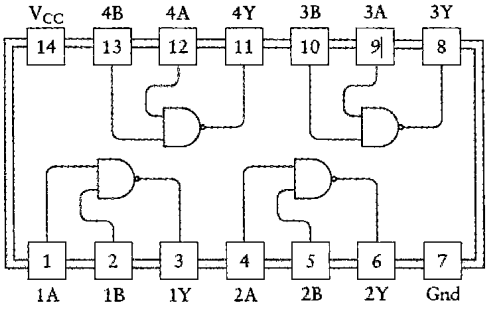
一部《星球大战》电影发行前的时代，而 VLSI 处于萌芽阶段。那时，人们使用几种不同的技术来制造集成电路的组件。有时每一种技术被称之为一个 IC 家族，到 20 世纪 70 年代中期，有两个“家族”盛行开来：TTL 和 CMOS。

TTL 代表 transistor-transistor logic（晶体管-晶体管逻辑）。20 世纪 70 年代中期，如果你身为一名数字电路设计师（用 IC 设计大规模电路），那么一本 1.25 英寸厚、由德州仪器公司在 1973 年出版的名为 *The TTL Data Book for Design Engineers*（《TTL 工程师设计数据手册》，以下简称《TTL 数据手册》的书将会是你书桌上的常客。这是一本德州仪器和其他几个公司出售的 TTL 集成电路 7400 系列完整的参考书，由之所以这样称呼是因为这个 IC“家族”的每一名“成员”都是以数字 74 开头。

7400 系列中的每一个集成电路都是由以特定方式连接的预留逻辑门组成。一些芯片提供简单的预留的逻辑门，设计者可以用它们来组成更大规模的组件；另外一些芯片则提供通用组件，例如：触发器、加法器、选择器以及解码器。

7400 系列中第一个集成电路标号即为 7400，在《TTL 数据手册》中这样描述它——“四个双输入正与非门”。这意味着这个特殊的集成电路包含四个双输入与非门。“正”与门则是指 1 对应为有电压，而 0 对应为没有电压。下图是一个 14 管脚的芯片，数据手册中的一张小图显示了管脚对应的输入与输出。



上面这张图为芯片的俯视图（管脚在下面），小凹槽位于左边。

14 号管脚标注为 V_{CC}，与符号 V 一样，用来代表电压（顺便说一下，大写字母 V 的双下标代表电压源。下标的字母 C 指晶体管的电压输入端，即集电极，collector）。7 号管脚标注的 GND 代表接地（ground）。在特定电路中使用的所有集成电路都必须有接电源端与接地端。

拿 TTL7400 系列来说，V_{CC} 值必须介于 4.75V 和 5.25V 之间。换句话讲，电压必须在 5V ± 5% 的范围。电压低于 4.75V 时，芯片将无法工作。而高于 5.25V 时，芯片将被烧坏。即便有一个 5V 的电池，那也不能用来对 TTL 进行供电，因为电池的电压不可能刚好适合这些芯片。通常情况下，TTL 需要从墙上接入电源。

7400 芯片中每一个与非门有两个输入端和一个输出端，且相互独立工作。上一章中已经区分了输入为 1（有电压）或者为 0（无电压）的情况。实际上，与非门的输入端电压可以为 0V（接地）到 5V（V_{CC}）范围内的任一值。TTL 中，当电压介于 0～0.8V 任一值则可以认为是逻辑“0”，而介于 2～5V 则可以认为是逻辑“1”。0.8～2V 范围的电压输入则应当尽量避免。

TTL 的典型输出是以 0.2V 表示逻辑“0”，以 3.4V 表示逻辑“1”。考虑到电压值不稳定，有时会有一些波动，集成电路的输入和输出端有时不用“0”和“1”表示，而是用“低”和“高”表示。此外，有时候低电压可以表示逻辑“1”，而高电压则可以表示逻辑“0”，这种配置称为“负逻辑”。7400 芯片被称为“四个双输入正与非门”，而这里的“正”则代表了上述所讲的正逻辑。

如果 TTL 的典型输出 0.2 V 代表逻辑“0”，而 3.4 V 代表逻辑“1”，那么这个输出电压确实是在输入允许的范围内，即逻辑“0”为 0～0 . 8 V，且逻辑“1”为 2～5 V，这就是 TTL 可以隔离噪声的原因。逻辑“1”输出的电压哪怕下降 1.4V 也仍可以作为逻辑“1”的高电压输入，同样的，逻辑“0”输出的电压哪怕升高 0.6V 也仍可以作为逻辑“0”的低电压输入。

