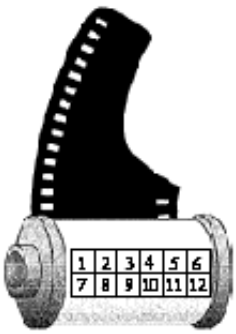
它与下述句子等价：

$$\log_2 128 = 7$$

因此，如果以2为底的128的对数等于7，以2为底的256的对数等于8，那么，以2为底的200的对数等于多少呢？大约是7.64，但实际上并不需要知道它。如果要表示200种不同的事物，我们共需要8个比特。

比特通常无法从日常观察中找到，它深藏于电子设备中。我们看不到压缩磁盘（CD）、数字手表或计算机中编过码的比特，但有时候比特也可以清晰地看到。

下面就是一个例子。如果你手头有一个使用35毫米胶片的相机，观察一下它的卷轴。这样拿住胶卷：



胶卷上有像国际跳棋棋盘一样的银色和黑色方格，方格已用数字1～12标识。这叫作DX编码，这12个方格实际上是12个比特。一个银色的方格代表值为1的比特，一个黑色的方格代表值为0的比特。方格1和7通常是银色的（代表1）。

这些比特是什么意思呢？你可能知道有些胶片对光的敏感程度要比其他胶片强，这种对光的敏感程度称作胶片速度。说对光非常敏感的胶片很快是因为这种胶片的曝光速度快。曝光速度是由ASA（American standards association，美国标准协会）来制定等级的，最常用的等级有100、200和400。ASA等级不只是以十进制数字的形式印在胶卷的外包装和暗盒上，而且还以比特的形式进行了编码。

胶卷总共有24个ASA等级，它们是：

25	32	40
50	64	80
100	125	160
200	250	320
400	500	640
800	1000	1250
1600	2000	2500
3200	4000	5000

为ASA等级编码需要多少个比特呢？答案是5个比特。我们知道，$2^4=16$，与24比太小了；$2^5 = 32$，又超过了所需的编码数。

比特值与胶片速度的对应关系如下所示：

方格2	方格3	方格4	方格5	方格6	胶片速度
0	0	0	1	0	25
0	0	0	0	1	32
0	0	0	1	1	40
1	0	0	1	0	50
1	0	0	0	1	64
1	0	0	1	1	80
0	1	0	1	0	100
0	1	0	0	1	125
0	1	0	1	1	160
1	1	0	1	0	200
1	1	0	0	1	250
1	1	0	1	1	320
0	0	1	1	0	400
0	0	1	0	1	500
0	0	1	1	1	640
1	0	1	1	0	800
1	0	1	0	1	1000
1	0	1	1	1	1250
0	1	1	1	0	1600
0	1	1	0	1	2000
0	1	1	1	1	2500
1	1	1	1	0	3200
1	1	1	0	1	4000
1	1	1	1	1	5000

多数现代的35毫米照相机胶片用的都是这些代码（除了那些要手工进行曝光的相机和具有内置式测光表但需要手工设置曝光速度的相机以外）。如果你看过照相机的内部放置胶卷的地方，你应该能够看到和胶片的金属方格（1～6号）相对应的6个金属可接触点。银色方格实际上是胶卷暗盒中的金属，是导体；油漆了的黑色方格，是绝缘体。

照相机的电子线路中有一支流向方格1的电流，方格1通常是银色的。这支电流有可能流到方格2～6，这要依方格中是纯银还是涂了油漆而定。这样，如果照相机在接触点4和5检测到了电流而在接触点2、3和6没有检测到，胶片的速度就是400ASA。照相机可以据此调节曝光时间。

廉价的照相机只要读方格2和方格3，并且假定胶片速度是50、100、200或400ASA四种可能速度之一。

多数相机不读方格8～12。方格8、9、10用来对这卷胶卷进行编码；方格11和12指出曝光范围，依胶片用于黑白照片、彩色照片还是幻灯片而定。

也许最常见的二进制数的表现形式是无处不在的UPC（universal product code，通用产品代码），即日常所购买的几乎所有商品包装上的条形码。条形码已经成为计算机在日常生活中应用的一种标志。

尽管UPC常常使人多疑，但它确实是一个无辜的小东西，发明出来仅仅是为了实现零售业的结算和存货管理的自动化，且其应用是相当成功的。当它和一个设计精良的结算系统共同使用时，顾客可以拿到列出细目的售货凭条，这一点是传统现金出纳员所无法做到的。

有趣的是，UPC也是二进制代码，尽管它初看起来并不像。将UPC解码并看看UPC码具

体是怎样工作的是很有益的。

通常情况下，UPC是30条不同宽度的垂直黑色条纹的集合，由不同宽度的间隙分割开，其下标有一些数字。例如，以下是Campbell公司$10\frac{3}{4}$盎司的罐装鸡汁面包装上的UPC：

可将条形码形象地看成是细条和黑条，窄间隙和宽间隙的排列形式，事实上，这是观察条形码的一种方式。黑色条有四种不同的宽度，较宽的条的宽度是最细条的宽度的两倍、三倍或者四倍。同样，各条之间的间隙中较宽的间隙是最窄间隙的两倍、三倍或者四倍。

但是，看待UPC的另一种方式是将它看作是一系列的比特。记住，整个条形码与条形码扫描仪在结算台“看”到的并不完全一样。扫描仪不会识别条形码底部的数字，因为识别数字需要一种更复杂的技术——光学字符识别技术，又称作OCR（optical character recognition）。实际上，扫描仪只识别整个条形码的一条窄带，条形码做得很大是为了便于结算台的操作人员用扫描仪对准顾客选购的物品。扫描仪所看到的那一条窄带可以这样表示：

它看上去是不是很像摩尔斯编码？

当计算机自左向右进行扫描时，它给自己遇到的第一个条分配一个值为1的比特值，给与条相邻的间隙分配一个值为0的比特值。后续的间隙和条被当作一行中一系列比特中的1个、2个、3个还是4个比特读进计算机要依据条或间隙的宽度而定。扫描进来的条形码的比特形式很简单：

因此，整个UPC只是简单的由95个比特构成的一串。本例中，这些比特可以像下面这样分组：

10100011010110001001100100011010001101000110101010111001011001101101100100111011001101000100101

比特	意义
101	最左边的护线
0001101	左边的数字
0110001	
0011001	
0001101	
0001101	
0001101	
01010	中间的护线
1110010	右边的数字
1100110	
1101100	
1001110	
1100110	
1000100	
101	最右边的护线

起初的3个比特通常是101，这就是最左边的护线，它帮助计算机扫描仪定位。从护线中，扫描仪可以知道代表单个比特的条或间隙的宽度，否则，所有包装上的 UPC印刷大小都是一样的。

紧挨着最左边的护线是每组有 7个比特位的六组比特串，每一组是数字 0～9的编码之一，我们在后面将证明这一点。接着的是 5个比特的中间护线，此固定模式（总是 01010）是一种内置式的检错码。如果扫描仪在应当找到中间护线的地方没有找到它，扫描仪就认为那不是UPC。中间护线是防止条形码被窜改或错印的方法之一。

中间护线的后面仍是每组 7个比特的6组比特串。最后是最右边的护线，也总是 101。最后的最右护线使得 UPC反向扫描（也就是自右向左扫描）同正向扫描一样成为可能，这一点我们将在后面解释。

因而整个UPC对12个数字进行了编码。左边的 UPC包含了6个数字的编码，每个数字占有7个比特位。你可以用下表进行解码：

左边的编码

0001101=0	0110001=5
0011001=1	0101111=6
0010011=2	0111011=7
0111101=3	0110111=8
0100011=4	0001011=9

注意，每个7位代码都是以0开头，以1结尾的。如果扫描仪遇到了第一个比特位值为 1或最后一个比特位值为 0的情况，它就知道自己没有将 UPC正确地读入或者是条形码被窜改了。另外我们还注意到每个代码都仅有两组连续的值为 1的比特位，这就意味着每个数字对应着条形码中的两个竖条。

上表中的每个代码中都包含有奇数个值为 1的比特位，这也是用于检测差错和数据一致性的一种机制，称为奇偶校验。如果一组比特位中含有奇数个 1，就称之为奇校验；如果含有偶数个1，就称之为偶校验。这样看来，所有这些代码都拥有奇校验。

为了给UPS右边的7位一组的数字解码，可以采用下面的表格：

右边的编码

1110010=0	1001110=5
1100110=1	1010000=6
1101100=2	1000100=7
1000010=3	1001000=8
1011100=4	1110100=9

这些代码都是前述代码的补码或补数：凡是 1的地方都换成0，凡是0的地方都换成1。这些代码都是以1开始，以零结束，并且每组都有偶数个 1，称之为偶校验。

现在，可以对UCP进行解码了。借助前两个表格， Campbell公司$10\frac{3}{4}$盎司的罐装鸡汁面的包装上用UPC编码的12个数字是：

0 51000 01251 7

这个结果是令人失望的，正如你所看到的那样，它们和印在 UPC底部的数字完全相同。

（这样做是有意义的，因为由于某种原因，扫描仪可能无法识别条形码，收银员就可以手工将这些数字输进去。）我们还没有完成解码的全部任务，而且，我们也无法从中解码任何秘密信息。然而，关于UPC的解码工作已经没有了，那30个竖条已经变成了12个数字。

第一个数字（在这里是0）被称为数字系统字符，0的意思是说这是一个规范的UPC编码。如果是具有不同重量的货物的UPC（像肉类或其他商品），这个数字是2；订单、票券的UPC编码的第一个数字通常是5。

紧接着的5个数字是制造商代码。在上例中，51000是Campbell鸡汁面公司的代码。Campbell公司生产的所有产品都使用这个代码。再后面的5个数字（01251）是该公司的某种产品的编号，上例中是指$10\frac{3}{4}$盎司的罐装鸡汁面。别的公司的鸡汁面可能有不同的编号，且01251在另外一个公司可能是指一种完全不同的产品。

和通常的想法相反，UPC中没有包含该种产品的价格。产品的价格信息可以从商店中使用的与该扫描仪相联的计算机中检索互到。

最后的数字（这里是7）称作模校验字符，这个字符可用来进行另外一种错误检验。为了解释校验字符是怎样工作的，将前11个数字（是0 51000 01251）各用一个字母来代替：

A BCDEF GHIJK

然后，计算下式的值：

3×（A+C+E+G+I+K）+（B+D+F+H+J）

从紧挨它并大于等于它的一个10的整倍数中减去它，其结果称为模校验字符。在上例中，有：

3×（0+1+0+0+2+1）+（5+0+0+1+5）=3×4+11=23

紧挨23并大于等于23的一个10的整倍数是30,故：

30－23=7

这就是印在外包装上并以UPC形式编码的模校验字符，这是一种冗余措施。如果扫描仪计算出来的模校验结果和UPC中编码中的校验字不一致，计算机就不能将这个UPC作为一个有效值接收。

正常情况下，表示从0～9的十进制数字只需4个比特就足够了。在UPC中，每个数字用了7个比特，这样总共有95个比特来表示11个有用的十进制数字。事实上，UPC中还包括空白位置（相当于9个0比特），位于左、右护线的两侧。因而，总共有113个比特用来编码11个十进制数，平均每个十进制数所用超过了10个比特位！

正像我们所知道的那样，有部分冗余对于检错来讲是必要的。这种商品编码如果能够很容易地被顾客用粗头笔修改的话，这种代码措施也就难以发挥其作用了。

UPC编码可以从两个方向读，这一点是非常有益的。如果扫描仪解码的第一个数字是偶校验（即：每7位编码中共有偶数个1），扫描仪就知道它正在从右向左进行解码。计算机系统用下表对右边的数字解码：

逆向时右边数字的代码

0100111 = 0	0111001 = 5
0110011 = 1	0000101 = 6
0011011 = 2	0010001 = 7
0100001 = 3	0001001 = 8
0011101 = 4	0010111 = 9

下面是对左边数字的解码表：

逆向时左边数字的代码

1011000 = 0	1000110 = 5
1001100 = 1	1111010 = 6
1100100 = 2	1101110 = 7
1011110 = 3	1110110 = 8
1100010 = 4	1101000 = 9

这些7位编码与扫描仪由左向右扫描时所读到的编码完全不同，但不会有模棱两可的现象。

让我们再看看本书中提到的由点、划组成其间用空格分开的摩尔斯电码。摩尔斯电码看上去不像是由0和1组成的，但它确实是。

下面回忆一下摩尔斯电码的编码规则：划的长度等于点长度的三倍；单个的点或划之间用长度与点的长度相等的空格分开；单词内的各个字母之间用长度等于划的长度的空格分隔；各单词之间由长度等于两倍的划长度的空格分开。

为使分析更加简单，我们假设划的长度是点长度的两倍而不是 3倍。也就是说，一个点是一个值为1的比特位，一个划是两个值为 1的比特位，空格是值为0的比特位。

下面是第2章的摩尔斯电码的基本表：

A	·—	J	·———	S	···
B	—···	K	—·—	T	—
C	—·—·	L	·—··	U	··—
D	—··	M	——	V	···—
E	·	N	—·	W	·——
F	··—·	O	———	X	—··—
G	——·	P	·——·	Y	—·——
H	····	Q	——·—	Z	——··
I	··	R	·—·		

下面是将它转化为比特形式的结果：

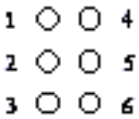
A	101100	J	1011011011000	S	1010100
B	110101010000	K	110101100	T	1100
C	110110110100	L	101101010000	U	101011000
D	110101000	M	110110000	V	101010110000
E	100	N	110100	W	101101100
F	101011010000	O	110110110000	X	110101011000
G	110110100	P	101101101000	Y	110101101100
H	101010000	Q	110110101100	Z	110110101000
I	10000	R	10110100		

注意，所有的编码都以1开头，以两个0结束。结尾处的两个零代表单词中各个字母之间的空格，单词之间的空格用另外的一对0来表示。因而，“Hi,there”的摩尔斯电码通常是这样的：

但是，采用比特形式的摩尔斯电码看起来像UPC编码的横切面：

101010100101000011001010101001001011010010000

用比特的形式表示布莱叶盲文比表示摩尔斯电码容易得多。布莱叶编码是6比特代码。布莱叶盲文中的每一个字母都是由6个点组成的，点可能是凸起的，或没有凸起(平滑)的。如在第3章中讲的那样，这些点通常用数字1～6编号：



例如，单词“code”可以用布莱叶盲文这样表示：

如果突起的点是1，平坦的点是0，则布莱叶盲文中的每一个符号都可以用6个比特的二进制代码表示。单词“code”中的四个布莱叶字母符号就可以简单地写成：

100100 101010 100110 100010

最左边的一位对应编号为1的位置，最右边的一位对应编码为6的位置。

正如前面所讲到的，比特可以代表单词、图片、声音、音乐、电影，也可以代表产品编码、胶片速度、电影的受欢迎程度、英军的入侵以及某人所挚爱的人的意愿。但是，最基本的一点是：比特是数字。当用比特表示信息时只要将可能情况的数目数清楚就可以了，这样就决定了需要多少个比特位，从而使得各种可能的情况都能分配到一个编号。

比特在哲学和数学的奇怪混合物——逻辑——中发挥作用。逻辑最基本的目标是证明某个语句是否正确，正确与否也可以用1和0来表示。

第10章　逻辑与开关

真理是什么呢？亚里士多德认为逻辑与它有关。他的讲义合集《工具论》（Organon，可追溯到公元前4世纪）是最早的关于逻辑的详细著作。对于古希腊人而言，逻辑是追寻真理的过程中用于分析语言的一种手段，因此它被认为是一种哲学。亚里士多德的逻辑学的基础是三段论。最有名的三段论（它并非是在亚里士多德的著作中发现的）是：

All men are mortal;
Socrates is a man;
Hence, Socrates is mortal.

(所有的人都是要死的;
苏格拉底是人;
所以,苏格拉底是要死的。)

在三段论中，两个前提被假设是正确的，并由此推出结论。

苏格拉底之死这个例子看上去似乎太直白了，但还有许多其他不同的三段论。例如，考虑下面两个由19世纪数学家Charles Dodgson（也就是Lewis Carroll）提出的前提：

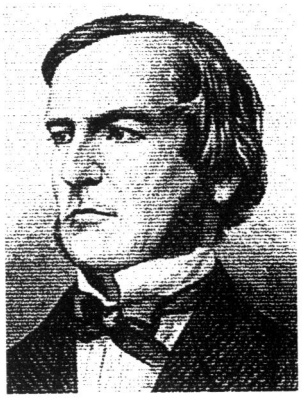
All philosophers are logical;
An illogical man is always obstinate.

(所有的哲学家都是有逻辑头脑的;
一个没有逻辑头脑的人总是顽固的。)

它所能推出的结论一点儿也不明显。（事实上，结论是“一些顽固的人不是哲学家(Some dostinate persons are not philosophers)”)请注意结论中一个出乎意料且令人迷惑的词“一些(some)”。

两千多年来，数学家们对亚里士多德的逻辑理论苦苦思索，试图用数学符号和操作符来表现它。19世纪以前，唯一能接近这个目标的人是莱布尼兹（1648—1716），他早年涉足逻辑学领域，后来转向其他学科（比如说，他几乎和牛顿同时独立地发明了微积分）。

接下来有所突破的是乔治·布尔。



乔治·布尔1815年生于英格兰，他周围的环境对他的成长很不利。他父亲是鞋匠，而母亲曾是女仆，英国森严的等级制度使布尔学不到什么有别于父辈的东西。但是，靠着他自身强烈的好奇心及父亲的帮助（其父对科学研究、数学和文学有浓厚的兴趣），年轻的乔治自学了上层阶级男孩才能学到的课程，包括拉丁文、希腊语及数学。由于他早年在数学方面发表的论文，1849年，布尔被任命为爱尔兰Cork市的皇后大学数学系的首席教授。

19世纪中期的几位数学家在逻辑理论的数学定义上做了一些工作（最著名的是迪摩根），但只有布尔有真正概

念上的突破。他最早的贡献是发表的一本很简短的书《 The Mathematical Analysis of Logic, Being an Essay Towards a Calculus of Deductive Reasoning》(1847)，接着又发表了一篇很长且充满抱负的文章：《An Investigation of the Laws of Thought on Which Are Founded the Mathematical Theories of Logic and Probabilities》(1854)，简称为《The Laws of Thought》。1864年的一天，布尔在雨中赶去上课时不幸感染上了肺炎，不治身亡，享年 49岁。

我们可以从布尔在 1854年所著书的题目中看出他富于野心的想法：由于充满理性的人脑用逻辑去思考，那么，如果能用数学来表征逻辑，我们也就可以用数学来描述大脑是如何工作的。当然，现在看来这种想法似乎十分幼稚。（但却超越了他所在的年代。）

布尔发明了一种和传统代数看起来、用起来都十分相似的代数。在传统代数中，操作数（通常是字母）代表数字，而操作符（多是“ +”或“ ×”）指明这些操作数如何结合到一起。一般我们可用传统代数解决类似下面的问题：如果安娜有 3磅豆腐，贝蒂的豆腐是安娜的 2倍，卡门的豆腐比贝蒂多 5磅，迪尔德丽的豆腐是卡门的 3倍。那么，迪尔德丽有多少豆腐呢？

为了计算这个问题，我们首先把语句转化为算术式子，用四个字母代表每个人拥有豆腐的数量，即：

$$A = 3$$
$$B = 2 \times A$$
$$C = B+5$$
$$D = 3 \times C$$

可以通过代入把上述四个表达式合为一个式子，最后执行加法和乘法，即：

$$D = 3 \times C$$
$$D = 3 \times （B+5）$$
$$D = 3 \times （（2 \times A）+5）$$
$$D = 3 \times （（2 \times 3）+5）$$
$$D = 33$$

