附录 A

PC 硬件

本文介绍供 x86 运行的个人计算机(PC)硬件平台。

PC 是指遵守一定工业标准的计算机，它的目标是使得不同厂家生产的机器都能够运行一定范围内的软件。这些标准随时时间迁移不断变化，因此90年代的 PC 与今日的 PC 看起来已是大不相同。

从外观来看，PC 是一个配置有键盘、屏幕和各种设备的"盒子"。盒子内部则是一块集成电路——*主板*，上面有 CPU 芯片，内存芯片，显卡芯片，I/O 控制器芯片，以及负责芯片间通信的总线。总线会遵守某种标准（如 PCI 或 USB），从而能够兼容不同厂家的设备。

我们可以把 PC 抽象为三部分：CPU、内存和 I/O 设备。CPU 负责计算，内存用于保存计算用的指令和数据，其他设备用于实现存储、通讯等其他功能。

你可以想象主存以一组导线与 CPU 相连接，其中一些是地址线，一些是数据线，还有一些则是控制线。CPU 要从主存读出一个值，需要向地址线上输出一系列表示0和1的电压，并在规定的时间内在"读"线上发出信号1，接下来再从数据线上的高低电压中获取数据。CPU 若要向内存中写入一个值，则向数据线和地址线上写入合适的值，并在规定时间内在 "写" 位上发出信号1.真实的内存接口比这复杂的多，但除非你是在追求高性能，否则你不必考虑这么多的细节。

处理器和内存

CPU（中央处理单元，或处理器）其实只是在执行一个非常简单的循环：从一个被称为『程序计数器』的寄存器中获取一个内存地址，从个地址读出机器指令，增加程序计数器的值，执行机器指令，不断反复。某些机器指令如分支和函数调用会改变程序计数器，如果执行机器指令没有改变程序计数器，这个循环就会从程序计数器开始一条一条地执行指令。

如果不能保存和修改程序数据，那么执行指令就是毫无意义的。速度最快的数据存储器是处理器的寄存器组。一个寄存器是处理器内的一个存储单元，能够保存一个字大小的值（按照机器不同，一个字通常会是16，32或者64位）。寄存器内的值能在一个 CPU 周期内被快速地读写。

PC 处理器实现了 x86 指令集，该指令集由 Intel 发布并成为了一种标准，一些产商生产实现了该指令集的处理器。和其他的 PC 标准一样，这个标准也在不断更新，但是新的标准是向前兼容的。由于 PC 处理器启动时都是模拟1981年 IBM PC 上使用的芯片 Intel 8088，所以 boot loader 需要作出改变以应对标准的更新。但是，对于 xv6 的绝大部分内容，你只需要关心现代 x86 指令集。

现代 x86 提供了8个32位通用寄存器--`%eax, %ebx, %ecx, %edx, %edi, %esi, %ebp, %esp` 和一个程序计数器 `%eip`（*instruction pointer*）。前缀*e*是指*扩展的*（*extended*），表示它们是16位寄存器`%ax, %bx, %cx, %dx, %di, %si, %bp, %sp` 的32位扩展。这两套寄存器其实是相互的别名，例如 `%ax` 是 `%eax` 的低位：我们在写 `%ax` 的时候也会改变 `%eax`，反之亦然。前四个寄存器的两个低8位还有自己的名字：`%al, %ah` 分别表示 `%ax` 的低8位和高8位，`%bl, %bh, %cl, %ch, %dl, %dh` 同理。另外，x86 还有8个80位的浮点寄存器，以及一系列特殊用途的寄存器如*控制寄存器* `%cr0, %cr2, %cr3, %cr4`，*调试寄存器* `%dr0, %dr1, %dr2, %dr3`；段寄存器 `%cs, %ds, %es, %fs, %gs, %ss`；还有全局和局部描述符表的伪寄存器`%gdtr, %ldtr`。控制寄存器和段寄存器对于任何操作系统都是非常重要的。浮点寄存器和调试寄存器则没那么有意思，并且也没有在 xv6 中使用。

寄存器非常快但是也非常昂贵。大多数处理器都会提供至多数十个通用寄存器。下一个层次的存储器是随机存储器（RAM）。主存的速度大概比寄存器慢10到100倍，但要便宜得多，所以容量可以更大。主存较慢的一个原因是它不在处理器芯片上。一个 x86 处理器只有十多个寄存器，但今天的 PC 通常有 GB 级的主存。由于寄存器和主存在读写速度和大小上的巨大差异，大多数处理器，包括 x86，都在芯片上的缓存中保存了最近使用的主存数据。缓存是主存和寄存器在速度和大小上的折衷。现在的 x86 处

理器通常有二级缓存，第一级较小，读写速率接近处理器的时钟周期，第二级较大，读写速率在第一级缓存和主存之间。下表显示了 Intel Core 2 Duo 系统的实际数据：

Intel Core 2 Duo E7200 at 2.53 GHz *备忘：换上真实数字!*

存储器	读写时间	大小
寄存器	0.6ns	64 字节
L1缓存	0.5ns	64K 字节
L2缓存	10ns	4M 字节
主存	100ns	4G 字节

通常 x86 对操作系统隐藏了缓存，所以我们只需要考虑寄存器和主存两种存储器，不用担心主存的层次结构引发的差异。

I/O

处理器必须像和主存交互一样同设备交互。x86 处理提供了特殊的 `in, out` 指令来在设备地址（称为'I/O 端口'） 上读写。这两个指令的硬件实现本质上和读写内存是相同的。早期的 x86 处理器有一条附加的地址线：0表示从 I/O 端口读写，1则表示从主存读写。每个硬件设备会处理它所在 I/O 端口所接收到的读写操作。设备的端口使得软件可以配置设备，检查状态，使用设备；例如，软件可以通过对 I/O 端口的读写，使磁盘接口硬件对磁盘扇区进行读写。

很多计算机体系结构都没有单独的设备访问指令，取而代之的是让设备拥有固定的内存地址，然后通过内存读写实现设备读写。实际上现代 x86 体系结构就在大多数高速设备上（如网络、磁盘、显卡控制器）使用了该技术，叫做 *内存映射 I/O*。但由于向前兼容的原因，`in, out` 指令仍能使用，而比较老的设备如 xv6 中使用的 IDE 磁盘控制器仍使用两个指令。