影响一个集成电路性能的最重要因素可以认为是传播时间（propagation time），也就是输入端发生变化引起输出端发生相应变化所需要的时间。

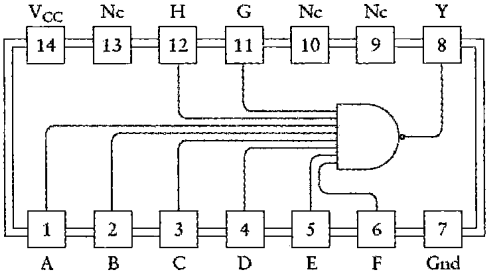
通常以纳秒来衡量芯片的传播时间，缩写为 nsec，即 ns。1 纳秒是非常短的时间。千分之一秒称为毫秒，百万分之一秒称之为微秒，那么十亿分之一秒则称为纳秒。7400 芯片中与非门的传播时间应该保证小于 22 ns，即 0.000000022s。

感觉不到纳秒长短的并非只有你一人。地球上的所有人对纳秒只有概念上的理解，除此之外别无所有。纳秒比人类感觉到的任何事情都要短暂得多，以至于永远无法理解它。任何解释只会将纳秒变得更加难以捉摸。比如，当你拿着一本离你脸部距离为 1 英

尺的书时，那么纳秒可以定义为光从书页到你的眼睛的时间，但这种解释可以使你更好地认识纳秒么？

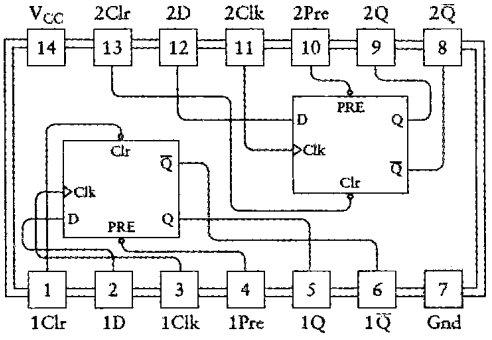
然而，纳秒使计算机成为可能。正如在第 17 章中看到的，计算机处理器迟钝地做着简单的事情——从存储器中取出一个字节放到寄存器中，再将两个字节相加，然后将结果存放回存储器。迅速地完成这些操作是计算机（并非第 17 章中的计算机，而是当今使用的计算机）能完成任何实际工作的唯一原因。诺依斯说过："当更好地认识了纳秒后，从概念上来讲计算机操作是相当简单的"。

继续阅读《TTL 数据手册》会发现书中很多熟悉的小条目。7402 芯片有 4 个双输入或非门，7404 芯片有 6 个反相器，7408 芯片有 4 个双输入与门，7432 芯片有 4 个双输入或门，以及 7430 芯片有一个 8 输入与非门，如下图所示。

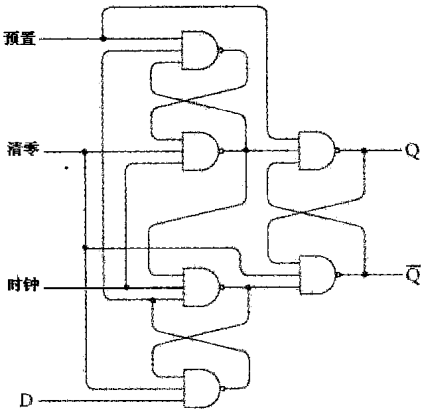


缩写 Nc 表示无连接（no connection）。

7474 芯片是另一个较为熟悉的芯片。它是一个"带预置和清零的双D型正边沿触发器"，如下图所示。



《TTL 数据手册》中甚至还囊括了这个芯片中每个触发器的逻辑图。



除了使用的是异或门外，你会发现上面的这个图与第 14 章结尾的图很相似。《TTL 数据手册》中的逻辑表也稍微不同。

输 入				输 出	
Pre	Clr	Clk	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	L	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

表格中，“H”代表高电平，“L”代表低电平。当然，可以将这些想象成 1 和 0。在触发器中，预置（Pre）和清零输入（Clr）通常为 0；这里通常为 1。

继续阅读《TTL 数据手册》会发现，7483 芯片是一个 4 位二进制全加法器，74151 是一个 8-1 的数据选择器，74154 芯片是一个 4-16 的解码器，74161 芯片是一个 4 位同步二进制计数器，以及 74175 芯片是一个带清零的 4 输入 D 型触发器。从上述芯片中挑出两种可以制作一个 8 位锁存器。