当做传统代数题时，要遵循一定的规则。这些规则可能已经和实践融为一体，以至于我们不再认为它们是规则，甚至忘记了它们的名字。但规则确实是任何形式的数学的基础。

第一个规则是加法与乘法的交换律，即我们可以在操作符两边交换操作数的位置：

$$A+B = B+A$$
$$A \times B = B \times A$$

相反，减法和除法是不满足交换律的。

加法和乘法也满足结合律，即：

$$A+（B+C）=（A+B）+C$$
$$A \times （B \times C）=（A \times B）\times C$$

最后，乘法对加法可以进行分配：

$$A \times （B+C）=（A \times B）+（A \times C）$$

传统代数的另外一个特点是它总是处理数字，如豆腐的重量或鸭子的数量，火车行驶的距离或家庭成员的年龄。是布尔超凡的智慧使代数脱离了数字的概念而变得更加抽象。在布尔代数中（布尔的代数最终被这样命名）操作数不是指数字，而是指集（类）。一个类仅仅表示一组事物，也就是后来熟知的集合。

让我们来讨论一下猫。猫或公或母，为方便起见，我们用字母 M指代公猫的集合，用 F指代母猫的集合。记住，这两个符号并不代表猫的数量，公猫或母猫的数量随着小猫仔的出生和老猫的不幸离去而变化，这两个字母代表的是猫的种类 —— 具有某种特点的猫。因而我们不说公猫，而是用M来代表它们。

我们也可以用其他字母代表猫的颜色。例如，用 T代表黄褐色的猫，用 B代表黑猫，用W代表白猫，而用O代表所有其他颜色的猫。

最后（至少就这个例子而言），猫要么是阉过的要么是有生育能力的。我们用字母 N代表阉过的猫，而用U代表有生育能力的猫。

在传统代数中，操作符 +和×被用于表示加法和乘法。在布尔代数中，同样用到了 +和×。这似乎会引起混淆。人人都知道在传统代数中如何对数字进行加和乘，但是我们如何对“ 类 ”进行加和乘呢？

事实上，在布尔代数中我们并不真正地做加或乘，相反，这两个符号有着完全不同的意思。

在布尔代数中，符号 +意味着两个集合合并，两个集合的合并就是包含第一个集合的所有成员及第二个集合的所有成员。例如， B+W表示黑猫和白猫的集合。

布尔代数中的符号×意味着取两个集合的交集，两个集合的交集包含的元素既在第一个集合中，也在第二个集合中。例如， F×T代表了一种猫的集合，这个集合中的猫既是母猫又是黄褐色的。与传统代数一样，我们可以把 F×T写成F · T或简写为FT（这正是布尔代数所期望的）。你可以把这两个字母看成是连在一起的两个形容词：黄褐色的母猫。

为避免传统代数和布尔代数之间的混淆，有时候用符号∪和∩而不用 +和×来表示并运算和交运算。但布尔对数学的解放性的部分影响是使熟悉的操作符更加抽象，所以，我们决定坚持他的决定，而不为他的代数引入新的符号。

交换律、结合律和分配律在布尔代数中均适用。而且，在布尔代数中，操作符 +可以对×进行分配，这在传统代数中是不成立的，即：

$$W+（B\times F）=（W+B）\times（W+F）$$

这个式子表示白猫（W）和黑色母猫（B×F）的并集和等式右边两个集合的交集是一样的，这两个集合是白猫和黑猫的并集（ W+B）及白猫和母猫的并集（ W+F）。要掌握这个规则有些困难，但它的确有用。

为了使布尔代数更加完整，我们还需要两个符号。这两个符号看上去像数字，但它们并不真的是数字，因为有时候它们和数字有些不同。符号“ 1 ”在布尔代数中表示“ 整个宇宙(全集) ”，也就是我们所谈论的每件事物。本例中，符号“ 1 ”表示“ 所有的猫 ”。这样：

$$M+F=1$$

即母猫和公猫的并集是所有的猫。同样，黄褐色猫、黑猫、白猫及其他颜色的猫的并集也是所有的猫，即：

$$T+B+W+O=1$$

你也可以这样表示所有的猫：

$$N+U=1$$

符号1可以用一个减号－来排除一些事物。例如：

$$1-M$$

表示除了公猫以外的所有猫。排除公猫以后的全集就是母猫的集合：

$$1-M=F$$

我们所需要的另外一个符号是“ 0 ”。在布尔代数中，“ 0 ”表示空集，即不含任何事物的集合。当求取两个完全相互排斥的集合的交集时，空集就产生了。例如，既是母的又是公的猫的集合可以表示为：

$$F\times M=0$$

注意，符号1和0有时的用法与传统代数相同。例如，所有的猫和母猫求交集即是母猫这个集合：

$$1\times F=F$$

空集和母猫求交集还是空集：

$$0\times F=0$$

空集和母猫的并是母猫这个集合：

$$0+F=F$$

但有时与传统代数中得到的结果就不太一样了。例如，所有的猫和母猫的并集是所有的猫：

$$1+F=1$$

这个表达式在传统代数中是没有意义的。

由于F代表母猫的集合，1－F代表所有其他猫的集合，则这两个集合的并集是 1：

$$F+(1-F)=1$$

并且它们的交集是0：

$$F\times(1-F)=0$$

历史上，这个公式代表了逻辑中一个十分重要的概念，即矛盾律。它表明一个事物不能同时是它自己和它自己的反面。

使布尔代数和传统代数看起来完全不同的是下面这个表达式：

$$F\times F=F$$

这个式子在布尔代数中有着完美的意义：母猫的集合和母猫的集合的交集仍旧是母猫的集合。但若F代表一个数字的话，这个公式显然就不对了。布尔认为：

$$X^2=X$$

是使他的代数与传统代数区分开来的唯一表达式。另一个按照传统代数看起来很有趣的布尔表达式是：

$$F + F = F$$

母猫和母猫的并集仍是母猫这个集合。

布尔代数为解决亚里士多德的三段论提供了一个数学方法。再看看这个著名三段论的两个前提：

所有的人都是要死的；

苏格拉底是人。

我们用字母P代表所有人的集合，M代表要死的东西的集合，S代表苏格拉底。那么，所谓“所有的人都是要死的”意味着什么呢？它其实表示了所有人的集合和所有要死的东西的集合的交集是所有的人这个集合，即：

$$P \times M = P$$

而 $P \times M = M$ 这个式子是错误的，因为要死的东西还包括猫、狗、榆树等等。

而“苏格拉底是人”意味着苏格拉底这个集合（非常小）和所有人的集合（很大）的交集是苏格拉底这个集合：

$$S \times P = S$$

由于从第一个式子中知道 $P=P \times M$，所以可以把它代入第二个式子，即：

$$S \times (P \times M) = S$$

根据结合律，上式等同于：

$$(S \times P) \times M = S$$

但我们已经知道 $S \times P$ 等于S，所以上式可简化为：

$$S \times M = S$$

现在计算完毕。这个表达式告诉我们，苏格拉底和所有要死东西的集合的交集是苏格拉底，也就是说苏格拉底是要死的。相反，如果认为 $S \times M$ 等于0，那么结论就是苏格拉底不会死。再如果，若 $S \times M$ 等于M，则能推出的结论就是苏格拉底是唯一会死去的东西，而其他任何东西都是不朽的！

用布尔代数来证明显而易见的事实似乎有些小题大做（尤其当考虑到苏格拉底早已在2400年以前就去世了时），不过，布尔代数还可以用来判断一些事物是否满足一定的标准。也许有一天，你走进宠物店对店员说：“我想要一只阉过的公猫，白的或黄褐色的均可；或者要一只没有生殖能力的母猫，除了白色，其他任何颜色均可；或者只要是只黑猫，我也要。”店员对你说：“看来您想要的猫是下面的式子表示的集合中的一只：

$$(M \times N \times (W+T)) + (F \times N \times (1-W)) + B$$

对吗？”你回答道：“是的，完全正确！”

为了证明店员是正确的，你可能想放弃并和交的概念而转向“OR（或者/或）”和“AND（并且/与）”。大写这两个词是因为虽然在通常情况下它们代表语言中的概念，但它们也代表了布尔代数中的操作。当求两个集合的并集时，你实际上是从第一个集合“或”从第二个集合中取得事物放入结果集合里。当求两个集合的交集时，满足条件的事物必定在第一个集合中“并且”也在第二个集合中。此外，每当你看见后跟减号的1，你可以使用单词“NOT（非）”

来表示。小结如下：

- +（以前表示求并集）现在表示OR。
- ×（以前表示求交集）现在表示AND。
- 1-（以前表示从全集中排除一些事物）现在表示NOT。

这样，刚才的表达式可以写成下面的形式：

（M AND N AND （W OR T））OR（F AND N AND （NOT W））OR B

这与你的口头描述已经十分接近了。注意圆括号是如何清楚地表达出你的意图的。你想要的猫来自下面三个集合之一：

（M AND N AND（W OR T））

或

（F AND N AND （NOT W））

或

B

写下这个公式后，店员就可以进行布尔测试的工作了。别这么大惊小怪的，这里已经悄悄转移到另一种不同形式的布尔代数中去了。在这种形式的布尔代数中，字母不再只表示集合，字母还可以被赋予数字，但需要注意的是它们只能被赋予 0或者1。数字1表示“是的”、“正确”，本例中的意思是“这只猫符合我的要求”；数字 0表示“否定”、“错误”、本例中即“这只猫不符合我的要求”。

首先，店员拿出一只未阉过的黄褐色的公猫。下面是满足条件的猫的集合：

（M×N×（W+T））+（F×N×（1－W））+B

当用0和1代替字母后就变成了下面的样子：

（1×0×（0+1））+（0×0×（1-0））+0

注意被赋予了1的字母只有M和T，因为拿来的这只猫是公的，黄褐色的。

现在必须要做的是简化这个表达式。如果简化后表达式的结果是 1，这只猫就满足了你的要求，否则就不是你想要的猫。当简化表达式时，千万记住我们并不是在真正地做加法和乘法。当+表示OR，×表示AND时，大部分规则是相同的。（现代课本中有时用∧和∨分别表示AND和OR，而不用×和+；但这里用+和×这两个符号却是恰到好处的。）

当用×表示AND时，可能的结果是：

0×0 = 0

0×1 = 0

1×0 = 0

1×1 = 1

换句话说，只有当×的左、右两个操作数均为 1时，结果才为1。这个过程和普通乘法一模一样。若用一张小表总结一下，你会发现它们和第 8章的加法表和乘法表的形式相似：

AND	0	1
0	0	0
1	0	1

当用+表示OR时，可能的结果是：

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 1

当+的左、右操作数中有一个为1时，结果就是1。除了1+1=1这种情况，这种计算和普通加法产生的结果是一致的。可用另一张小表来总结：

OR	0	1
0	0	1
1	1	1

现在可以用这些表来计算前面那个表达式的结果了：

（1×0×1）+（0×0×1）+0 = 0 + 0 + 0 = 0

结果是0，表示“否定”、“错误”，即这只小猫不满足客户需求。

接下来，店员拿来一只无生育能力的白色的小母猫。原始表达式是：

（M×N×（W+T））+（F×N×（1－W））+B

把0和1代入上式：

（0×1×（1+0））+（1×1×（1-1））+0

并且把它简化一下：

（0×1×1）+（1×1×0）+0=0+0+0=0

看来，这只可怜的小猫还是不符合要求。

然后，店员又拿来一只无生育能力的灰色的小母猫。（灰色是非白色、黑色或黄褐色的一种其他颜色。）下面是表达式：

（0×1×（0+0））+（1×1×（1-0））+0

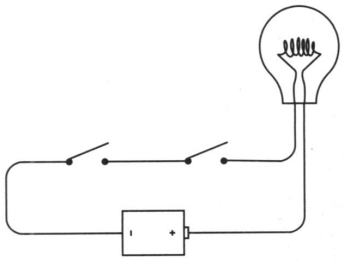
现在把它简化为：

（0×1×0）+（1×1×1）+ 0 = 0 + 1 + 0 = 1

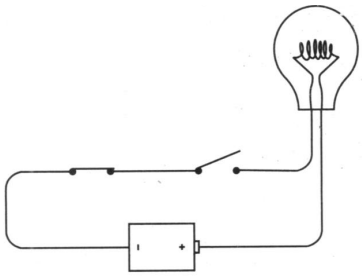
最后的结果1表示“是的”、“正确”，这只小猫总算找到新家了！

在你买到小猫的那天晚上，当小猫蜷身睡在你的腿上时，你开始考虑是否能够通过电线连接一些开关和灯泡来决定哪些小猫满足你的要求。（你真是一个奇怪的家伙。）你丝毫没有意识到你将要实现一个关键概念上的突破。你要做的是一些试验，这些试验把布尔代数和电路结合在一起，从而使使用二进制数字工作的计算机的设计和制造成为可能。（可别让这些话吓着你。）

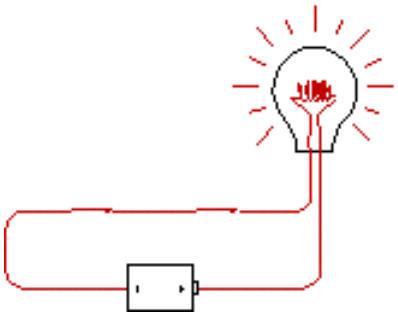
下面就开始了。你像往常一样把灯泡和电池连接在一起，这一回你用了两个开关：



开关这种方式的连接——一个在另一个的右边——称为串联的。如果你闭合了左边的开关，什么也不会发生：



同样，如果你让左边的开关断开而闭合右边的开关，结果还是一样。只有当左右两个开关都闭合时，灯泡才会发光，如下所示：



这里的关键是“都”。只有左边和右边的开关都闭合时，电流才能流过回路。

这个电路执行了一个逻辑运算。事实上，灯泡回答了这个问题：“两个开关都处于闭合状态吗？”可以把电路的工作总结为下面这张表：

左开关状态	右开关状态	灯泡状态
断开	断开	不亮
断开	闭合	不亮
闭合	断开	不亮
闭合	闭合	亮

在前一章中，我们已知道二进制数字（或“位”）是如何表示信息的：它可以表示从最普通的数字到Roger Ebert的拇指方向等的一切事情。可以说“0”代表“Ebert拇指向下的方向”，而“1”表示“Ebert拇指向上的方向”。一个开关有两个位置，所以它可以代表一个位。“0”表示“开关是断开的”，而“1”表示“开关是闭合的”。一个灯泡有两种状态，所以它也可以表示一个二进制位。“0”表示“灯泡不亮”而“1”表示“灯泡亮”。现在可以把上面的表简化一下：

左开关状态	右开关状态	灯泡状态
0	0	0
0	1	0
1	0	0
1	1	1

注意，如果交换左、右开关，结果是一样的，所以没必要指明哪个开关是左开关或右开关。因此这张表可以重画成类似于前面“AND”表和“OR”表的样子：

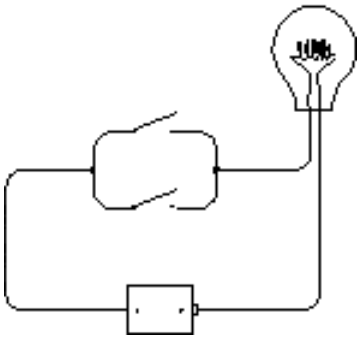
开关串联	0	1
0	0	0
1	0	1

事实上，这和“ AND ”表是一样的。让我们检查一下：

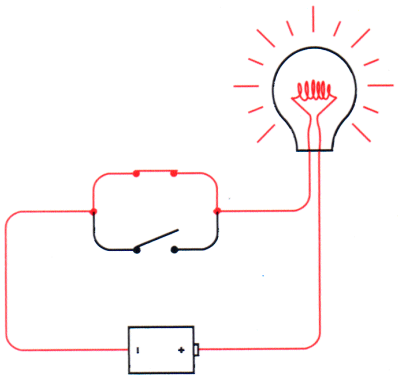
AND	0	1
0	0	0
1	0	1

这个简单的电路实际上执行了布尔代数的“ AND ”操作。

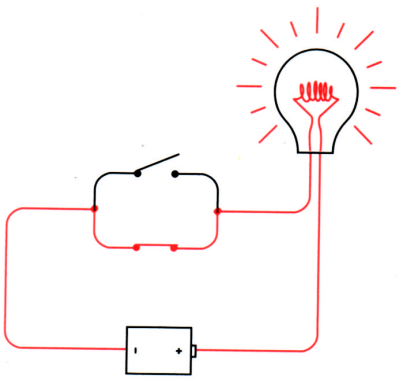
现在试着用另一种方式连接电路：



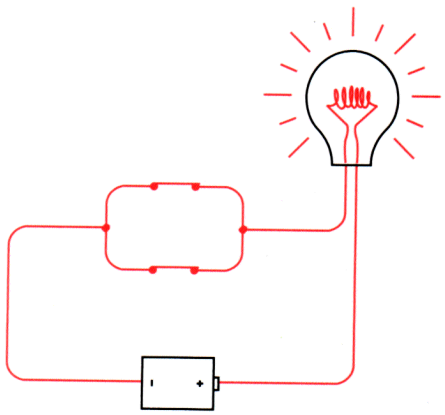
这些开关称为并行连接。它和前一种连接方式的区别是，如果闭合了上面的开关，灯泡会亮：



如果闭合了下面的开关，灯泡会亮：



如果同时闭合上、下两个开关，灯泡还是会亮：



可见，当上面或下面的开关有一个闭合时，灯泡就会亮。这里的关键字是“或”。

这个电路也执行了一个逻辑运算，灯泡回答了这样一个问题：“是否有开关闭合？”下面的表总结了这个电路是如何工作的：

上开关状态	下开关状态	灯泡状态
打开	打开	不亮
打开	闭合	亮
闭合	打开	亮
闭合	闭合	亮

仍然用“0”表示开关断开或灯泡不亮，用“1”表示开关闭合或灯泡亮。这张表可以这样：

上开关状态	下开关状态	灯泡状态
0	0	0
0	1	1
1	0	1
1	1	1

同样，这两个开关交换位置也没关系，所以这张表可以重写成如下的样子：

开关并联	0	1
0	0	1
1	1	1

你可能已经猜到了这和布尔代数中的“OR”表是一样的：

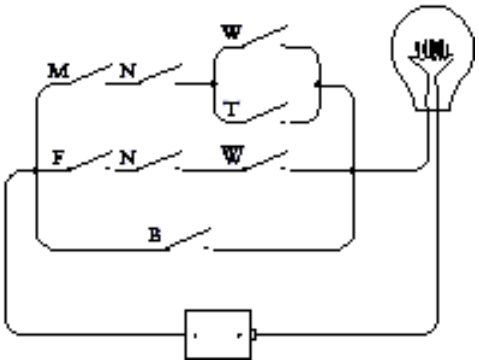
OR	0	1
0	0	1
1	1	1

这意味着两个并联的开关执行的是和布尔一样的操作。

当你再进入宠物店时，你告诉店员：“我想要一只阉过的公猫，白的或黄褐色的均可；或者要一只没生育能力的母猫，除了白色，其他任何颜色均可；或者只要是只黑猫，我也要。”店员便得到了如下的表达式：

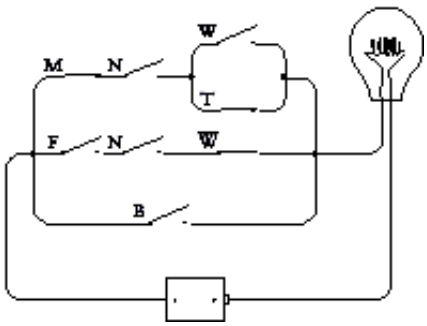
（M×N×（W+T））+（F×N×（1－W））+B

现在你知道两个串联开关执行的是逻辑与（AND，由符号×来表示），两个并联开关执行的是逻辑或（OR，由符号+来表示），你可以按如下方法连接8个开关：

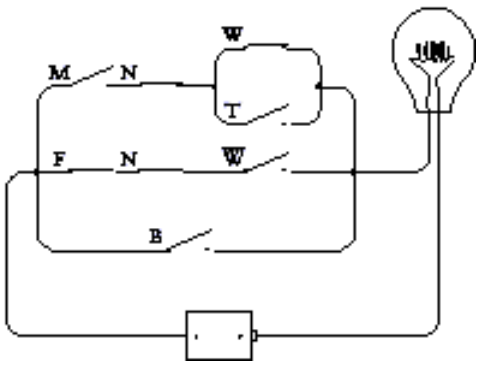


这个电路中的每一个开关都被标上了一个字母（与布尔表达式中所用字母相同）。$\overline{W}$ 表示非W，是1－W的另一种写法。事实上，如果按从左至右，从上至下的顺序来阅读这个电路图，你遇到的字母的顺序和它们在布尔表达式中出现的次序是一样的。表达式中的乘号(×)都对应角是电路图中串联的两个或两组开关的位置；表达式中的加号（＋)号对应的是电路图中并联的两个或两组开关的位置。