现在你应该知道我是怎么想出从第 11 章中就开始使用的这么多不同种类的组件的，这些都是从《TTL 数据手册》借鉴而来。

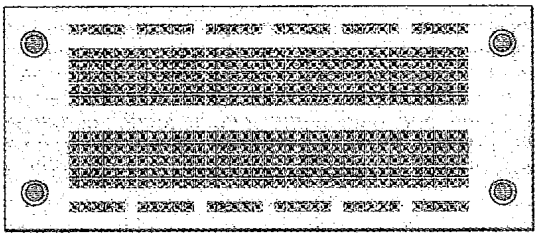
作为一名数字电路设计工程师，你应当多花点时间看完《TTL 数据手册》这本书，使自己熟悉用得到的 TTL 芯片类型。一旦你熟知了使用的工具，那么就可以使用 TTL 芯片构造一台第 17 章中的计算机。将芯片连接起来比将一个个的晶体管连接起来容易得多，但你可能不会考虑使用 TTL 来做 64KB RAM 阵列。1973 年出版的《TTL 数据手册》中，列出的容量最大的 RAM 芯片仅仅只有 256×1 位，制造 64 KB 需要 2048 个芯片！对制造存储器来讲，TTL 绝非最好的技术。关于存储器将在第 21 章中详细讨论。

或许你想使用更好一点的振荡器。只要将 TTL 反相器的输出连接到输入，就会获得一个振荡器，而且其振荡频率更容易计算。这种振荡器使用石英晶体制造相当简单，石英晶体放在带有两个引线的密封小扁罐中。这些石英晶体的振荡频率在一个特定的值，通常情况下是每秒至少振荡一百万个周期，称 1 兆赫兹，缩写为 MHz。如果要使用 TTL 制造第 17 章中的计算机，那么需要时钟频率为 10 MHz 才可以使其运行良好，每条指令执行时间为 400 ns。当然，这比使用继电器来做任何我们所构想的事情都要快。

芯片家族中另一位明星（至今仍是）是 CMOS，CMOS 表示互补金属氧化物半导体（complementary metal-oxide semiconductor）。如果你是一名 20 世纪 70 年代中期的使用 CMOS 集成电路进行电路设计的爱好者，那么可能会使用到一本名为《CMOS 数据手册》的书作为参考源，该书由美国国家半导体公司出版，可以在当地的 Radio Shack 商店买到，书中涵盖了 CMOS 家族中 4000 系列的 IC 的信息。

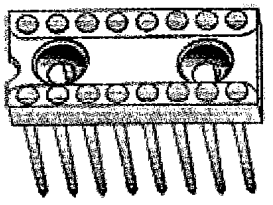
TTL 供电电压要求在 4.75～5.25V 范围内。而对于 CMOS 来说，范围在 3～18V 内的电压均可，这是非常灵活的。此外，CMOS 相比 TTL 需要更少的能量，这使得用电池运行小型 CMOS 电路变得可行。CMOS 的缺点是速度慢，比如，在供电电压为 5 伏的情况下，CMOS 4008 4 位全加器可以保证的传播时间只有 750 ns。CMOS 芯片的传播速度会随着电压提高而提高——10V 时为 250ns，15 伏时为 190 ns，但仍不能与 TTL4 位加法器的 24 ns 的传播时间相媲美（25 年前，TTL 的速度与 CMOS 低功率之间的权衡是非常清晰的，斗转星移，今天 TTL 已经拥有了低功率版本而 CMOS 也有了高速版本）。

在实际应用中，芯片的连接是在一个塑料“面包板”（如下图所示）完成的。



每一短行有 5 个孔，在塑料板背面这 5 个孔通过电线相连。把芯片插入到面包板中，芯片将横跨在中间长槽的两侧，芯片的管脚则分别插到槽两侧的孔里，这样芯片的每一个管脚都会与其他四个孔里的管脚相连接，通过在孔之间连接电线可以实现芯片之间的连接。

使用一种叫做“钢丝包装”（wire-wrapping）的技术可以使芯片之间连接更加牢固，芯片插入到带有几个长长的方柱的插槽中，如下图所示。



每一个柱体对应芯片的一个管脚，而插口本身则插入到事先穿孔的薄板中。在板子的另一面，使用特殊的钢丝包装枪将每一个柱体周围紧紧包上绝缘线。柱体的直角边缘则从绝缘线中破出，与导线相连。

如果实际应用中使用集成电路制造一个特定的电路，那可能会用到“印刷电路板”（printed circuit board）。很久以前，这是集成电路爱好者做的事情。这种电路板上面布满了洞，并且被一层薄铜片覆盖。基本上，可以让防酸物质覆盖铜片上所有你想保护的地方，而使用酸蚀刻剩余的部分，接着就可以将集成电路的插口（或者是集成电路本身）直接焊接到电路板上的铜片上，但由于集成电路中存在很多互相连接，仅覆盖一层铜片通常情况下无法完成电路，所以商业制造的印刷电路板有多层互连。

到 20 世纪 70 年代早期，使用集成电路在一块电路板上制造一个完整的计算机处理器变得可能，实际上这距离将整个处理器放入一块芯片中，只是一个时间问题。虽然德州仪器公司在 1971 年为一块单芯片计算机提交了专利申请，但真正制造出一块这种单片

机芯片的荣誉却属于英特尔公司（英特尔公司成立于 1968 年，由仙童公司以前的雇员罗伯特·诺伊斯和戈登·摩尔合伙创办）。1970 年，英特尔发售了第一款产品，一个可以存储 1024 位数据的芯片，在当时这是单一芯片中可以存储的最大位数。

英特尔在为日本吉康（Busicom）公司生产的可编程计算器设计芯片的过程中，决定采取一种不同的方法。正如英特尔工程师特德·霍夫（Ted Hoff）说的："我想让它成为一个具有通用功能的计算机，进而可以通过编程成为一个计算器，而不是使这个设备成为一个只有一些编程能力的计算器，"这导致了 Intel 4004 的产生，它是第一块"计算机芯片"，或者叫做"微处理器"。1971 年 11 月，4040 芯片已经可以得到使用，它拥有 2300 个晶体管（依照摩尔定律，18 年后微处理器包含的晶体管数将是这个数字的 4000 倍，或者说是 1000 万。这是一个相当精确的预测）。

我们已经知道 4004 芯片包含的晶体管数目，下面是 4004 芯片另外三种重要的特征。自 4004 芯片开始，在比较微处理器性能时，通常采用三个衡量标准。

第一个标准：4004 是一个 4 位微处理器，这意味着处理器中数据通路宽度只有 4 位。每次做加、减运算时，它只能处理 4 位的数字。对比来看，第 17 章中的计算机数据通路是 8 位，因此被称为 8 位处理器。我们即将看到 8 位处理器很快就超越了 4 位处理器。但技术并没有停止于此，20 世纪 70 年代末期，16 位微处理器已经得到了应用。回想一下第 17 章中的内容，以及在 8 位处理器中进行两个 16 位数加法所必需的指令码，你就会欣喜地发现 16 位处理器带来的优势。到 20 世纪 80 年代中期，32 位微处理器诞生了，并自此一直作为家用计算机的主要处理器。

第二个标准：4004 每秒最大时钟频率为 108,000 周期，即 108 KHz。时钟频率是指连接到微处理器并驱动它运行的振荡器的最大频率，超过此时钟频率，微处理器将不能正常工作。到 1999 年，家用计算机的微处理器已经达到了 500 MHz——比 4004 要快 5000 倍。

第三个标准：4004 的可寻址存储器只有 640 字节，现在来看这个数字小得有点荒唐，但这与当时可得的存储芯片的容量是一致的。下一章中你将会看到，两年之中微处理器的寻址能力就达到了 64 KB，与第 17 章中计算机的能力比肩。1999 年英特尔生产的芯片可以寻址 64 TB 的空间，尽管当时多数人的家用电脑 RAM 容量还不到 256 MB。

上述三个数字指标并不能影响一台计算机的计算能力。比如，4 位处理器同样可以实

现 32 位数字加法，只不过是将其简单拆分为 4 位的数来进行。某种意义上讲，所有的数字计算机都是相同的，如果一台处理器从硬件上无法做到另外一台可以做到的事情，那么它可以通过软件途径做到，最终它们可以完成相同的事情，这是 1937 年图灵在论文里面关于可计算性的一种定义。