你应该记得，店员最先挑出的是只未阉过的褐色的公猫。闭合相应的开关：

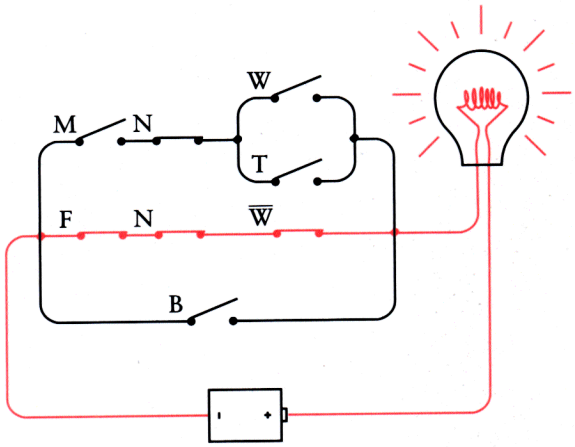


尽管M、T和非W这三个开关都闭合了，但没有构造出一个完整的电路来点亮灯泡。接着，店员拿出一只无生育能力的白色的母猫：



这次，由于右边开关未闭合也无法构成一个完整的电路。但最后，店员拿出一只无生育

能力的灰色的母猫：



这样就可以构出一个完整的电路，灯泡被点亮并表示小猫符合你的要求。

乔治·布尔从来没有连接过这样一个电路，他也没能看到用开关、电线和灯泡来实现一个布尔表达式。当然，其中的一个原因是直到布尔死后 15年，白炽灯泡才被发明。但摩尔斯在1844年展示了他的电报机，比布尔的《The Laws of Thought》的发表早10年，而用一个电报发声器来代替所示电路中的灯泡是十分简单的。

可惜19世纪没有人把布尔代数中的与、或和串联、并联一些简单的开关联系起来。数学家没有、电工没有、电报操作员也没有，没有人想到过这种联系，甚至连计算机革命的创始人查尔斯·巴贝芝（1792—1871）也没有。他曾和布尔联系过，并了解过他的工作，他一生中大部分时间致力于设计第一台差分机及接下来的解析机。一个世纪之后，这些机器被认为是现代计算机的雏型。我们现在知道，帮助巴贝芝的是他认识到计算机应产生于电报继电器中，而非那些齿轮和控制杆。

是的，问题的答案正是电报继电器。

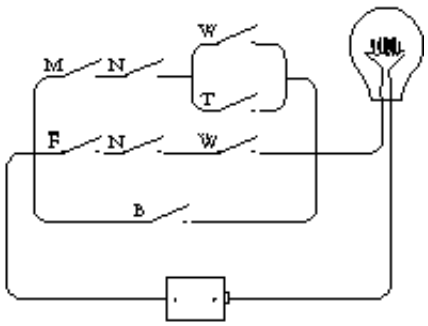
第11章　逻辑门电路

在遥远的将来，当人们回顾20世纪的计算机发展史时，有人可能会以为一种称为“logic gates（逻辑门）”的设备是以著名的微软公司创始人的名字命名的（Bill Gates中的Gates在英语中有“门”的意思），其实并非如此。我们很快就会明白，逻辑门和通常让水和人通过的门十分相似。逻辑门通过阻挡或允许电流通过在逻辑中执行简单的任务。

回忆一下在上一章中你走进一个宠物店所要的那只猫，这可以由下面的布尔表达式说明：

$$(M \times N \times (W+T)) + (F \times N \times (1-W)) + B$$

同时，也可以用下面的电路来选择符合条件的小猫：



这样一个电路有时被称为网络。但在今天，网络这个词更多地被用来指连接起来的计算机，而不仅仅只是开关的集合。

尽管这个电路包含的全是19世纪发明的东西，但那时却没有人意识到布尔代数可以直接由电路实现。这种等同性直到20世纪30年代才被发现，主要贡献人是克劳德·香农(生于1916年)。香农在他著名的、于1938年在麻省理工学院所写的硕士论文《A Symbolic Analysis of Relay and Switching Circuits》中阐述了这个问题。（10年之后，香农的文章The Mathematical Theory of Communication》是使用“位(bit)”这个字来表示二进制数字的第1篇出版物。）

1938年以前，人们已经知道当把两个开关串联起来时，只有两个开关都闭合电流才能流通；而当把两个开关并联起来时，只需闭合其中的一个即可构成回路。但没有人能像香农那样清晰地阐述电子工程师可以使用布尔代数的所有工具来设计带开关的电路。此外，如果你简化了描述网络的布尔表达式，你也可以相应地简化网络。

例如，描述你想要的小猫的表达式是：

$$(M \times N \times (W+T)) + (F \times N \times (1-W)) + B$$

用结合律把用×结合的变量重新排序并按下面的方式重写表达式：

$$(N \times M \times (W+T)) + (N \times F \times (1-W)) + B$$

为更清楚地表达意图，可以定义名为X和Y的两个新变量：

$$X = M \times (W+T)$$

$$Y = F \times (1 - W)$$

现在，描述你想要的小猫的表达式可以写成下面的样子：

$$(N \times X) + (N \times Y) + B$$

完成简化后，我们再把X、Y代回原来的式子。

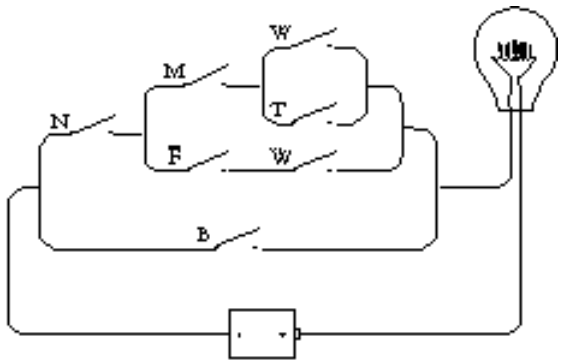
注意，变量N在表达式中出现了两次。使用分配律，表达式可以按如下方式重写，并只使用一个N：

$$(N \times (X+Y)) + B$$

现在把X、Y表达式代入：

$$(N \times ((M \times (W+T)) + (F \times (1-W)))) + B$$

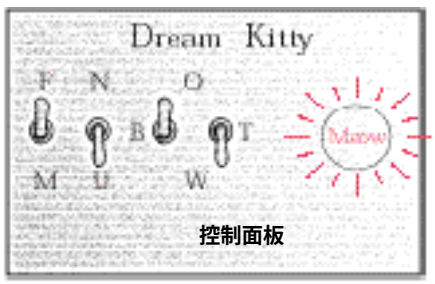
由于有很多圆括号，该表达式看上去似乎仍很复杂。但表达式中少了一个变量项（减少了一次×运算），也就意味着网络中少了一个开关。这是修改后的电路图：



确实，证明修改前后的两个电路图功能是一样的比去证明两个表达式功能是相同的要简单。

可是，网络中仍然多余了三个开关。理论上讲，你只需要四个开关来定义你心目中的猫咪。为什么是四个呢？因为每个开关都是一个“位”。你需要一个开关来定义性别（断开表示公的，而闭合表示母的）；一个开关来定义是否有生育能力 (闭合表示阉过的，断开表示未阉过的)还需要两个开关表示颜色。因为只有四种可能的颜色（白、黑、褐和其他所有颜色），而我们知道四种选择可以用两个二进制位来定义，所以只需要两个开关来表示颜色。例如，两个开关都断开表示白色，一个闭合表示黑色，另一个闭合表示褐色，两个开关都闭合就表示其他所有颜色。

现在，让我们做一个控制面板来选择一只猫。控制面板上有四个开关（正如你家里的电灯开关）和一个灯泡：



控制面板

开关打到上面是指开关闭合，反之是指开关断开。也许表示猫的颜色的两个开关标识得不是很清楚，这是为了把控制面板做得更简练不得已而造成的。在表示颜色的一对开关中，左边的开关标着B，如果只有它往上就表示黑色；右边的开关标着T，如果只有它往上就表示黄褐色；B、T两个开关均往上则表示其他颜色，由O标识；B、T两个开关均往下则表示白色，由W标识。

在计算机专业术语中，开关是一种输入设备，输入是控制电路如何工作的信息。本例中输入开关对应于描述一只猫咪的4位信息，输出设备是灯泡。如果开关描述了一只符合条件的猫，灯泡就会亮。上面介绍的控制面板上的开关被设置成表示一只无生育能力的黑母猫，这是符合你的要求的，所以灯泡亮了。

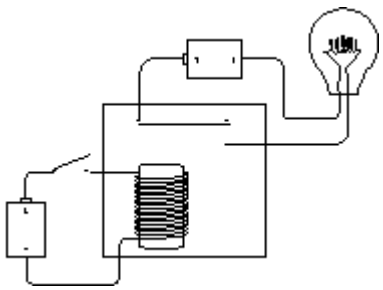
现在所要做的是设计一个使控制面板工作的电路。

前面提到过香农的论文题目是《A Symbolic Analysis of Relay and Switching Circuits》，他所指的relay和第6章中所讲的电报系统的继电器很类似。在香农的论文发表时，继电器已被用作其他目的，尤其是用于电话系统的大型网络。

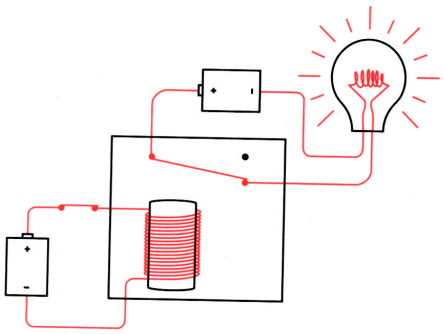
像开关一样，继电器也可以串联或并联以执行逻辑中的简单任务。继电器的组合称为逻辑门。这里所说的“逻辑门执行简单逻辑任务”是指逻辑门只完成最基本的功能。继电器比开关好是因为继电器可以被其他继电器控制而不必用手指控制，这意味着逻辑门可以被组合起来以执行更复杂的任务，比如一些简单的算术操作。事实上，下一章就要展示如何用电线连接开关、灯泡、电池和继电器来构造一个加法机（尽管它只能工作于二进制数字状态）。

继电器对电报系统的工作十分重要。连接电报站的电线长距离时电阻很大，需要一种方法来接收微弱的信号并把它增强后发送出去。继电器通过使用电磁铁控制开关可做到这一点。事实上，继电器放大了一个很弱的信号使其成为一个强信号。

就我们的目的而言，我们并不对它的信号放大能力感兴趣，真正使我们着迷的是继电器作为开关可用电来控制而不用手指。可以用电线把继电器、开关、灯泡和一对电池做如下连接：



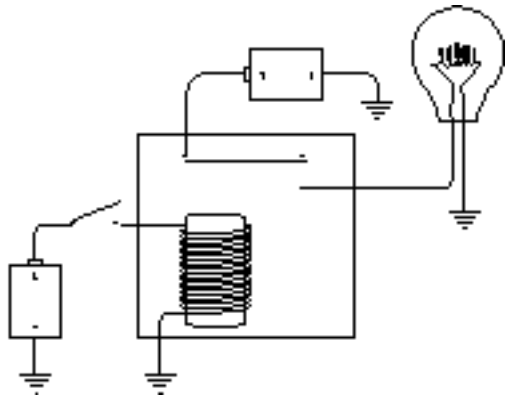
注意左边的开关是断开的，灯泡不亮。当闭合开关时，电流流过围绕在铁棒上的线圈，于是铁棒具有了磁性，并把上面有弹性的金属簧片拉下来，从而连通了电路，使灯泡发光：



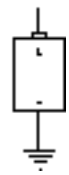
当电磁铁把上面的金属簧片拉下来时，这个继电器被称为“触发了”。当左边的开关断开时，铁棒不再有磁性，继电器中的金属簧片则弹回到原来的位置。

这看上去似乎是用一种很不直接的方式点亮灯泡的，但实际上这种方式是很直接的。如果我们只对点亮灯泡感兴趣，我们完全可以舍弃继电器。但我们的兴趣并非只是点亮灯泡这么简单，我们有更宏伟的目标。

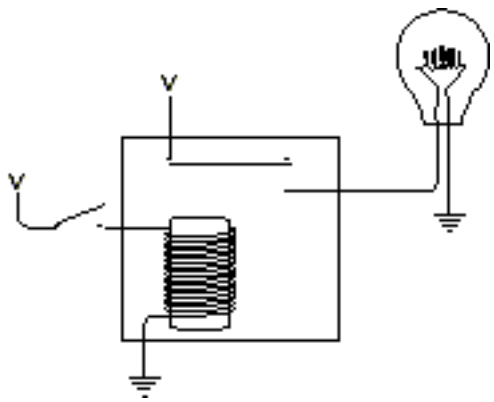
本章要多次用到继电器（当逻辑门建好之后就会很少再用了），所以要把上面那幅图简化一下。可以通过大地省去一些导线。在这种情况下，大地仅代表了一个公共端，并不是指真正的物理接地：



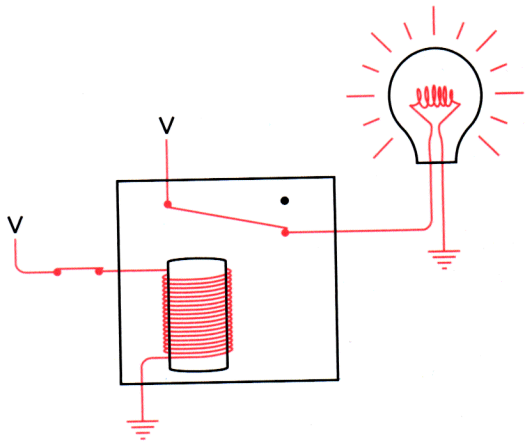
这看上去仍然不够简化，但还不至于那样做。注意两个电池的负极均接地，所以当看到的电池是这样的时：



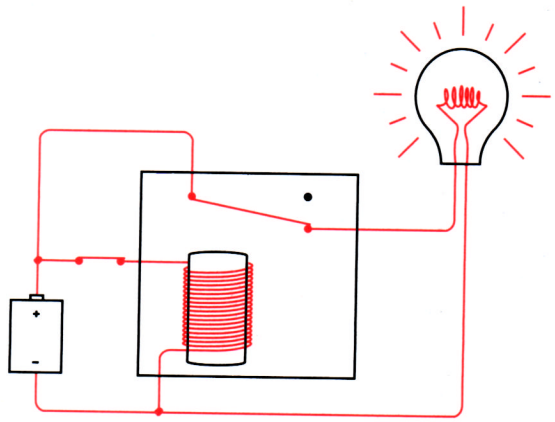
可用大写字母“V（它代表电压）”代替上图中的电池(如在第5和第6章中所做的)。现在继电器看上去如下图所示：



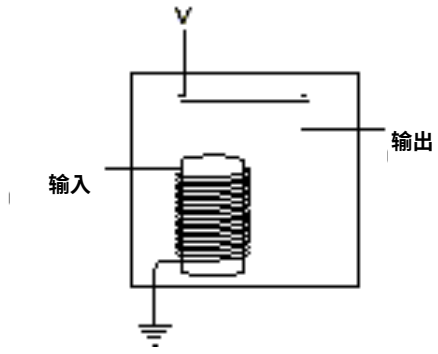
当右边开关闭合时，电流从V端流出，经过电磁铁芯流到地上。这使得电磁铁把上面有弹性的金属簧片拉下来，从而连通了接有灯泡的电路，灯泡点亮：



上面图显示了两个电源和两个接地，但本章的所有图中，电源，即“ V ”，可以互连，接地端也可以互连。本章及下一章的所有继电器和逻辑门的网络只要求有一个电池，但可能是一个大容量的电池。例如，上一幅图可只用一个电池，如下所示：



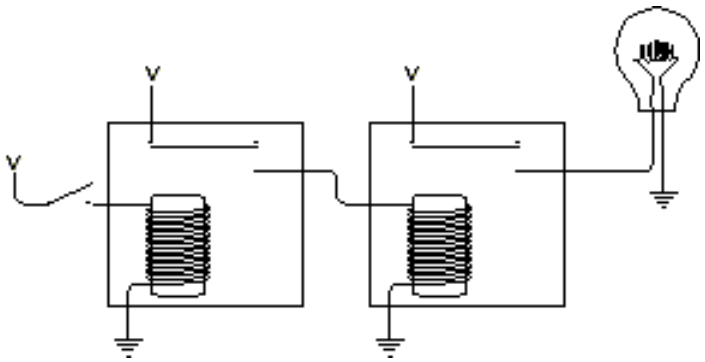
但这幅图并不能清楚地表明要用继电器做什么。先不考虑电路而把注意力放到输入和输出上，就像前面的控制面板一样：



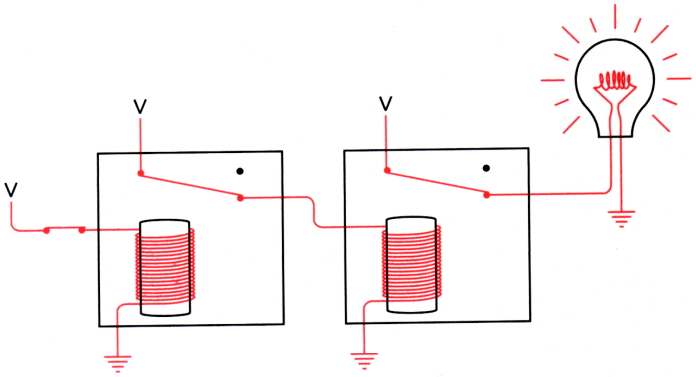
如果电流流经输入（例如，用一个开关把输入连到“ V ”端），电磁铁就会被触发，输出就有了一个电压。

继电器的输入不一定只能是开关，其输出也未必只限于灯泡。一个继电器的输出可以连

到另一个继电器的输入，如下所示：

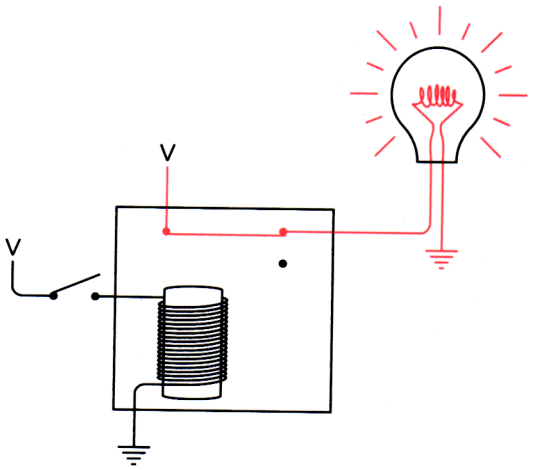


当闭合开关时，第一个继电器被触发，它为第二个继电器提供了输入电压，于是第二个继电器也被触发，灯泡被点亮了：

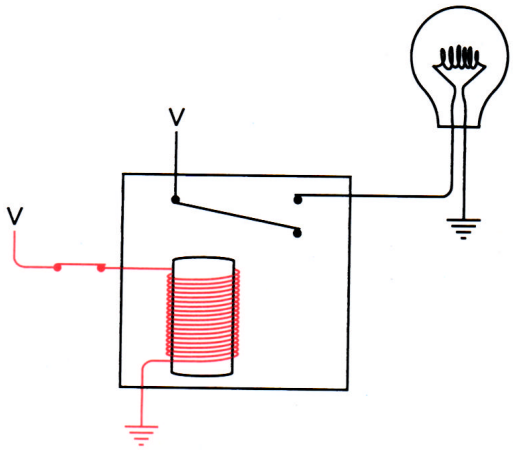


把继电器连接起来是构造逻辑门的关键。

事实上，灯泡可以两种方式连到继电器上。注意，具有弹性的金属簧片是被电磁铁拉下来的。平时，金属簧片与上端接触，当电磁铁吸引它的时候，它便和下端接触。我们一直把金属簧片与下端的接触作为继电器的输出，但我们也可以把它与上端的接触作为输出。当使用这种输出时，结果正好相反，输入开关断开时灯泡是亮的：



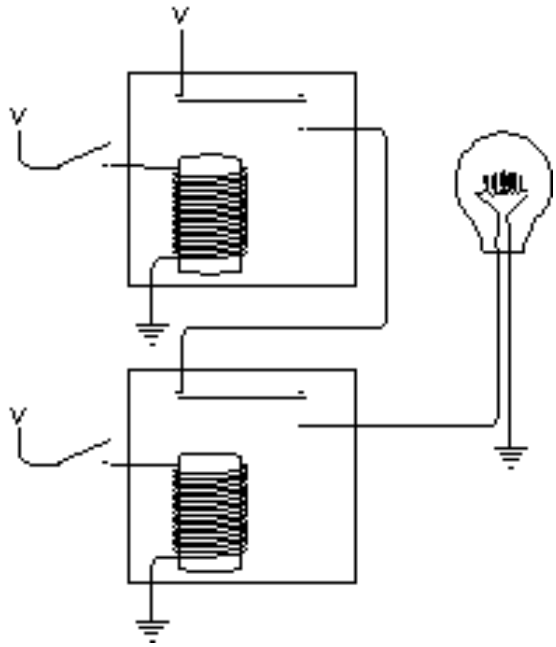
而当输入开关闭合时，灯泡便灭了：



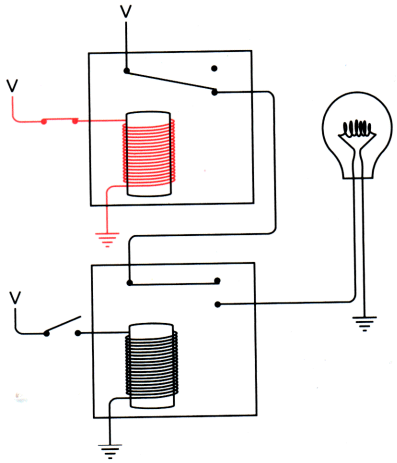
使用这种开关的继电器称为双掷继电器，它的两个输出在电性上是相反的——当一个有电压时，另一个则没有。

顺便说一下，如果你不知道现在的继电器是什么样子，你可以很方便地从当地的电器行的透明小包里看到一些。有些继电器就像（加入饮料的）方形小冰块一样大小，如元件号为275-206和275-214的继电器就是这种大小的且经久耐用的继电器。它们被封在一个干净的塑料外壳里，所以你可以看到电磁铁和弹性金属簧片。本章和下一章所描述的电路都使用的是元件号为275-240的继电器，它体积小且价格便宜（每个$ 2.99）。

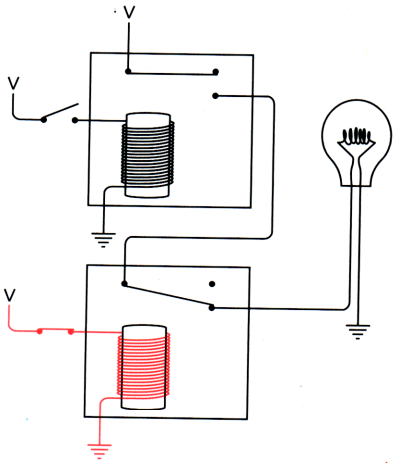
正如两个开关可被串联一样，两个继电器也可以串联：



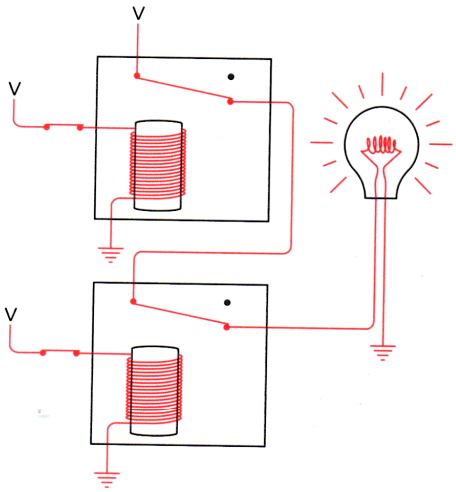
上面继电器的输出为下面的继电器提供了输入电压。如上所示，当两个开关均断开时，灯泡不会发光。试着闭合上面的开关：



由于下面的开关仍旧断开，下面的继电器没有触发，所以灯泡仍然不亮。若断开上面的开关而闭合下面的开关：

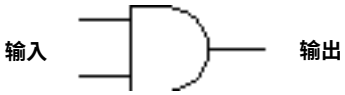


灯泡仍旧不亮。因为上面的继电器未被触发，电流无法流经灯泡。点亮灯泡的唯一方法是闭合两个开关：



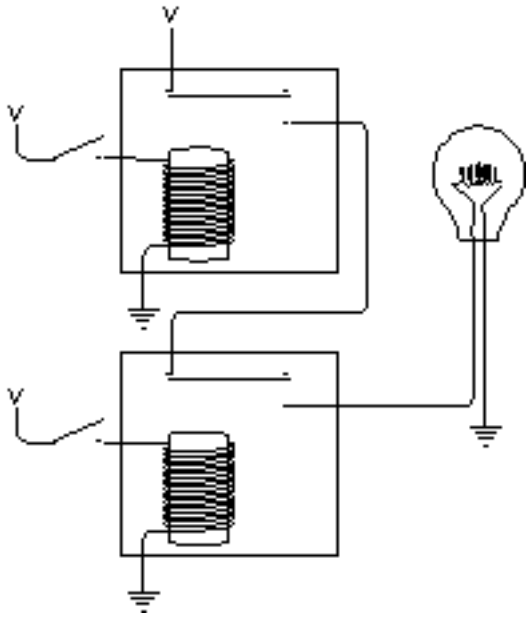
现在，两个继电器都被触发了，电流可以在电源、灯泡和接地点之间流通。

同串联开关一样，这两个继电器也执行了逻辑操作。只有当两个继电器都被触发时，灯泡才会点亮。串联的两个继电器就是一个“ AND gate（与门）”。为避免复杂的图示，电气工程师使用一个特殊的符号表示“ 与门 ”，如下图示：

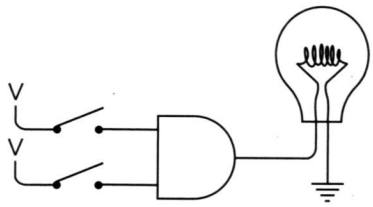


这是四个基本逻辑门中的第一个。与门有两个输入端（在图的左部），一个输出端（在图的右部）。这样表示的与门通常输入在左部，输出在右部。这是因为人们习惯于从左到右读图。但是与门同样可以画成输入在上部、右部或底下。

有两个继电器、两个开关和一个灯泡的原始电路图如下所示：

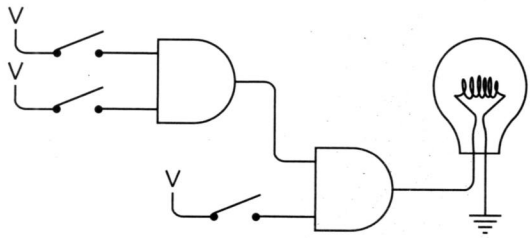


使用“ 与门 ”符号，上图可同样表示成：



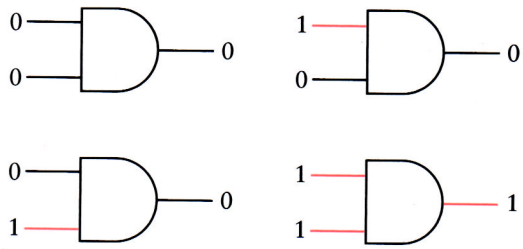
注意与门不仅代替了串联的两个继电器，同时也隐含了上面的继电器连着电源，且两个继电器都是接地的。只有当上下两个开关都闭合时，灯泡才会发光，这就是它之所以叫与门的原因。

与门的输入未必一定要和开关连接，且输出也不一定只能是灯泡。我们真正要处理的是输入端的电压和输出端的电压。例如，一个与门的输出可以是另一个与门的输入：



只有当三个开关都闭合时，灯泡才会发光。当上面的两个开关闭合时，第一个与门的输出会触发第二个与门的第一个继电器，而最下面的开关会触发第二个与门的第二个继电器。

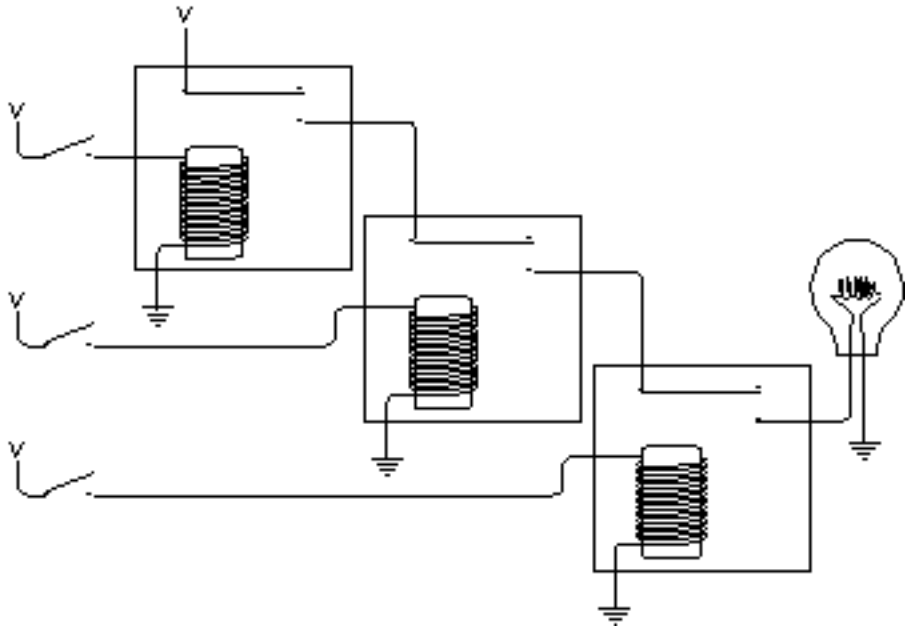
如果把不加电压视为0，加上电压视为1，则与门的输出按如下方式由输入来决定：



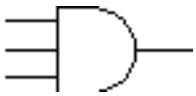
正如两个串联的开关一样，与门的输入输出关系可作如下描述：

AND	0	1
0	0	0
1	0	1

与门也可以有多于两个的输入端。例如，假设串联了三个继电器：

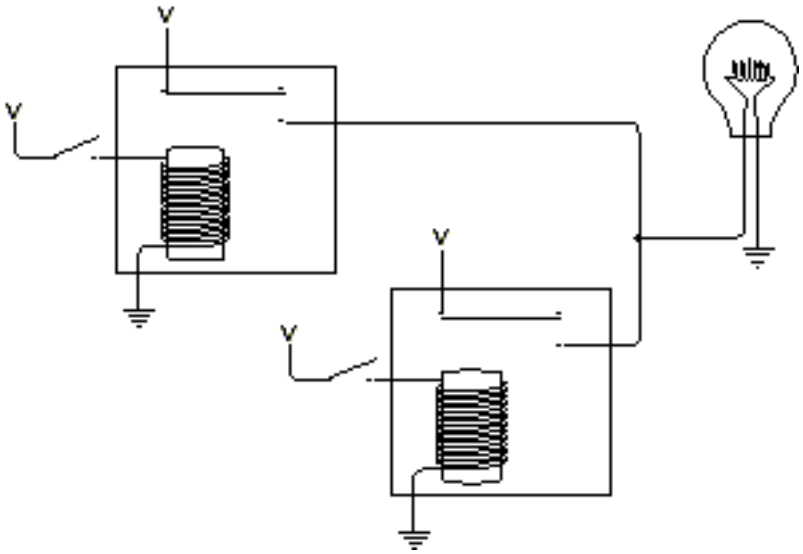


只有当三个开关同时闭合时，灯泡才会发光。这种配置可用如下符号表示：

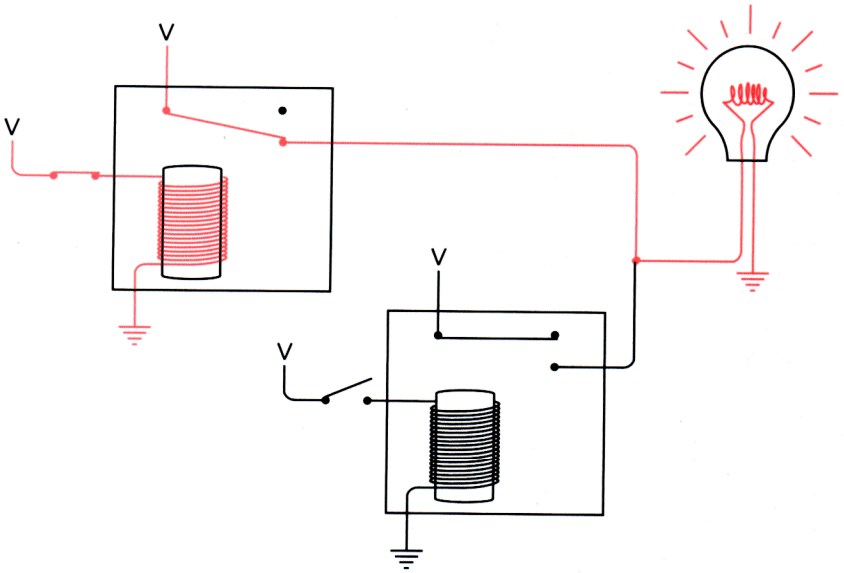


它被称为三输入端与门。

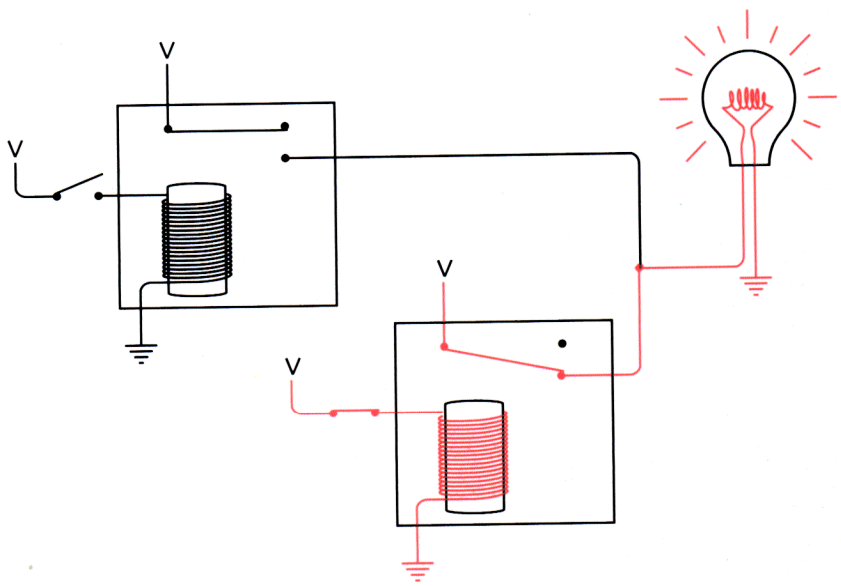
以下逻辑门可用并联的继电器解释：



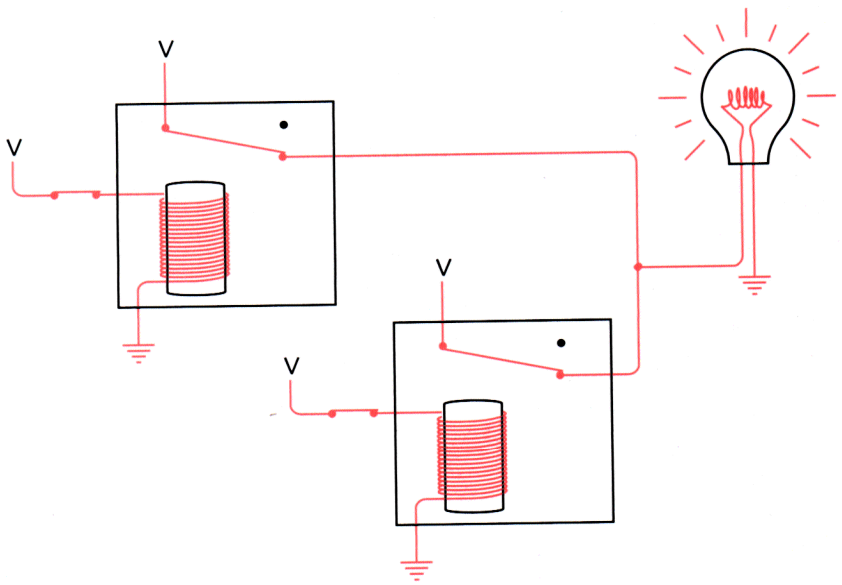
注意两个继电器的输出是连接在一起的，这个连接在一起的输出为灯泡提供了电源。任何一个继电器都可以点亮灯泡，例如，如果闭合上面的开关，灯泡会亮。这时，灯泡从左上角的继电器得到了电力供应。



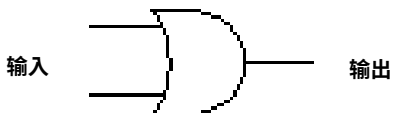
如果让上面的开关断开而闭合下面的开关，灯泡也会亮：



当两个开关都闭合时，灯泡同样会亮：

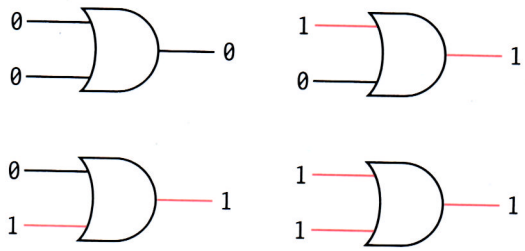


可见，当上开关或下开关中的任何一个闭合时，灯泡都会亮。这里的关键是“或”，所以这样的门叫“OR gate（或门）”。电气工程师使用如下符号表示或门：



它看上去和与门很相似，只是接输入端的一边是弧形的，很像英语“OR”中的字母“O”。或门的两个输入中，只要有一个加上电压，输出就是高电位。同样，如果约定不加电压

是0，而加电压是1，则或门也有四种可能的组合状态：

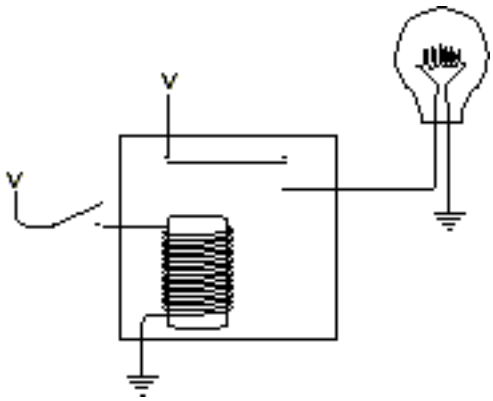


可以把或门的输入输出关系小结成如下表格：

OR	0	1
0	0	1
1	1	1

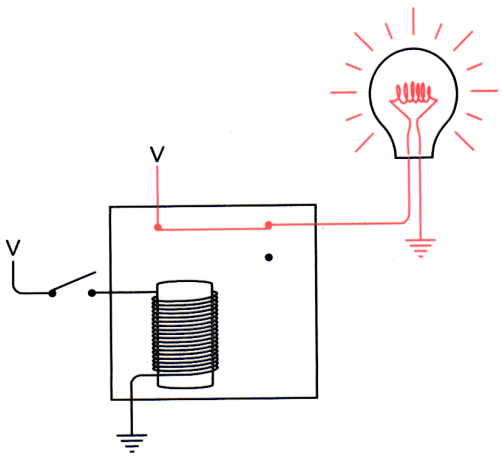
或门也可以有两个以上的输入端（当任一输入端为 1时，输出端就为 1；只有所有输入端均为0时，输出端才为0）。