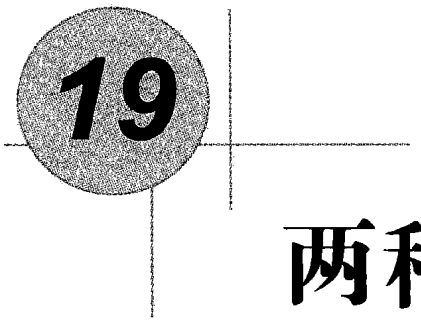
然而，速度是处理器之间的根本不同点，同时速度也是我们使用计算机的一大原因。

最大时钟频率（maximum clock speed），也称为主频，是影响处理器速度的决定性因素之一。时钟频率决定了执行一条指令所需要的时间，处理器的数据位宽也影响处理器的速度。尽管 4 位处理器可以完成 32 位数字的加法，但速度是不能与 32 位处理器相媲美的。然而，令人感到迷惑的是，处理器可寻址存储器的最大空间对处理器速度也是有影响的，首先，可寻址存储器看上去只反映了处理器的某些能力，尤其是在需要大容量存储器的前提下进行数据处理的能力，而与处理器速度无关。但其实，处理器可以利用某些存储器地址去控制其他的介质来存取信息，这样就绕开了存储器容量的限制（比如，假设在特定的存储器地址写入一个字节就在一个纸带上穿一个孔，而从存储器中读出一个字节等于从纸带上读取一个孔一样）。可是这种处理办法会降低整个计算机的速度——这就又回到了速度问题！

当然，这三个数字只能粗略地反映微处理器操作的速度，它们并不能体现出微处理器的内部构造以及机器指令代码的效率和能力。随着处理器变得越来越复杂，以前通过软件完成的任务都可以在处理器上完成，我们将会在后面的章节中看到这方面的例子。

即使所有的计算机具有相同的计算能力，即使它们只能做和图灵设计的早期计算机一样简单的事情，但有一点是无法回避的，那就是处理器的速度最终决定了其用途。比如那些表现比人脑还慢的计算机是毫无用处的，跟进一步来说，如果处理器需要用一分钟来画一帧图像，那么对于现代计算机而言，要想在其屏幕上播放电影也是不可能实现的。

回到 20 世纪 70 年代中期，虽然 4004 有很多局限性，但毕竟只是个开始。到 1972 年 4 月，英特尔发布了 8008 芯片——一个时钟频率为 200 KHz、可寻址空间为 16 KB 的 8 位微处理器（瞧，用三个数字来总结一个处理器是多么简单的事）。后来在 1974 年 5 月，英特尔公司和摩托罗拉公司同时发布了 8008 微处理器的改进版，正是两款芯片改变了整个世界。



两种典型的微处理器

微处理器——正是它，将计算机中央处理器的所有构成组件整合在一起，集成在一个硅芯片上——诞生于 1971 年。它的诞生有着很好的开端：第一个微处理器，即 Intel 4004 系列，包括了 2300 个晶体管。到现在，大约三十年过去了，家用计算机的微处理器中的晶体管数量也逐步逼近 10,000,000 个。

从本质上说，微处理器实际上所做的工作一直没有变。在现在的芯片上，新增的几百万个晶体管所做的很多事情令我们眼前一亮，但我们正处于微处理器探索的初期，过多的关心这些当代的芯片并不合适，因为它们只会分散我们的注意力而无法帮助我们去学习与理解它。为了更清晰地认识微处理器是如何工作的，让我们首先来看一下最原始的微处理器。

我们要讨论的微处理器出现于 1974 年。在这一年，英特尔公司在 4 月推出了 8080 处理器，摩托罗拉公司——从 20 世纪 50 年代生产半导体和晶体管——在 8 月推出了 6800 处理器。不仅如此，当年还有其他的一些微处理器面世。同年，德克萨斯仪器设备公司

（Texas Instruments）推出了 4 位的处理器 TMS 1000，它用于多种计算器、玩具和设备；国家半导体公司（National Semiconductor）推出了 PACE——首个 16 位微处理器。但当我们回顾历史的时候就会发现，8080 和 6800 是两个最具有重大历史意义的芯片。

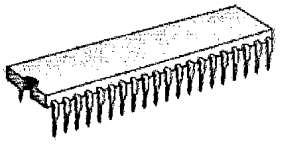
英特尔为 8080 最初定的价格为 360 美元，这个价格对 IBM 的 System/360 来说是极大的讽刺。System/360 是大型机处理系统，用户大都是一些大公司，售价动辄几百万美元（今天，你用 1.95 美元就可以买到一块 8080 芯片）。这并不是说 8080 可以与 System/360 相提并论，但在几年之内，IBM 自己也关注起这些非常小的计算机。

8080 是一个 8 位的微处理器，它包括 6000 个晶体管，运行的时钟频率为 2 MHz，寻址空间为 64 KB。摩托罗拉的 6800（今天的售价也是 1.95 美元）包括 4000 个晶体管，其寻址空间也是 64 KB。第一个版本的 6800 的运行速度为 1 MHz，但摩托罗拉于 1977 年推出了运行速度分别为 1.5 MHz 和 2 MHz 的版本。

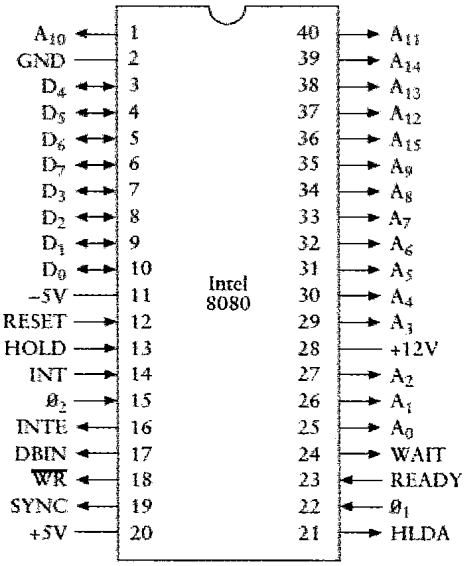
这些芯片被称为“单芯片微处理器”（single-chip microprocessors），不太准确的说法是“单芯片的计算机”。处理器只是计算机的一部分。除了处理器之外，计算机还需要其他一些设备，至少要包括一些随机访问的存储器（RAM），一些方便用户把信息输入计算机的设备（输入设备），一些使用户能够把信息从计算机中读取出来的设备（输出设备），以及其他一些能把所有构件连接在一块的芯片。本书会在第 21 章详细介绍这些构件。

现在，让我们来仔细研究一下微处理器本身。当描述微处理器的时候，我们总是习惯用一些框图来阐明其内部的构件及其连接情况。然而，在第 17 章我们已经使用了数不清的框图来描述它，现在我们将观察微处理器和外部设备的交互过程，以此来认识其内部的结构和工作原理。换句话说，为了弄清微处理器的工作原理，我们把它视做一个不需要详细研究其内部操作的黑盒。取而代之的方法是通过观测芯片的输入、输出信号，特别是芯片的指令集来理解微处理器的工作原理。

8080 和 6800 都是 40 个管脚的集成电路。这些芯片最常见的 IC 封装大约为 2 英寸长，1/2 英寸宽，1/8 英寸厚。



当然，你所看到的只是外部的封装。其内部的硅晶片是非常小的，例如在早期的 8 位微处理器中，硅晶片还不到 1/4 平方英寸。外包装可以保护内部的硅晶片，并且通过管脚提供了处理器的输入和输出访问接入点。下面给出了 8080 的 40 个管脚的功能说明图。



本书中我们所创建的所有电气或电子设备都需要某种电源来供电。8080 的一个特殊的地方就是它需要三种电源电压：管脚 20 必须接到 5V 的电压；管脚 11 需要接到-5V 的电压；管脚 28 需接 12V 的电压；管脚 2 接地。（英特尔在 1976 年发布了 8085 芯片，目的就是简化对这些电源的要求）

其他的管脚都标有箭头。从芯片引出的箭头表明这是一个输出（output）信号，这种信号由微处理器控制，计算机的其他芯片对该信号响应。指向芯片的箭头表明该信号是一个输入（input）信号，该信号由其他芯片发出，并由 8080 芯片对其响应。还一些管脚既是输入又是输出。

第 17 章所设计的处理器需要一个振荡器来使其工作。8080 需要两个不同的同步时钟输入，它们的频率都是 2 MHz，分别标记为 ϕ_1 和 ϕ_2，位于管脚 22 和 15 上。这些信号可以很方便地由英特尔生产的 8224 时钟信号发生器产生。为 8224 连接一个 18 MHz 的石英晶体后，它基本上就可以完成其余工作了。

一个微处理器通常有多个用来寻址存储器的输出信号。用于寻址的输出信号的数目

与微处理器的可寻址空间大小直接相关。8080 有 16 个用于寻址的输出信号，标记为 A_0 ~ A_{15}，因此它的可寻址空间大小为 2^{16}，即 65,536 字节。