前面解释过继电器可称为双掷继电器，因为其输出可以两种不同的方式连接。通常情况下，当开关断开时，灯泡不亮：

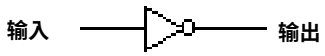


当开关闭合时，灯泡点亮。

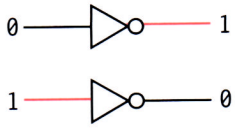
也可以用另外一种连接方式，使开关断开时灯泡点亮：



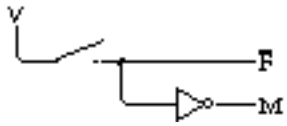
在这种情况下，只有闭合开关时灯泡才熄灭。以这种方式连接的继电器叫作反向器。反向器不是逻辑门（逻辑门通常有两个以上的输入），但它十分有用。反向器可以用下面的符号表示：



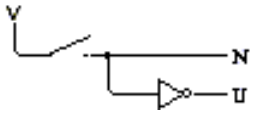
它被称为反向器的原因是当输入为0时输出却为1，反之亦然：



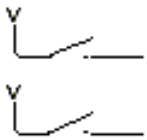
有了反向器、与门和或门，我们就可以制作控制板来自动选择理想的小猫了。让我们从开关开始。第一个开关的闭合表示母猫，断开表示公猫。这样，可以产生称为 F和M的两个信号，如下图所示：



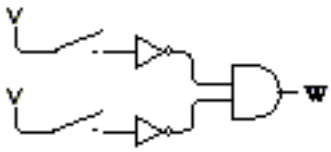
当F是1，M就是0，反之亦然。同样，第二个开关的闭合表示阉过的猫，而断开表示有生育能力的猫：



接下来的两个开关更复杂一些，不同的组合要代表四种不同的颜色。这里有两个开关，都与电源相连：

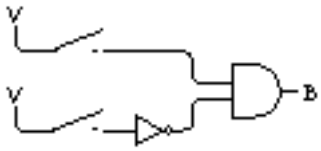


当两个开关都断开时，它们表示白色。我们用两个反向器和一个与门来产生信号 W。如果选择了一只白猫，W就为1，否则为0：

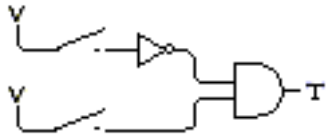


当开关断开时，两个反向器的输入是 0，这样反向器的输出（也就是与门的输入）为 1，这也就意味着与门的输出为1。一旦一个开关闭合，与门输出即为0。

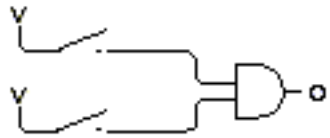
为表示一只黑猫，闭合第一个开关，这可以用一个反向器和一个与门实现：



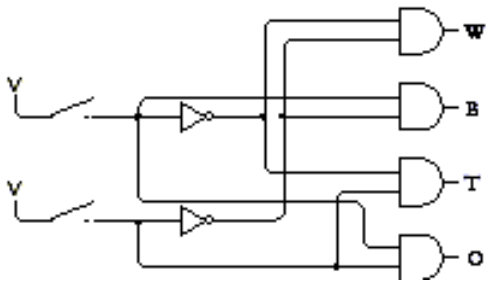
只有当第一个开关闭合而第二个开关断开时，与门的输出才是 1。同样，当第二个开关闭合而第一个开关断开时，与门的输出也为 1。我们用来表示褐色：



而如果两个开关都闭合时，用如下图示表示其他颜色：



现在把四个小电路集成为一个大电路（通常，黑点表示电线的连接点，没有黑点的交叉线是不连接的）：



这个连接图看起来十分复杂。但如果仔细地沿着线路走，看清楚每个与门的输入而不要关心这些输入又连到了别的什么地方，你就会明白电路是如何工作的。如果两个开关都断开，信号W会是1，其余信号都是0。如果第一个开关闭合，则信号 B会是1，其余信号都是0。

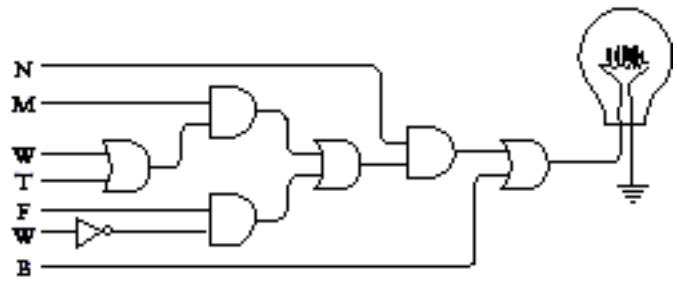
连接门和反向器时可以遵循一些简单的规则：一个门（或反向器）的输出可以作为其他门（或反向器）的输入，但是两个以上的门（或反向器）的输出永远不能互连在一起。

由4个与门和2个反向器组成的电路叫作“ 2-4译码器 ”。输入是两个二进制位的不同组合，共代表了4个不同的值。输出是 4个信号，任何时刻只能有一个是 1，至于哪一个是 1取决于两个输入位。用同样的原理还可以构造“ 3-8译码器 ”或“ 4-16译码器 ”等等。

选择小猫的表达式的简化表示是：

（N×（(M×（W+T)）+（F×（1-W））））+B

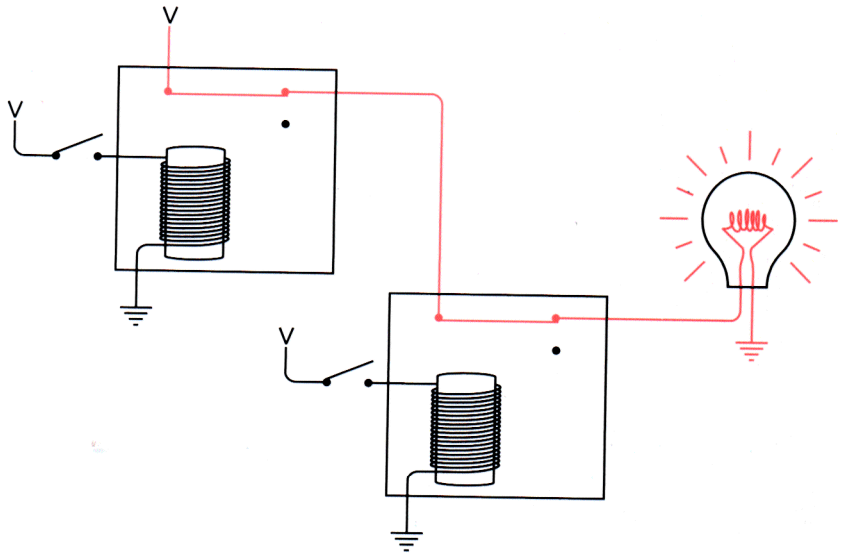
对于表达式中的每一个加号 (+)，必定对应电路中的一个或门。对于每一个乘号（×），则对应一个与门：



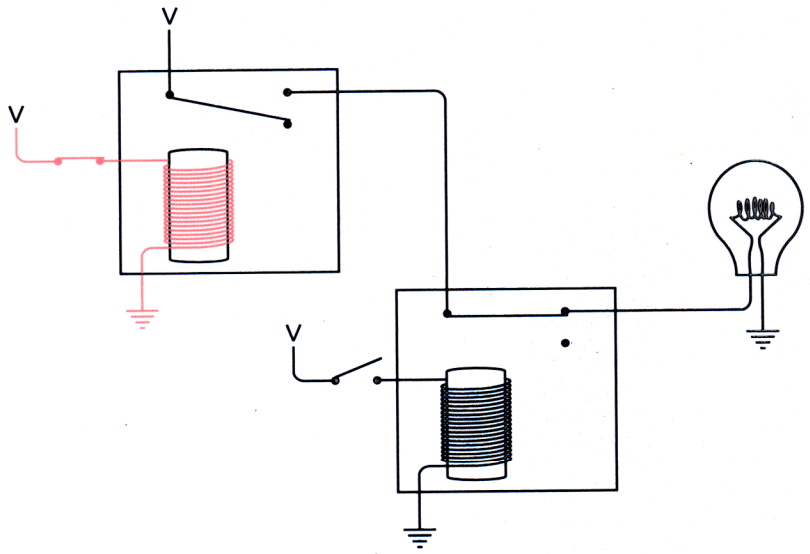
电路图左边的符号和它们在表达式中出现的顺序是一样的。这些信号来自于和反向器连接的开关及2-4译码器的输出。注意，图中用了反向器来表示表达式中的（1－W)。

你可能会说：“这不过是一堆继电器而已。”不错，这正是一堆继电器，每个与门和或门中都有两个继电器，一个反向器中有一个继电器，因而只能说你必须习惯它。以后的各章会用更多的继电器。不过，所幸的是你不用真正地去买一堆回家连起来。

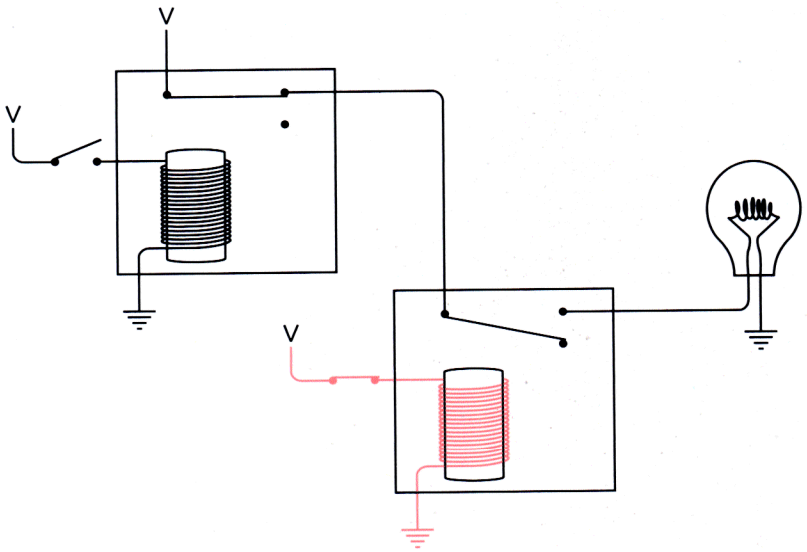
本章再看两个逻辑门。这两个门都会用到这样一个继电器，该继电器在不被触发时，其输出为高电位（这是用在反向器中的输出)。例如，下面配置中，一个继电器的输出为第二个继电器提供了电源。当两个输入都断开时，灯泡是点亮的：



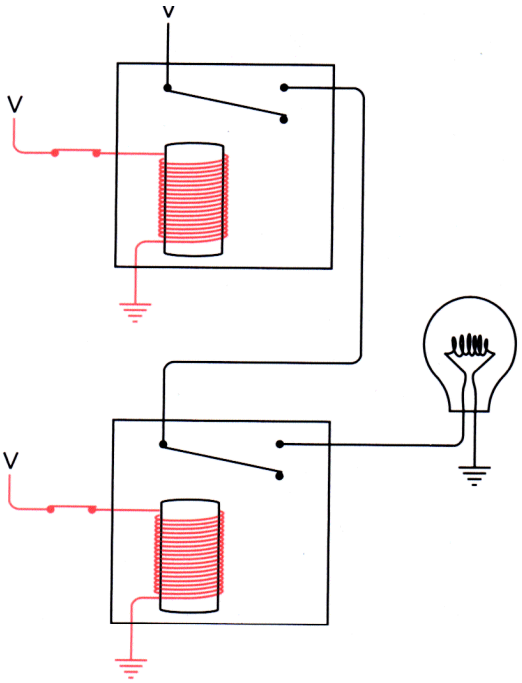
如果上面的开关闭合了，灯泡就会熄灭：



灯泡的熄灭是因为第二个继电器没有电源供应。同样，若下面的开关闭合灯泡也会熄灭：



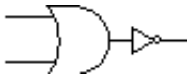
若两个开关都闭合，灯泡还是不亮：



这种行为和或门的行为正好相反，被称为“ NOR gate（或非门）”。下面是或非门的符号：

它和或门的符号很相像，只是在输出端有一个空心的小圆圈，这个小圆圈表示反向，故

而或非门也可用下面的表示：

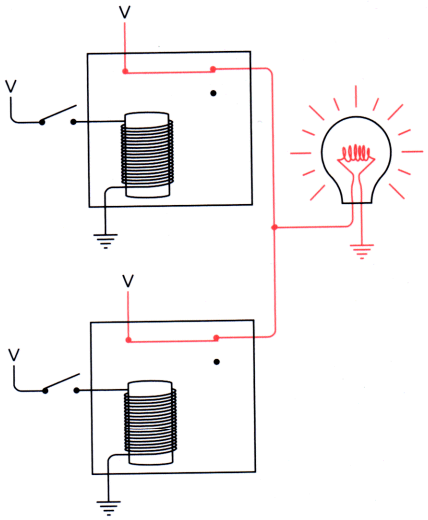


或非门的输出如下表所示：

NOR	0	1
0	1	0
1	0	0

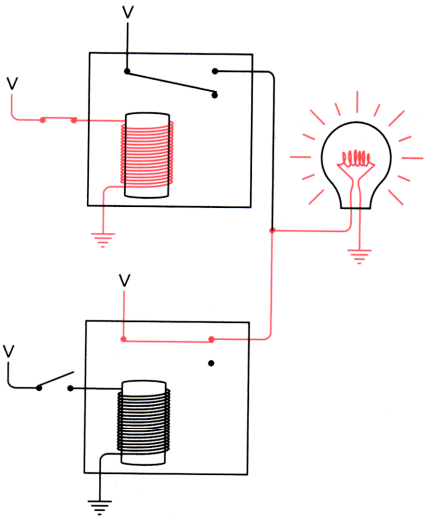
这张表显示的结果和或门相反。在或门中，输入端中只要有一个是 1，输出就是1；只有输入端均为0时，输出才为0。

连接两个继电器的另一种方式如下图所示：

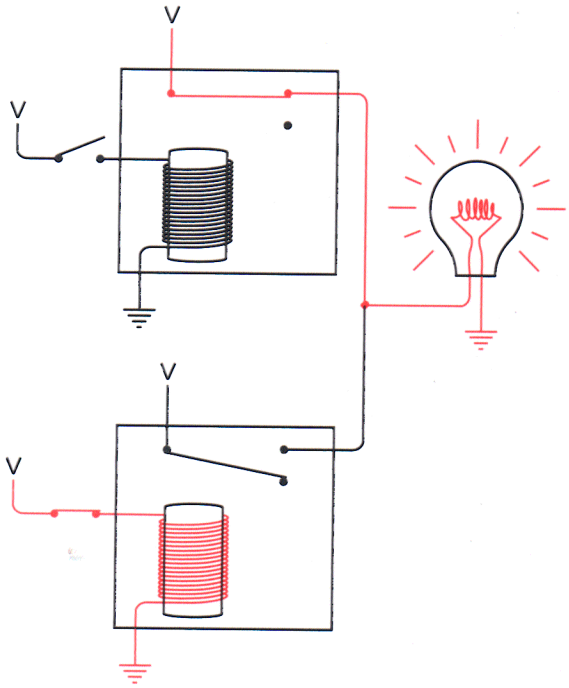


在这种情况下，两个输出连在一起。除了连在继电器的另一个触点上之外，这种连接形式与或门类似。当两个开关都断开时灯泡是亮的。

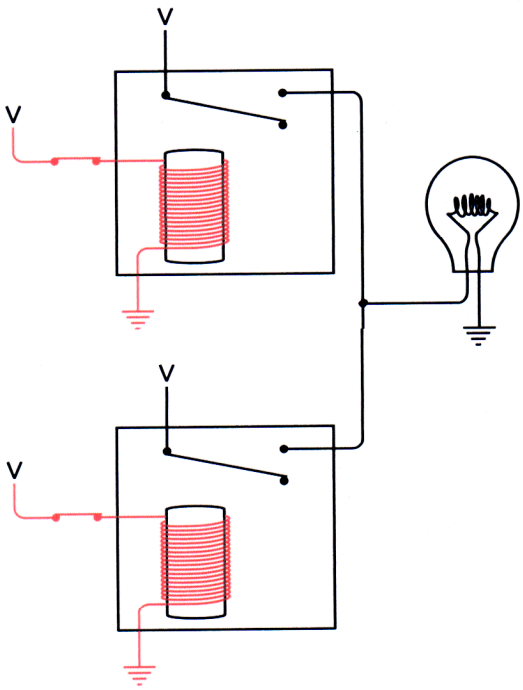
当只有上面的开关闭合时，灯泡也是亮的：



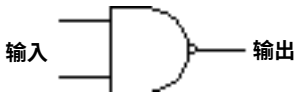
当只有下面的开关闭合时，灯泡也是亮的：



只有当两个开关都闭合时，灯泡才会熄灭：



这种行为和与门的行为正好相反，被称为“ NAND gate（与非门）”。与非门的画法和与门的画法很相像，只是在输出端加了一个小圆圈，表示其最后的输出和与门的输出是相反的：



与非门的输出如下表所示：

NAND	0	1
0	1	1
1	1	0

注意，与非门的输出与与门恰恰相反。对与门而言，当两个输入都为 1时，输出才为1；否则输出就是0。

到此为止，我们已经看到可用四种不同的方式来连接有两个输入、一个输出的继电器，每一种方式的行为功能都不一样。为避免画继电器，我们把这些连接称为逻辑门并使用电气工程师们使用的符号来表示它们。特定的逻辑门的输出取决于其输入，总结如下：

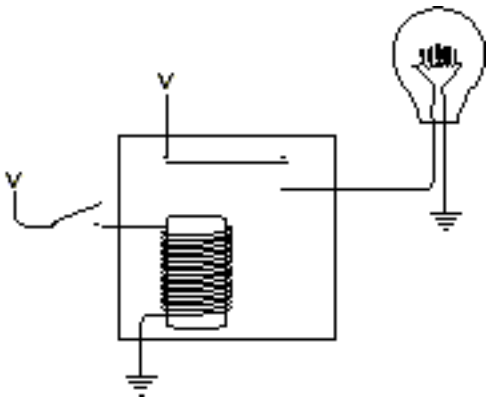
AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

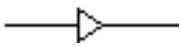
NAND	0	1
0	1	1
1	1	0

NOR	0	1
0	1	0
1	0	0

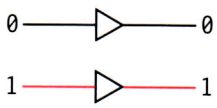
现在已有了四个逻辑门和一个反向器，完成这些工具的其实就是原始的继电器：



上图称为缓冲器，用符号表示如下：



它和反向器的符号类似，只是没有小圆圈。缓冲器的特点是“什么都不做”，其输出和输入是相同的：

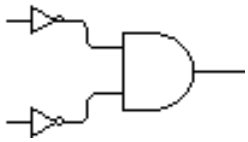


当输入信号很弱时，可以使用缓冲器，这是因为这也正是多年前继电器被用于电报当中

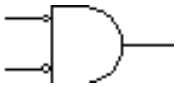
的原因。此外，缓冲器也可用于延迟一个信号，这是因为继电器可能要求多一点儿动作时间，如1秒的几分之一才被触发。

本书从现在开始不再画继电器，取而代之的是电路将由缓冲器、反向器、4个基本逻辑门及更复杂的电路（如2-4译码器）组成。当然，所有这些部件也是由继电器构成的，但我们用不着看到它了。

前面讲过，可用下面的小电路构造一个2-4译码器：

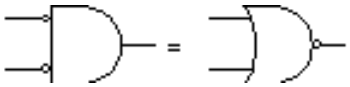


两个输入被反向后成为与门的输入。有时，像这样的配置可以去掉反向器而画成如下的样子：



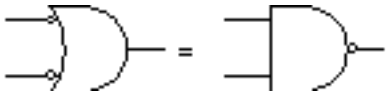
注意与门输入端的小圆圈，这些小圆圈表示信号在这些点上被反向了，0会变成1，而1变为0。

具有反向输入端的与门和或非门的行为是一样的：



只有两个输入端都为0时输出才为1。

同样，具有反向输入端的或门和与非门的行为是一样的：



只有输入端均为1时输出才为0。

这两对等同的电路实际上就是迪摩根定律的内容。迪摩根是维多利亚时代的另一位数学家，他比布尔年长9岁。据说，他的书《Formal logic》发表于1847年，和布尔的《The Mathematical Analysis of logic》恰好是同一天。事实上，布尔正是由于受到发生在迪摩根和另一个英国数学家之间的剽窃事件的触动而研究逻辑的。（迪摩根最后证明是清白的。）很早以前，迪摩根就意识到了布尔思想的重要性。他无私地鼓励和帮助布尔进行研究，但最终除了他的这个著名的定律外，他几乎被人们遗忘了。

迪摩根定律可以简单地表示成：

$$\overline{A} \times \overline{B} = \overline{A + B}$$
$$\overline{A} + \overline{B} = \overline{A \times B}$$

A和B是两个布尔操作数。在第一个表达式中，它们被取反（即反向）后再相与。这和先把它们相或后再取反（或非门的功能）的结果是一致的。第二个表达式中，两个操作数被取反后再相或，这和先把它们相与后再取反（与非门的功能）的结果是一样的。

迪摩根定律对于简化布尔表达式，进而简化电路是一个很重要的工具。从历史上讲，这正是香农的论文对电气工程师的真正含义。但是，专门简化电路并非本书的焦点，更重要的是让事物工作、起作用。下面我们要运行起来的就是一台简单的加法机。

第12章　二进制加法机

加法是最基本的算术运算。所以，如果想要建造一台计算机（这是本书隐含讨论的问题），必须首先知道如何构造一种机器，它可以把两个数加起来。当你解决了这个问题，你会发现加法正是计算机唯一所做的事情，因为通过使用用于加法的机器，我们还可以构造用加法来实现减法、乘法、除法以及计算房产抵押款、引导向火星发射卫星、下棋和电话计费等等功能的机器。

同现代的计算器和计算机比起来，本章构造的加法机庞大、笨重、速度慢且噪声大。但有意思的是构成它的部件完全是前几章学过的电子设备，如开关、灯泡、电线、电池以及可构成几种逻辑门的继电器。这个加法机包含的所有部件都于 120年以前就已发明，而且，我们并不用真正地在屋子里建造它，只需在纸上和脑子里构造这台机器就行了。

这个加法机只能工作于二进制数，而且它缺少很多现代计算机（器）的辅助设备。它不能用键盘来敲入你想加的数，代之的你只能用一系列开关表示待加的数。它也不能用显示器显示结果，你所看到的只是一排灯泡。

但这台加法机确实实现了两数相加的功能，而且其工作方式和计算机做加法十分相似。

二进制加法与十进制加法很像。当你相加十进制数如 245和673时，你把问题分解成简单的步骤，每一步只对一对十进制数字相加。本例中，第 1步是把5和3加起来。生活中，你若能记住加法表，问题的解决就快多了。

十进制加法和二进制加法的一大区别是二进制数字的加法表要比十进制数字的加法表简单得多：

+	0	1
0	0	1
1	1	10

你可能在学校里记过上面这张表，并背诵过如下口诀：

0加0等于0，
0加1等于1，
1加0等于1，
1加1等于0，进1。

把相加结果的数前加上零，可以把加法表改写成如下形式：

+	0	1
0	00	01
1	01	10

这样一来，二进制数字相加的结果是两位数，分别称为“和”和“进位”（比如“1加1等于0，进位是1”）。现在，可以把这张二进制加法表分成两张表，第 1张是表示“和”的表：

+和	0	1
0	0	1
1	1	0

第2张是表示“进位”的表：

+进位	0	1
0	0	0
1	0	1

以这种方式来看待二进制加法就很方便了，因为加法机会分开求和与进位。构造二进制加法机需要设计一个能执行表中所描述操作的电路。因为电路的所有部件，如开关、灯泡、电线都是可以表示成二进制数的，因而该电路由于仅工作于二进制数从而大大降低了电路的复杂性。

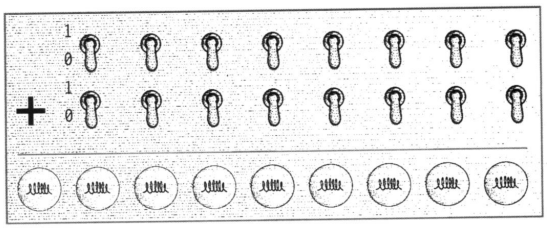
与十进制加法一样，二进制加法也从最右边的一列开始，逐列相加两个数：

```
  01100101
+ 10110110
----------
 100011011
```

注意，当从右边加到第3列的时候，产生了一个进位。同样的情况也发生在第6、7、8列。

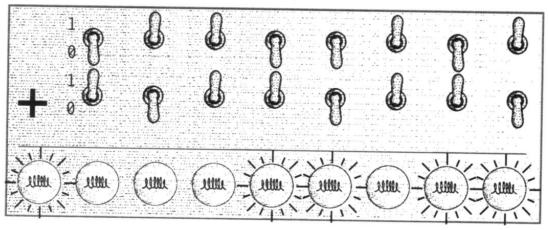
我们要加多大的数呢？由于这个加法机只是在脑子里构造，因而可以加很长的数字。为更合理一些，选择不超过8位的二进制数。也就是说，操作数的范围是从0000-0000～1111-1111，即十进制的0～255。两个8位二进制数的和最大可以是1-1111-1110，即510。

此二进制加法机的控制面板如下图所示：



板上有两行开关，每行8个。这些开关集是输入设备，我们将用它输入两个8位数。开关往下表示0，往上表示1，正如家里墙上的开关。输出设备在板的底部，是一行灯泡，共9个。这些灯泡用来表示加法的结果，不亮的灯泡表示0，亮的表示1。我们用了9个灯泡是因为两个8位数相加的结果可能是9位数。

加法机的余下部分包含了以不同方式连接而成的逻辑门。开关触发逻辑门中的继电器，继电器接着点亮相应的灯泡。例如，如果我们想把0110-0101和1011-0110加起来（即前例中显示的两个数字），需把相应的开关设置成下面的样子：



灯泡的亮暗表明答案是1-0001-1011。（当然，这只是希望的情况。毕竟，我们还没有把这个加法机构造出来！）

上一章提到过本书将会用到很多继电器，本章中的 8位加法机就至少需要 144个继电器，其中每一对数进行加法操作需要18个继电器（8 × 18=144）。如果画出完整的电路图，你一定会大惊失色，任何人都无法将连成一堆的 144个继电器看得明明白白，所以我们将用逻辑门分步解决这个问题。

当你看到下面两个1位二进制数相加的进位表时，你可能立刻会想到逻辑门和二进制加法之间有某种联系：

+进位	0	1
0	0	0
1	0	1

你也许已意识到这和上章所述的与门的输出是一样的：

AND	0	1
0	0	0
1	0	1

所以，与门可以用来计算两个1位进制数位相加得到的进位。

看来我们已取得一点儿进展了，下一步就要看看有没有继电器能完成下面的工作：

+和	0	1
0	0	1
1	1	0

这是二进制加法运算中的另一半问题，虽说表示和的这一位不如进位那么容易实现，但我们会有办法。

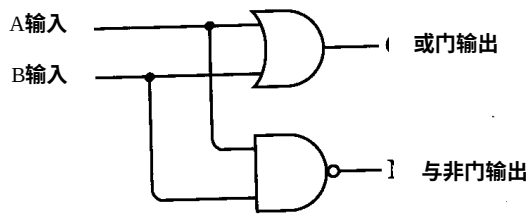
首先应意识到或门的输出和我们所期望的很近似，只是右下角的结果不同：

OR	0	1
0	0	1
1	1	1

而对于与非门而言，除了左上角的输出不同以外，其他结果也与期望的一样：

NAND	0	1
0	1	1
1	1	0

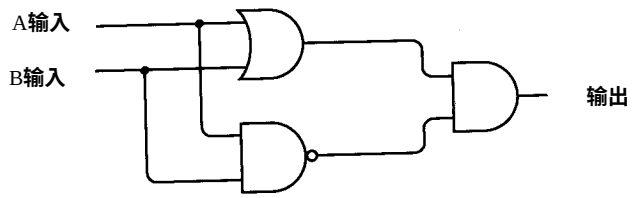
所以，使用相同的输入，让我们把与非门和或门连接起来：



下表总结了或门和与非门的输出，并将其结果和加法机所要求的结果进行比较：

A\输入	B输入	或门输出	与非门输出	需要的结果
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

注意，当或门和与非门的输出都为 1时，就可以得到期望的结果 1，这暗示着把两个输出作为与门的输入：

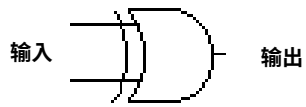


好，这样就能满足要求了。

整个电路仍然只有两个输入，一个输出。两个输入既连到了或门，也连到了与非门。或门和与非门的输出作为与门的输入，从而得到预期的结果：

A输入	B输入	或门输出	与非门输出	与输出
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

这个电路有它自己的名字，称为“异或门（Exclusive OR gate 或 XOR）”。异或门输出为1时，A输入为1或B输入为1，但不能同时为1。不用再去画一个或门、一个与非门和一个与门，可以用电气工程师规定的符号来表示它：



它看上去和或门很像，只是在输入端还有一条曲线。异或门的行为表示如下：

XOR	0	1
0	0	1
1	1	0

异或门是本书需要详细描述的最后一个逻辑门（在电气工程中有时还会遇到第六个门，称为“同或门”，同或门只有两个输入相等时输出才为 1。同或门描述的输出情况正好和异或门相反，所以这个门的符号和异或门相同，同时在输出端有一个小圆圈）。

让我们来总结一下。两个二进制数相加产生两个表，一个是表示“和”的表，另一个是表示“进位”的表：

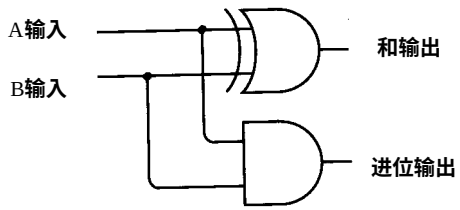
+和	0	1
0	0	1
1	1	0

+进位	0	1
0	0	0
1	0	1

用下面两个逻辑门可以得到同样的结果：

XOR	0	1		AND	0	1
0	0	1		0	0	0
1	1	0		1	0	1

二进制数的“和”可以由异或门得到，而“进位”可以由与门得到，所以可以把异或门和与门结合起来来完成两个二进制数A和B的加法：



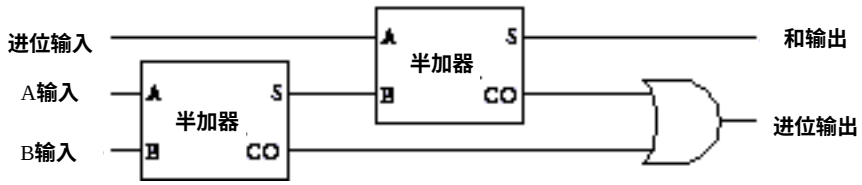
不用画与门和异或门，可以把上图简单地表示成如下的样子：

其中的方块称为“半加器（Half Adder）”，它可以把两个二进制位A和B相加，从而得到一个和输出(简称S)和一个进位输出(简称CO)。但大部分二进制数是多于1位的，半加器不能够把前一步的进位加到本次运算中。例如做如下加法：

$$\begin{array}{r} 1111 \\ +\ 1111 \\ \hline 11110 \end{array}$$

只能用半加器来计算最右边一列数：即1加1等于0，进位为1。对于右边第2列数，由于进位的存在，需要加3个数。接下来的几列都有这个问题，每一列二进制位的加法都包括了来自前一列的进位。

要把3个二进制数相加，需要按如下方式把两个半加器和一个或门连接起来：



要理解它的工作原理，先从最左边第一个半加器的A输入和B输入开始，其输出是一个和及相应的进位。这个和必须和前一列的进位输入(简称CI)加起来，然后把它们输入到第二个半加器。第二个半加器的和输出是最后的和。两个半加器的进位输出又输入到一个或门，或门产生了本次加法的进位输出。你可能会想这里还需要一个半加器，这当然是可行的。但

当你把所有的可能情况考虑完，你会发现两个进位不可能同时为 1。当两个输入不能同时为 1 时，或门已足够用于表示两个进位的加法，此时或门和异或门的功能是相同的。

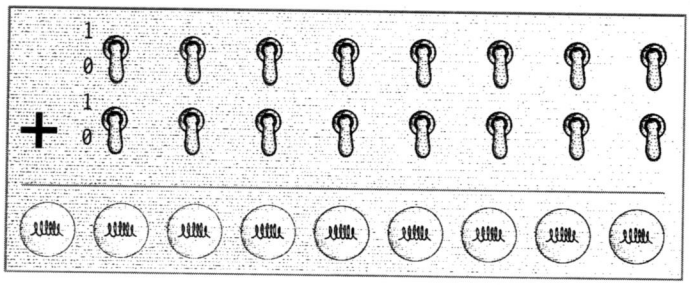
上图可简化表示为下面的方块图，称其为“全加器（Full Adder）”：

下面的表是对全加器所有可能的输入及其相应输出的小结：

A输入	B输入	进位输入	和输出	进位输出
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

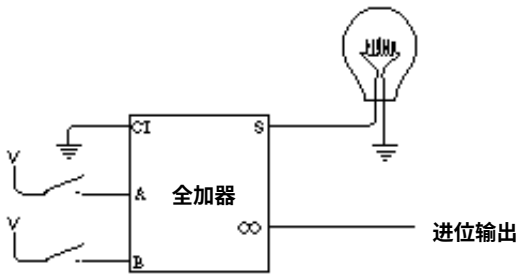
前面说过加法机需要144个继电器，这个数目是如何得到的呢？每个与门、或门、与非门都需要2个继电器，所以，一个异或门需 6个继电器。一个半加器由一个异或门和一个与门构成，所以它要8个继电器。 1个全加器需要两个半加器和一个或门，所以它要 18个继电器。对于8位二进制加法机而言，共需8个全加器，因而总共是144个继电器。

回想一下本章最开始那个带开关和灯泡的控制面板：



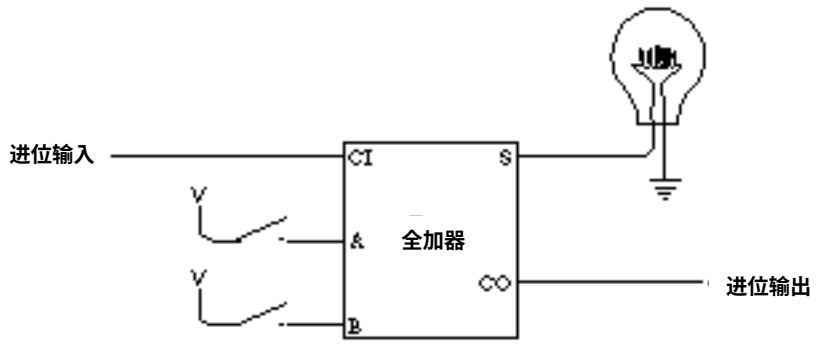
现在可以把这些开关和灯泡连接成全加器了。

首先把最右边的两个开关和一个灯泡连到一个全加器上，如下图所示：



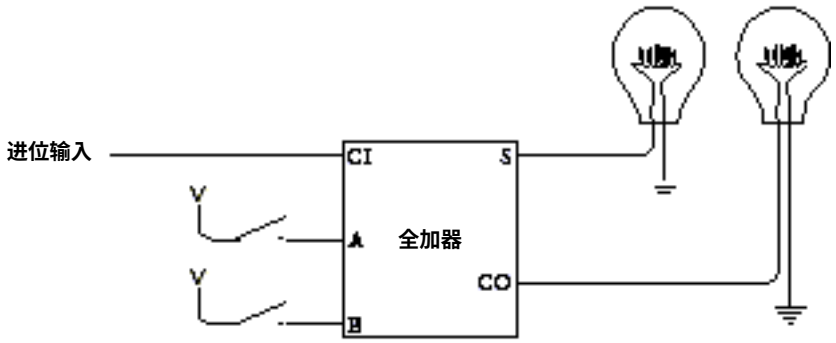
当把两个二进制数相加时，第1列的处理有所不同。因为接下去的几列可能包括来自前面加法的进位，而第1列不会有进位，所以全加器的进位输入端是接地的，这表示输入为“ 0”。第1列相加后很可能会产生一个进位输出，这个进位输出是下一列加法的输入。

对于接下去的两个二进制位和灯泡，可以按如下办法连接全加器：



第一个全加器的进位输出是第二个全加器的进位输入。接下去的每一列数都以这种方式连接，每一列的进位输出都是下一列的进位输入。

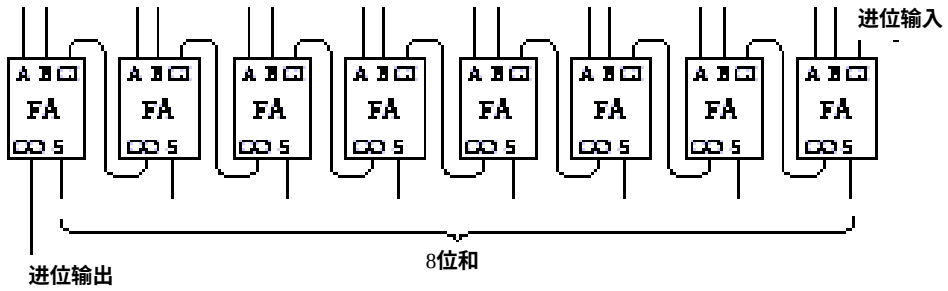
第八个灯泡和最后一对开关连到最后一个全加器上，连接方式如下图所示：



这里最后的进位输出连到第九个灯泡上。

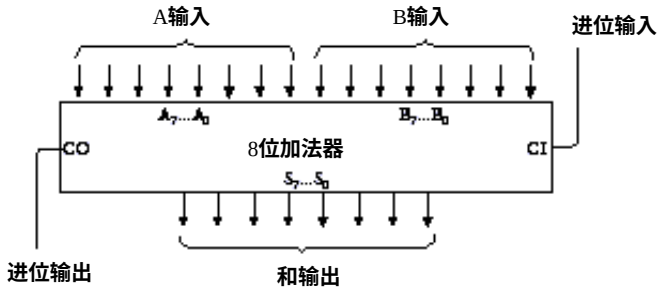
这样，8个全加器就构造成功了。

还可以用另一种方式来看8个全加器的集成，每个全加器的进位输出都是下一个全加器的进位输入：



下面是一个完整的屏蔽在一个盒子里的8位加法器。输入是A和B标识为从A_0～A_7及B_0～B_7。

输出为和输出，标识为从S_0～S_7：



这是标识多位数字的常用方法。下标为 0的位A_0、B_0和S_0表示最右边的、最不起眼的位。而位A_7、B_7和S_7是最左边的、最引人注目的位。例如，下面展示的是这些字母是如何用来表示二进制数0110-1001的：

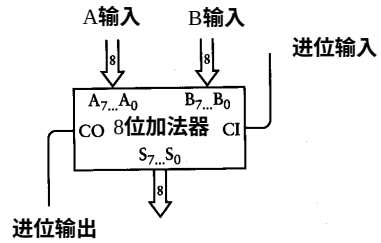
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	1	1	0	1	0	0	1

下标始于0，且向高位递增的原因是它们和 2的乘方数（幂）是对应的：

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	0	1	0	0	1

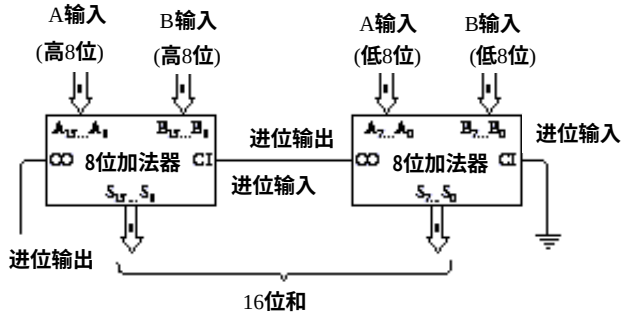
如果把每个二进制位和对应的 2的幂次方相乘再依次相加，你就会得到 0110-1001的十进制数表示，即64+32+8+1＝105。