8080 是一个 8 位的微处理器，可以一次从存储器读取或向存储器写入 8 位数据。该芯片还包括标记为 D_0 ~ D_7 的 8 个信号，这些信号是芯片仅有的几个既可以用做输入又可以用做输出的信号。当微处理器从存储器中读取一个字节时，这些管脚的功能是输入；当微处理器向存储器写入一个字节时，其功能又变成了输出。

芯片的其余 10 个管脚是控制信号（control signals）。例如，RESET（复位）输入用于控制微处理器的复位。输出信号 $\overline{WR}$ 的功能是指明微处理器需要向 RAM 中写入数据（$\overline{WR}$ 信号对应于 RAM 阵列的写输入）。此外，当芯片读取指令时，在某些时刻一些控制信号会出现在 D_0 ~ D_7 管脚处。使用 8080 芯片构建的计算机系统通常使用 8228 系统控制芯片来锁存附加的控制信号。本章在后面将会讲述一些控制信号。但 8080 的控制信号是极其复杂的，因此，除非你准备用该芯片搭建一台计算机，否则最好不要在这些控制信号上过多花费时间。

假设 8080 微处理器连接了一个 64KB 的存储器，这样我们就能独立地读写数据而不依赖于微处理器。

8080 芯片复位后，它把锁存在存储器 0000h 地址处的字节读入微处理器，通过在地址信号端 A_0 ~ A_{15} 输出 16 个 0 实现该过程。它读取的字节必须是 8080 指令，读取该字节的过程被称为取指令（instruction fetch）。

在第 17 章设计的计算机中，所有的指令（除了 HLT 指令）都是 3 个字节长，包括 1 字节的操作码和 2 字节的地址。在 8080 中，指令的长度可以是 1 字节、2 字节，或者 3 字节。有些指令使 8080 从存储器的一个特定地址读取字节到微处理器，有些指令使 8080 将一个字节从微处理器写入存储器的特定地址；还有些指令使 8080 在其内部执行而不需要访问 RAM。8080 执行完第一条指令后，接着从存储器读取第二条指令，并依此类推。这些指令组合在一起构成了计算机程序，可以用来做一些很有趣的事情。

当 8080 以最高速度 2 MHz 运行时，每个时钟周期是 500ns（1 ÷ 2,000,000 = 0.000000500s）。第 17 章中的计算机的所有指令都需要 4 个时钟周期，8080 的每条指令需要 4 ~ 18 个时钟周期，这就意味着每条指令的执行时间为 2 ~ 9μs。

也许了解某个特定微处理器的功能的最好办法就是全面地测试其完整的指令集。

第 17 章最后完成的计算机仅包括 12 条指令。一个 8 位处理器的指令数很容易达到 256，每一条指令的操作码就是一个特定的 8 位数（如果某些指令包含 2 字节的操作码，其指令集会更大）。8080 虽然没有这么多指令，但是其指令数也已经达到了 244。这看起来似乎是很多，但从总体上说，其功能并不比第 17 章的计算机强大。例如，如果想利用 8080 进行乘法或除法运算，你仍然需要自己写一小段代码。

在第 17 章曾经讲到过，为了方便地引用指令，我们为处理器的每一条指令的操作码都指派了一个特殊的助记符，而且其中的一些助记符是可以带有参数的。这种助记符只是在我们使用操作码时提供方便，它对于处理器是没有帮助的，处理器只能读取字节，对于助记符组成的文本的含义一无所知（为了讲解清楚，本书选用了 Intel 8080 说明文档中用到的部分助记符为例来说明）。

第 17 章设计的计算机的指令集包括两条非常重要的指令，我们称之为加载（Load）和保存（Store）。每条指令占 3 个字节。在 Load 指令中，第一个字节是操作码，其后的两个字节是要加载的操作数的 16 位地址。当处理器执行加载指令时，会把该指定地址中的字节加载到累加器。与之相似，当 Store 指令被执行时，累加器中的内容被保存到该指令指定的地址中。

我们可以用助记符把上述代码简写为以下形式：

```
LOD A, [aaaa]
STO [aaaa], A
```

这里的 A 表示累加器（它既是 Load 指令的目的操作数也是 Store 指令的源操作数），aaaa 表示 16 位的存储器地址，通常用 4 个 16 进制的数来表示一个地址。

同第 17 章的累加器一样，8080 的 8 位累加器也记做 A。8080 也有与第 17 章的计算机的 Load 指令和 Store 指令功能相同的两条指令，它们也称做加载（Load）和保存（Store）。在 8080 中，加载指令和保存指令的操作码分别是 32h 和 3Ah，每个操作后面也同样跟着一个 16 位的地址。在 8080 中，它们的助记符分别是 STA（Store Accumulator，表示加载到累加器）和 LDA（Load Accumulator，表示保存到累加器）：