8位加法器的另一种画法是：



双线箭头包含了8个输入端，代表一组8个分开的信号。它们标识为A_7…A_0、B_7…B_0、S_7…S_0也用来表示一个8位二进制数。

一旦构造了一个8位加法器，就可以构造另一个加法器。把它们级联起来可以很容易地构成16位加法器：



右边加法器的进位输出连到左边加法器的进位输入端。左边加法器的输入包含了两个加数的高8位，同时产生了结果的高8位。

现在，你可能会问：“计算机真的是以这种方式把数字加起来的吗？”

基本上是这样的，但不完全是。

首先，加法器应该做得更快。如果你明白这个电路是如何工作的，你会看到最低位相加产生的进位作为下一列数相加的一个输入，而第3列的加法又等着第2列加法的进位，依此类推。加法器总体的速度等于加数的位数乘以单个全加器的速度。这种进位方式称为行波进位。更快的加法器使用称为先行进位的加法电路，从而加快了加法进程。

第二（但是十分重要），计算机再也不用继电器了！尽管它们曾经用过。建于20世纪30年代初的第一批数字计算机使用继电器，后来又用了真空管。现代计算机用晶体管。当用在计算机中时，晶体管和继电器的功能差不多，但是晶体管速度更快，体积更小，更安静，更省电，而且还便宜不少。构造一个8位加法器仍然需要144个晶体管（如果采用先行进位，则需要更多），但整体电路的体积却小多了。

第13章　如何实现减法

在你确信继电器可以连接起来以构成二进制加法器后，你可能会问：“减法器如何实现呢？”本章将会为你解答这个问题，且提出这个问题也表明你有了一定的理解力。减法和加法在某些方面是互为补充的，但两种计算的机制不同。加法从最右边一列向最左边一列计算，每一列的进位都加到下一列中去。减法不用进位，相反，要用到借位——一种本质上与加法不同的机制。

例如，让我们看一道典型的不断借位的减法题目：

```
  253
- 176
-----
  ???
```

要做这道题，从最右边一列开始。首先，6比3大，所以需要从5借1，这样就变成了13减6，结果是7。由于从5借了1，5就变成了4，4比7小，所以继续从2借1，14减7等于7。2被借1后成为1，1减1为0，所以最后结果是77：

```
  253
-176
-----
   77
```

如何用逻辑门来实现这看似不合常理的逻辑呢？

我们不会直接用这种方法，代替的是用一个小技巧，使不通过借位来实现减法。这会是一个使大家都满意的好办法。详细地了解减法的完成是很有用的，因为它和用二进制编码在计算机中存储负数的机制有很大联系。

为解释这样的工作，需要清楚地指明两个操作数，即减数和被减数。减数从被减数中去掉后，结果是二者之差：

```
  被减数
-  减数
-------
    差
```

要想不借位，首先将减数从999中减去：

```
  999
-176
-----
  823
```

这里用999是因为操作数是3位，如果是4位数，就用9999。把一个数从一串9中减去得到的结果称为9的补数或补码。176的9的补数是823，反之，823的9的补数是176。这样做的好处在于，无论减数是什么，计算9的补数永远不需要借位。

在计算出减数的9的补数之后，把它加到原来的被减数上：

```
  253
+ 823
------
 1076
```

最后，你再加１并且减去1000：

```
  1076
+    1
-1000
------
   77
```

这样就得到结果了。答案和以前一样，且你根本不用借位。

这是什么原理呢？原来的减法题目是：

253－176

表达式加一个数再减同一个数得到的结果是一样的。所以先加上 1000，再减去1000：

253－176 + 1000－1000

这个式子等同于下面的式子：

253－176 + 999 + 1－1000

再按如下方式重新组合：

253+（999-176）+1－1000

这与前面描述过的用9的补数进行的计算是一致的。虽然用了两个减法和两个加法来代替一个减法，但是也因此省去了讨厌的借位。

但是，如果减数比被减数大怎么办呢？例如如下计算：

```
 176
-253
-----
 ???
```

通常情况下，你看到这个式子后可能会说：“减数比被减数大只需交换两数位置，再做减法，然后给结果取个相反数。”于是你在脑子里交换了它们的位置，并求出了答案：

```
 176
-253
-----
-77
```

要省去借位来做这道题和前面的例子有所不同。首先你要求出 253的9的补数，即

```
 999
-253
-----
 746
```

再把该补数和原来的被减数相加：

```
 176
+746
-----
 922
```

这时候，按照上一道题的步骤，你应该对其加1再减去1000，但在本题中，这种方法不会生效。如果你还按这种步骤做，就需要从923中减去1000，这又导致了借位。

既然实际上前面已经加了999，这里再减去999：

```
  922
-999
-----
  ???
```

当做到这一步时，可看出结果是个负数，故需要交换两数位置，不过这样再做减法时已不需要借位，答案如预期所料：

```
  922
-999
-----
 -77
```

同样的方法可用于二进制数减法，而且会比十进制数减法来得简单。让我们看看该如何做。

原来的减法题目是：

```
  253
-176
-----
  ???
```

当把这些数转化为二进制数时，问题变成：

```
  11111101
-10110000
---------
  ????????
```

步骤1 用11111111减去减数：

```
  11111101
- 10110000
----------
  01001111
```

当计算十进制数减法时，减数是从一串9中减去，得到称为9的补数的结果。对于二进制数减法，减数从一串1中减去，差称为1的补数。但请注意，求1的补数实际上并不需要做减法，因为1的补数中，原来的0变成1，原来的1变成0，所以，1的补数有时也称为相反数或反码。（你是否还记得第11章中反向器的作用是把0变成1，把1变成0。）

步骤2 把步骤1中求得的补数和被减数相加：

```
  11111101
+ 01001111
----------
 101001100
```

步骤3 对结果加1：

```
 101001100
+         1
----------
 101001101
```

步骤4 减去100000000（256）：

```
  101001101
- 100000000
-----------
    1001101
```

该结果就是十进制数77。

现在把两数颠倒位置后再做一遍。在十进制中，减法题目对应于：

```
  176
-253
-----
  ???
```

而在二进制中，即是：

```
  10110000
- 11111101
----------
  ????? ???
```

步骤1　从11111111中减去减数。得到补数：

```
  11111111
- 11111101
----------
  00000010
```

步骤2　把步骤1中的补数和被减数相加：

```
  10110000
+ 00000010
----------
  10110010
```

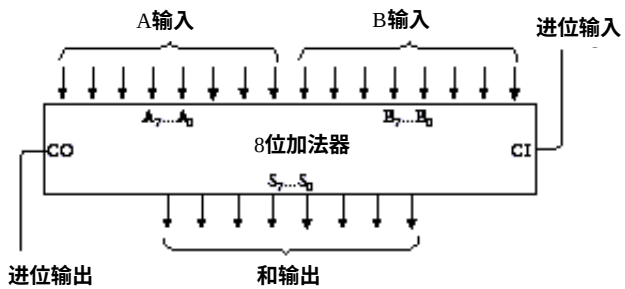
现在，11111111必须再从结果中减掉。当减数比被减数小时，可以通过先加　1再减去100000000来达到此目的。但现在这样做却会用到借位。所以，我们先用　11111111减去步骤2中的结果：

```
  11111111
-10110010
---------
  01001101
```

这实际上是对步骤2中得到的结果取反。最后的结果是77，而真正的答案应该是－77。

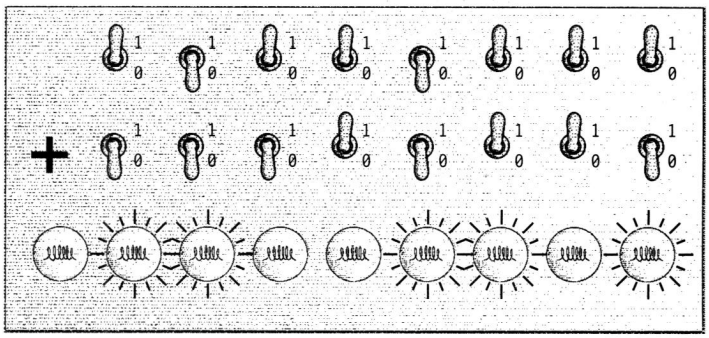
现在，已经可以改进加法机使它既能执行加法操作亦能执行减法操作。为使简便起见，这个加/减法机只执行被减数大于减数的减法操作，即差为正数的操作。

该加法机的核心部件是由逻辑门集成的8位全加器：

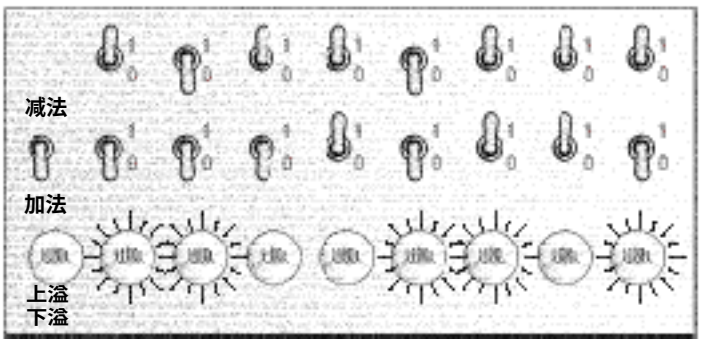


前面讲过输入 A_0～A_7及B_0～B_7连接到开关上，用于表示 8位操作数。进位输入端接地。S_0～S_7连接8个灯泡，用于表示加法的和。由于和可能会是 9位数，进位输出端也连了一个灯泡。

控制面板如下图所示：

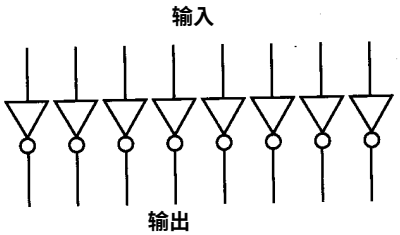


上图中，开关被设为 183（或10110111）和22（或00010110），产生的结果是 205或11001101)。用于加/减法的新的控制面板有一点儿修改，它包含了一个用于选择做加法还是做减法的额外开关。

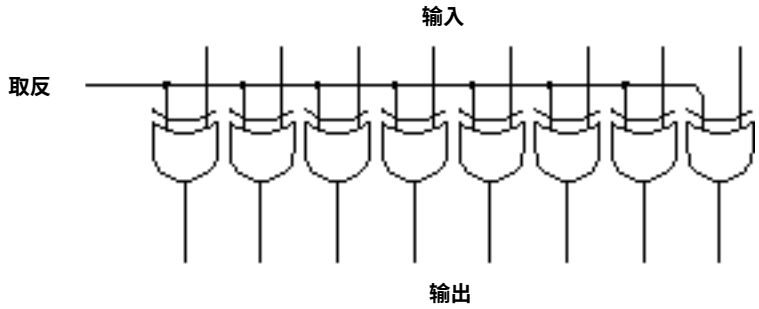


如图所示，当这个开关向下时表示选择加法运算，反之是选择减法运算。此外，只有最右边的8个灯泡用于表示结果，第九个灯泡用来标识上溢 /下溢，它指明了一个不能用 8个灯泡表示的数。当加法操作得到的和大于 255（称为上溢）或减法计算中出现一个负数（下溢）时，这个灯泡就会亮。减数比被减数大时，结果就是一个负数。

这个加法机主要增加了一个求 8位二进制数的补数的电路。由于一个数的补数就是取其每一位的相反数，所以这个电路看起来很简单，就是 8个反向器而已。



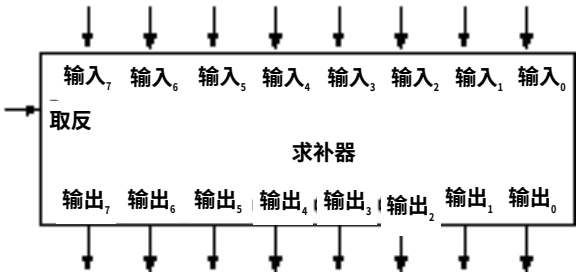
该电路存在一个问题，就是它不分情况地对输入求反。我们需要一台既能做加法又能做减法的机器，而此电路只有做减法时才取反。对它进行一下改进，如下图所示：



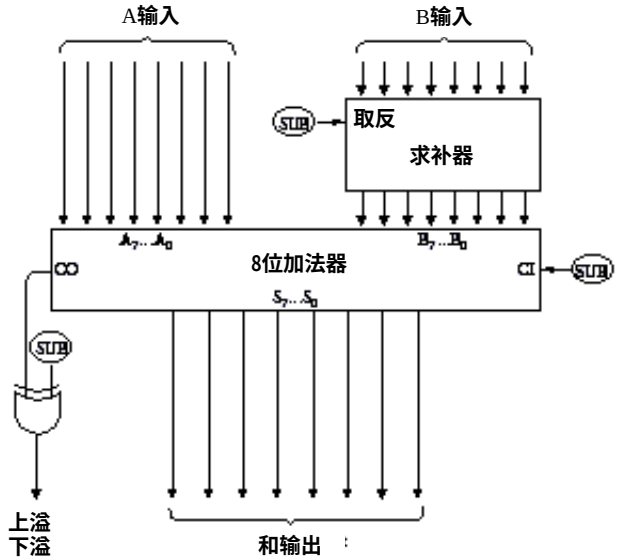
图中标识为“取反”的信号输入到每一个异或门中。回忆一下异或门的功能：

XOR	0	1
0	0	1
1	1	0

如果“取反”信号为 0，则异或门的 8个输出和 8个输入是相同的。例如，如果输入是01100001,则输出也是 01100001；若“取反”信号为 1，则输出取反。例如，当输入是01100001时，输出为10011110。让我们把8个异或门集成到一个盒子里，称为求补器：



求补器、8位加法器及一个异或门可按下图连接：



注意上图中有3个信号都标识为“ SUB ”，这是加/减法转换开关。当该信号为0时做加法，为1时做减法。做减法时， B输入在送入加法器之前先求补。此外，做减法时，通过设置加法器的进位输入端 (CI)为1，使由加法器得到的结果加 1。对加法而言，求补电路没有起作用，CI输入也就是0。

“ SUB ”信号及加法器的CO输出作为异或门的输入来控制表示上溢 /下溢的小灯泡。如果“ SUB ”信号为0（表示做加法），则当CO输出为1时灯泡点亮，这表示加法的和大于255。

当做减法时，如果被减数大于减数，则加法器的 CO端正常输出1，这表示在减法的最后一步中要减去100000000。所以，只有当加法器的 CO输出为0时，上溢/下溢灯泡才被点亮。这时减数大于被减数，差是个负数。上面这个加 /减法器现在还不能表示负数。

你一定兴致勃勃地想知道该如何实现减法了。

本章一直在谈论负数，但没有指出二进制负数的表示方法。你可能会认为它的表示和十进制负数一样，只需在数的前面加个负号。例如，－ 77在二进制中写成－1001101。你当然可以这么表示，但别忘了用二进制数的目的在于只用 0和1表示所有的东西，当然也包括一个小小的负号了。

你可以用某一位代替负号，当该位为 1时就表示负数，为0时表示正数，这似乎也是可行的。但还有一种方法，它不仅能表示负数，而且还很适于把正数和负数相加到一起。这种方法的不足之处是你必须提前决定数字需要多少位。

通常用来表示正、负数的方法的好处是这种方法能表示所有的正数、负数。我们把 0想象成向一个方向延伸的无穷的正数流和向另一个方向延伸的无穷的负数流的中点：

… －1 000 000 -999 999… －3 －2 －1 0 1 2 3… 999 999 1 000 000 …

但是,如果并不需要无限大或无限小的数，而是完全可以确定计算中所遇到的数的范围，情况便有所不同了。

下面来看看帐户的例子，人们有时可以在帐户上看到负数。假设帐户上从来没有超过$500的存款，而银行给我们的预支额是$ 500，这就意味着帐户上的数字在$499～－$ 500之间。假设我们不会一次取出 $500，也不会写一张超过 $500的支票，同时我们只处理美元，而不考虑到更小的货币单位 —— 美分。

这些假设表明帐户能处理的数字范围是从－ 500～499，总共1000个数。这个限制暗示我们只能用3位十进制数，且可不用负号来表示这 1000个数。其中的关键在于我们不需要 500～999之间的正数，所以它们就可以用来表示负数。下面是其工作原理：

用500表示－500
用501表示－499
用502表示－498
⋮
用998表示－2
用999表示－1
用000表示0
用001表示1
用002表示2
⋮

用497表示 497

用498表示498

用499表示499

换句话说，以5、6、7、8、9开头的3位数实际上都表示负数。不用如下的表示法：

－500 －499 －498 … －4 －3 －2 －1 0 1 2 3 4 … 497 498 499

而用这样的表示法：

500 501 502 … 996 997 998 999 000 001 002 003 004 … 497 498 499

注意这样形成了一个环形排序，最小的负数（500）看上去是最大的正数（499）的延续。数字999是比零小的第一个负数。如果给999加上1，通常得到1000。但由于只处理3位数，所以实际上是000。

这种处理称为10的补数。要把3位负数转换成10的补数，需从999中减去它再加1。换句话说，10的补数是9的补数再加1。例如，要把－255写成10的补数，应先从999中减去255得到744，再加上1后得到745。

你可能听说过“减法不过是负数的加法”，你也可能回答过“其实还是不得不做减法”。然而，通过使用10的补数，就不用去做减法了，全部都可以用加法来计算。

假设你有余额为$143的帐户，并写了一张$78的支票，这表明你要把－78加到143上。－78的补数是999－78+1，即922。所以新的余额是143＋922（忽略上溢），即65。若我们再写一张$150的支票，则必须减去150，用补数表示就是850。先前的余额065加上850等于915，所以，新的余额实际上是－$85。

二进制中对应的系统称为2的补数。假设我们用8位二进制数工作，范围从00000000～11111111，对应于十进制的0～255。这时如果你想要表达负数，则以1开头的每个8位数都表示一个负数，如下所示：

二进制数	十进制数
10000000	－128
10000001	－127
10000010	－126
10000011	－125
⋮	
11111101	－3
11111110	－2
11111111	－1
00000000	0
00000001	1
10000010	2
⋮	
01111100	124
01111101	125
01111110	126
01111111	127

你可以表示的数的范围从－128～127。最左边的一位称为符号位，1表示负数，0表示正数。

要计算2的补数得先求出1的补数再加上1，这等同于先求反再加1。例如，十进制数125是01111101，要用2的补数来表示－125，可先取反得10000010，再加1就得到10000011。可用上表来验证这个结果。要回到原来的数只需同样的操作：取反后加1。

这个系统使不用负号就能表示正、负数，它也使我们只用加法规则就可以随意进行正、负数运算。例如，计算－127+124，利用上表即得

```
  1 0 0 0 0 0 0 1
+ 0 1 1 1 1 1 0 0
-----------------
  1 1 1 1 1 1 0 1
```

和是十进制的－3。

这里要注意上溢或下溢，即结果大于127或小于－128的情况。例如，125加125：

```
  0 1 1 1 1 1 0 1
+ 0 1 1 1 1 1 0 1
-----------------
  1 1 1 1 1 0 1 0
```

因为最高位是1，结果代表一个负数：－6。再看－125加上它自已：

```
  1 0 0 0 0 0 1 1
+ 1 0 0 0 0 0 1 1
-----------------
1 0 0 0 0 0 1 1 0
```

由于限制了只取8位数，所以最左边的1被扔掉，剩下的8位表示6。

一般而言，若两个操作数的符号相同，而结果的符号与操作数的符号不相同时，这样的加法是无效的（即加法运算产生了溢出！）。

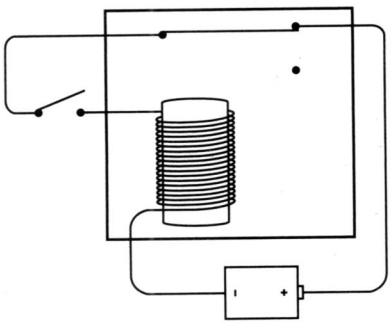
现在，二进制数可以有两种不同的使用方法。二进制数可以是无符号的或有符号的，无符号的二进制8位数的表示范围从0～255，有符号的二进制8位数的表示范围从－128～127。这些数本身不会告诉你它们是否带有符号。例如，假设有人问：“10110110对应于十进制数的几？”这时，你必须先问清楚它是无符号数还是有符号数？它可能是182或－74。

这就是二进制数的麻烦：它们仅仅是一些0和1而没有告诉它们的任何含义。

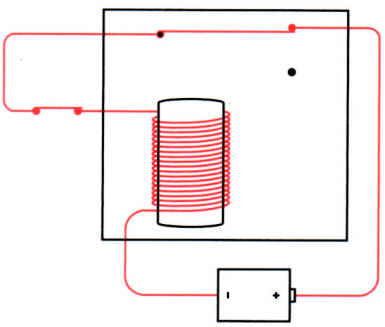
第14章　反馈与触发器

人人都知道电可以使物体运动。随便看一眼就会发现，很多家用电器中都装了电动机，如钟、风扇，食品加工机、 CD机等等。电也能使扬声器中的磁芯振动，从而使音响设备、电视机产生了声音、话音和音乐。不过，电使物体运动的一个最简单、最神奇的例子可能是电子蜂鸣器和电铃。

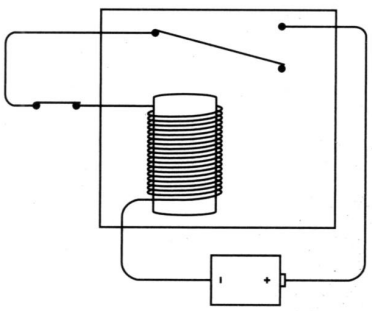
将继电器、电池、开关按如下形式连接：



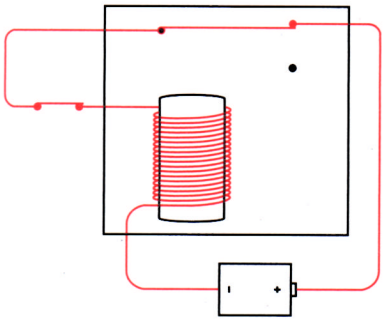
如果你觉得它看起来很奇怪，则你还没有发挥出你的想像力。我们还从未见过如此连接的继电器。原来的继电器中，输入和输出通常是分开的，这里却构成一个闭环。当闭合开关时，电路连通了：



接通的电路使电磁铁把金属簧片拉下来（电流的作用）：

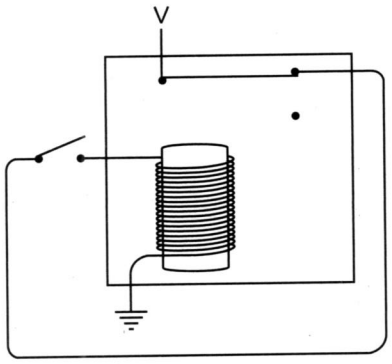


当金属簧片改变位置后，电路不再完整，电磁铁失去了磁性，金属簧片又弹回原来的位置：

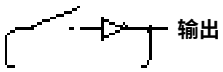


这样，电路便又一次接通了。可见，只要开关是闭合的，金属簧片就会上下跳动——使电路闭合或断开——并制造一种声音。如果金属簧片制造了一种刺耳的声音，它就构成了一个蜂鸣器。如果金属簧片附上一把小锤子，再加一个金属锣，它就构成了一个电铃。

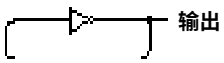
有两种方法可用来连接继电器以构造一个蜂鸣器，下面是另一种方法的描述：



你可能从上述图中认出了这是第11章介绍过的反向器，所以电路可以简化为：



对于反向器而言，当输入为0时，输出为1；输入为1时，输出为0。在该电路中闭合开关会使反向器中的继电器间断地闭合和断开。如果去掉开关，可以使反向器连续地工作，如下图示：



这幅图似乎在演示一种逻辑矛盾，反向器的输出是和其输入相反的，但是在这里，其输出同时又是其输入。需要特别指出的是，反向器实际上是一个继电器，而继电器从一个状态转换到另一个状态是需要时间的。所以，即使输入和输出是相等的，输出也会很快地改变，成为输入的倒置（当然，随即输出也就改变了输入，如此反复）。

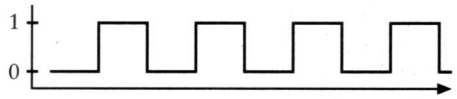
电路的输出是什么呢？其实就是提供电压和不提供电压之间的变换。或者说输出要么是0，要么是1。

这个电路称为振荡器，它和我们以前见到的每样东西都有本质上的区别。以前，所有的电路都靠手动地断开或闭合开关来改变状态，而振荡器却不需要人的干涉，它可以自主地工作。

当然，单独的一个振荡器不会有什么用，但在本章的后面及接下去的几章里，你会看到这个电路和其他电路连接后构成了自动控制中一个十分关键的部分。所有计算机都靠某种振荡器来使其他部件同步工作。

振荡器的输出是0和1的交替序列，可以用下图形象地来表示它：

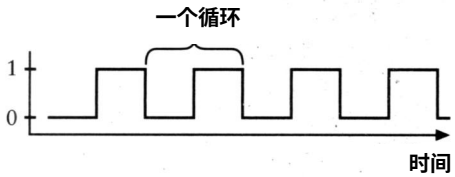
图中，水平轴表示时间，垂直轴表示输出是 0或1：



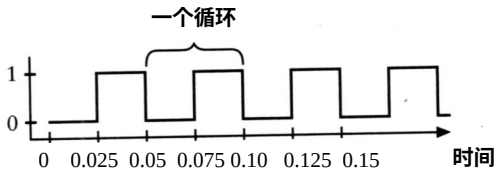
此图表示随着时间的变化，振荡器的输出在 0和1之间交替变化。基于这个原因，振荡器有时称为时钟（clock），因为通过对振荡次数记数还可确定时间。

那么，振荡器运行的速度有多快呢？也就是说，金属簧片上下跳动的频率是多少？每秒有多少次呢？很明显，这依赖于继电器是如何构造的。容易想到，一个大的、笨重的继电器只能迟钝地上下摆动；而一个小的、轻巧的继电器可以迅速地跳动。

我们把振荡器从某个时间的输出开始，经历一段变化又回到同样输出的这一段间隔称为振荡器的一个循环（cycle）：



一个循环所需要的时间称为振荡器的周期。假设一个振荡器的周期是0.05秒，则可以在水平轴上标出时间：



振荡器的频率是周期的倒数。本例中，若振荡器的周期是 0.05秒，则其频率是1÷0.05秒，即每秒钟20个循环。这表明振荡器的输出每秒钟改变 20次。

每秒循环数与每小时英里数、每平方英寸磅数、每份食物（饮料）的卡路里数等毋需多解释的术语一样是一个很容易理解的概念，但已不常用。为了纪念第一个发送和接收无线电波的人 —— 鲁道夫·赫兹(1857-1894)，我们用“赫兹”这个词表示每秒的循环数。这个用法

始于20世纪20年代的德国，后来传到其他国家。

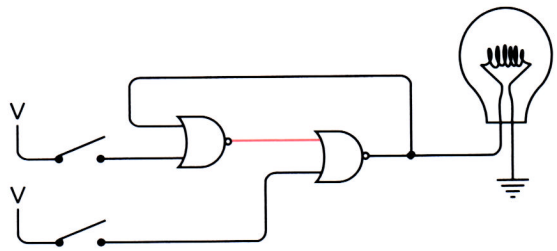
于是，我们可以说这个振荡器的频率是20赫兹，或直接简写为20Hz。

到目前为止，我们只是在假设一个振荡器的速度。到本章末尾，我们可以构造一种器件来真正地测量一个振荡器的速度。

为了构造这个器件，先看一个用特殊方式连接的一对或非门。或非门的特点是只有两个输入都为0时，输出才为1：

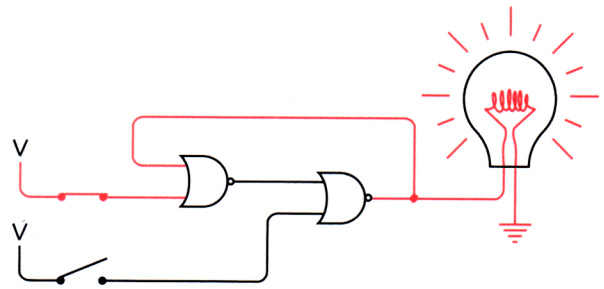
NOR	0	1
0	1	0
1	0	0

下图是含有两个或非门、两个开关和一个灯泡的电路：

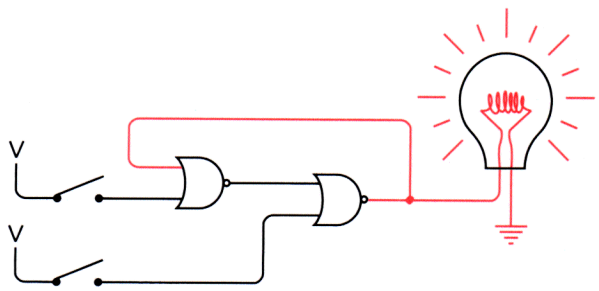


注意图中奇特的连接方式：左边或非门的输出是右边或非门的输入，右边或非门的输出是左边或非门的输入。这是一种反馈。事实上，这和在振荡器中类似，输出又返回作为一种输入。这是本章中大部分电路的特点。

在上图电路中，一开始，只有左边或非门的输出有电流，因为它的两个输入均为0。现在闭合上面的开关，左边或非门的输出变为0，于是右边或非门的输出变为1，灯泡点亮：

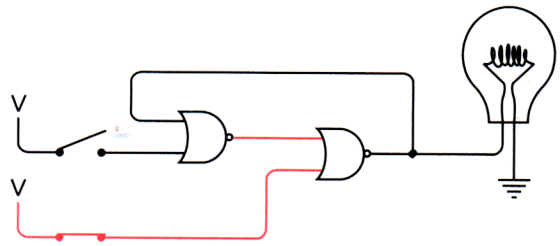


神奇之处在于当你断开上面的开关时，由于或非门的输入中只要有一个为1，其输出就是0，因而左边或非门的输出不变，灯泡仍然亮着：

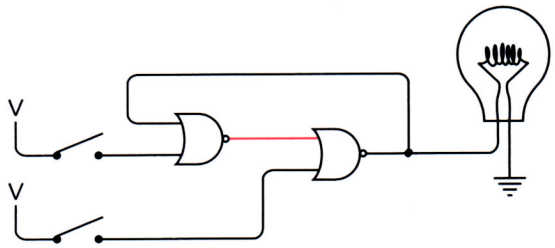


你不觉得奇怪吗？两个开关都断开着，和第一幅图一样，但灯泡却亮着。这种情形和以前所见到的完全不同。通常，一个电路的输出仅仅依赖于输入，这里的情况却不一样。无论断开或闭合上面的开关，灯泡总是亮着。这里开关对电路没有什么影响，原因是左边或非门的输出一直是0。

现在闭合下面的开关。由于右边或非门的输入中有一个是 1，则其输出变为0，灯泡熄灭。左边或非门的输出此刻变为 1：



现在，再断开下面的开关，灯泡仍旧不亮：



此电路和初始电路一样。然而这回却是下面开关的状态对灯泡没有什么影响。总结起来就是：

- 闭合上面的开关使灯泡点亮，当再断开时，灯泡仍然亮着。
- 闭合下面的开关使灯泡熄灭，当再断开时，灯泡仍然不亮。

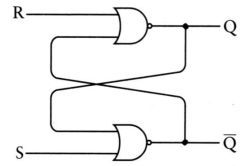
电路的奇特之处是：有时当两个开关都断开时，灯泡亮着；而有时，当两个开关都断开时，灯泡却不亮。当两个开关都断开时，电路有两个稳定状态，这样的一个电路称为触发器。触发器是1918年在英国射电物理学家 William Henry Eccles(1875-1966)和F.W.Jordan的工作中发明的。

触发器电路可以保持信息，换句话说，它有记忆性。它可以“记住”最近一次是哪个开关先闭合的。如果你遇到这样一个触发器，它的灯泡亮着时，你可以确定最近闭合的是上面的开关；而灯泡灭着时则是下面的开关。

触发器和跷跷板很像。跷跷板有两个稳定状态，它不会长期停留在不稳定的中间位置。你只要一看跷跷板就知道哪边是最近被压下来的。

触发器是十分关键的工具，尽管你现在可能还没看出来。它们赋予电路“记忆”，使其知道以前曾有过的状态。想像一下，如果你没有记忆力，你该如何去数数，你记不住你刚数过的数，当然也无法确定下一个数是什么。同样，一个能计数的电路（本章后面要提到）必定需要触发器。

触发器有很多种，刚才所看到的是最简单的一种，称为 R-S（或 Reset-Set，复位/置位）触发器。下面以对称的方式把它重新绘出来：



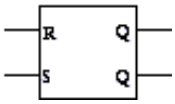
用于点亮灯泡的输出称为Q，另一个输出$\bar{Q}$是Q的倒置。如果Q是0，$\bar{Q}$就是1，反之亦然。两个输入端S（Set）和R（Reset）分别表示置位和复位。你可以把“置位”理解为把Q设为1，而“复位”是把Q设为0。当S为1时（对应于前面图中闭合上面开关的情况），Q变为1而$\bar{Q}$变为0；当R为1时（对应于前面图中闭合下面开关的情况），Q变为0而$\bar{Q}$变为1。当S和R都为0时，输出保持Q原来的状态。输入与输出的关系小结于下表中：

输入		输出	
S	R	Q	$\bar{Q}$
1	0	1	0
0	1	0	1
0	0	Q	$\bar{Q}$
1	1	禁止	

这张表称为功能表、逻辑表或真值表。它指明不同的输入组合能产生不同的输出结果。由于R-S触发器有两个输入端，因而不同的输入组合有4种，分别对应于表中的4行。

注意表中倒数第2行中S和R均为零，而输出标识为Q和$\bar{Q}$。这表示当S和R输入均为零时，Q和$\bar{Q}$端的输出保持S、R同时设为0以前的输出值。表中最后一行说明S和R输入都为1是非法的、禁止的。这是因为S、R同时为1时，两个输出Q和$\bar{Q}$均为零，这与Q和$\bar{Q}$互为倒置的关系相矛盾。所以，当你用R-S触发器设计电路时，要避免使R、S输入同时为1的情况。

R-S触发器通常画成有两个输入，两个输出的方块图，如下图所示：



R-S触发器能够记住哪一个输入端最近被输入高电位，这确实很有趣。但更有用的电路应该能记住某个特定时间点上上一个信号是0还是1。

在实际构造这种电路之前，先来思考一下它的行为功能。它需要两个输入，其中一个称为数据端（Data）。像所有数字信号一样，数据端输入可以是0或1。另一个输入称为保持位（Hold that bit）。通常情况下，保持位设为0，这时，数据端对电路没什么影响。当保持位置为1时，电路就反映出数据端的值。接着，保持位又置为0，这时，电路将记住数据端输入的最近一个值。数据端信号的任何改变不会对电路再有影响。

换句话说，它的功能表可以这样写：

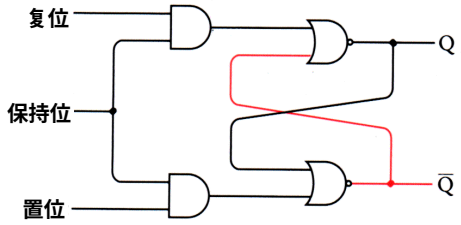
输入		输出
数据端	保持位	Q
0	1	0
1	1	1
0	0	Q
1	0	Q

在前两种情况下，保持位置为 1，Q端输出和数据端输入相同；后两种情况下，当保持位置为0时，Q端输出和它以前的值相同，即保持原状态。注意，后两种情况中当保持位为 0时，Q端输出不再受数据端输入的影响，功能表可以简化表示为：

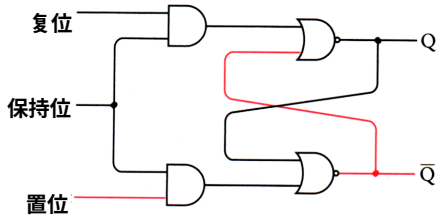
输入		输出
数据端	保持位	Q
0	1	0
1	1	1
X	0	Q

X表示不关心其取值情况，它的值对于电路输出没有影响。

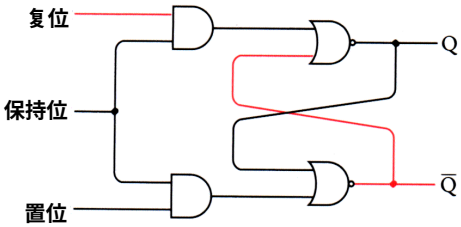
基于R-S 触发器来实现保持位的功能要求在输入端增加两个与门，如下图所示：



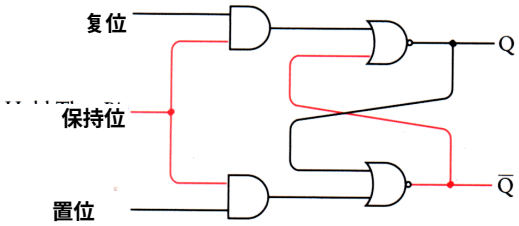
要使与门输出为1，两个输入端必须同时为1。在上图中，Q输出为0，而$\overline{Q}$输出为1。只要保持位置为0，置位信号对于输出就没有影响：



同样，复位信号对电路输出也没有影响：



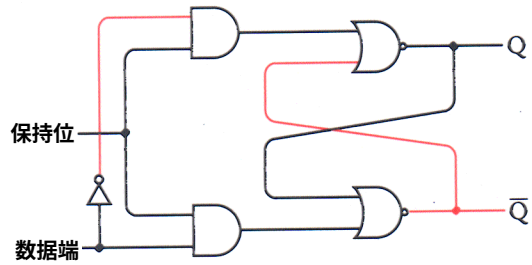
只有当保持位信号是1时，电路的功能才和前述的R-S触发器相同：



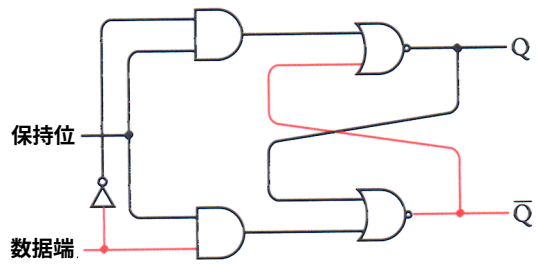
这时，由于上面与门的输出和复位端输入相同，而下面与门的输出和置位端输入相同，所以此电路的功能就和普通的R-S触发器是一样的了。

但我们还没有达到目标，我们只想要两个输入，而不是三个，怎么办呢？前面讲过 R-S触发器中两个输入同时为1的情况是禁止的；而两个输入同时为零的情况没有什么意义，因为那只是输出保持不变的简单情况。这里，只要将保持位置为 0，就可以完成同样的功能。

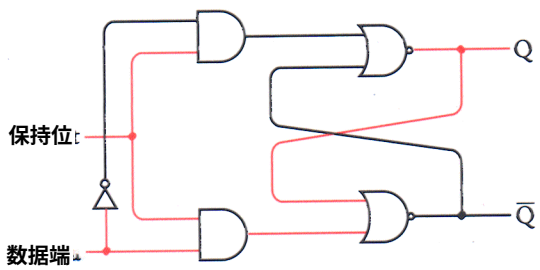
可见，真正有意义的输入是S为0，R为1或R为0，S为1。把数据端信号当作置位信号，它取反后的值就是复位端信号，如下图示：



在这种情况下，S和R输入以及输出Q均为0，$\bar{Q}$为1。只要保持位为0，数据端输入对于电路输出就没有影响：



当保持位为1时，电路反映出数据端输入的值：



Q端输出现在和数据端输入是一致的，$\bar{Q}$则相反。现在，保持位又回到0：