操作码	指　令
32	STA　[aaaa], A
3A	LDA　A, [aaaa]

8080 芯片的微处理器的内部除累加器外还设置了 6 个寄存器（register），每个寄存器可以存放一个 8 位的数。这些寄存器和累加器非常相似，事实上累加器被视为一种特殊的寄存器。这 6 个寄存器和累加器一样，本质上都是锁存器。处理器既可以把数据从存储器读入寄存器，也可以把数据从寄存器存回存储器。当然，其他的寄存器没有累加器所具有的丰富的功能，例如，当把两个 8 位数相加时，其结果总是保存到累加器而不会保存到其他寄存器。

在 8080 中用 B，C，D，E，H 和 L 来表示新增的 6 个寄存器。人们通常会问以下两个问题："为什么不使用 F 和 G 来表示？"，以及"I，J 和 K 用来代表什么？"答案是，使用 H 和 L 来命名寄存器是因为它们具有特殊的含义，H 可以代表高（High）而 L 可以代表低（Low）。通常把两个 8 位的寄存器 H 和 L 合起来构成一个 16 位的寄存器对（register pair），称做 HL，H 用来保存高字节而 L 用来保存低字节。这个 16 位的值通常用来对存储器寻址，我们将在下面看到它是怎样以简单的方式工作的。

寄存器是计算机必不可少的部件吗？为什么在第 17 章搭建的计算机中并没有寄存器的踪迹？从理论上讲，这些寄存器不是必需的，在第 17 章也没有用到它们，但在实际应用中使用它们将带来很大的方便。很多计算机程序都同时用到多个数据，将这些数据存放在寄存器比存放在存储器更便于访问，因为程序访问内存的次数越少其执行速度就越快。

在 8080 中有一条指令至少用到了 63 个操作码，这条指令就是 MOV，即 Move 的缩写。其实该指令是一条单字节指令，它主要用来把一个寄存器中的内容转移到另一个寄存器（也可能就是原来的寄存器）。因为 8080 微处理器设计了 7 个寄存器（包括累加器在内），因此应用中使用大量的 MOV 指令是很正常的。

下面列出了前 32 条 MOV 指令。再一次提醒你，两个参数中左侧的是目标操作数，右侧的是源操作数。

操作码	指　令	操作码	指　令
40	MOV　B, B	50	MOV　D, B
41	MOV　B, C	51	MOV　D, C
42	MOV　B, D	52	MOV　D, D
43	MOV　B, E	53	MOV　D, E
44	MOV　B, H	54	MOV　D, H
45	MOV　B, L	55	MOV　D, L
46	MOV　B, [HL]	56	MOV　D, [HL]
47	MOV　B, A	57	MOV　D, A
48	MOV　C, B	58	MOV　E, B
49	MOV　C, C	59	MOV　E, C
4A	MOV　C, D	5A	MOV　E, D
4B	MOV　C, E	5B	MOV　E, E
4C	MOV　C, H	5C	MOV　E, H
4D	MOV　C, L	5D	MOV　E, L
4E	MOV　C, [HL]	5E	MOV　E, [HL]
4F	MOV　C, A	5F	MOV　E, A

这些指令使用起来非常方便。利用上面的指令可以方便地把一个寄存器存放的数据转移到另一个寄存器。下面让我们研究一下以 HL 寄存器对作为操作数的 4 条指令。

```
MOV B, [HL]
```

前面讲过 LDA 指令，它可以把单字节的操作数从存储器转移到累加器；LDA 操作码后面直接跟着该操作数的 16 位地址。在上面列出的指令中，MOV 指令把字节从存储器转移到 B 寄存器，但该字节的 16 位地址却存放在 HL 寄存器对中。HL 是怎样得到 16 位存储器地址的呢？这并不难解决，有很多方法可以做到，比如通过某种计算实现。

总而言之，对于下面这两条指令：

```
LDA A, [aaaa]
MOV B, [HL]
```

它们的功能都是把一个字节从内存读入微处理器，但它们寻址存储器的方式并不相同。第一种方式称做直接寻址（direct addressing）；第二种方式称做间接寻址（indexed addressing）。

下面列出了其余 32 条 MOV 指令，我们看到 HL 保存的 16 位存储器地址也可以作为