令，解释指令中的位，执行该指令指示的简单操作，然后更新 PC，使其指向下一条指令，而这条指令并不一定和在内存中刚刚执行的指令相邻。

这样的简单操作并不多，它们围绕着主存、寄存器文件(register file)和算术/逻辑单元(ALU)进行。寄存器文件是一个小的存储设备，由一些单个字长的寄存器组成，每个寄存器都有唯一的名字。ALU 计算新的数据和地址值。下面是一些简单操作的例子，CPU 在指令的要求下可能会执行这些操作。

- 加载：从主存复制一个字节或者一个字到寄存器，以覆盖寄存器原来的内容。
- 存储：从寄存器复制一个字节或者一个字到主存的某个位置，以覆盖这个位置上原来的内容。
- 操作：把两个寄存器的内容复制到 ALU，ALU 对这两个字做算术运算，并将结果存放到一个寄存器中，以覆盖该寄存器中原来的内容。
- 跳转：从指令本身中抽取一个字，并将这个字复制到程序计数器(PC)中，以覆盖 PC 中原来的值。

处理器看上去是它的指令集架构的简单实现，但是实际上现代处理器使用了非常复杂的机制来加速程序的执行。因此，我们将处理器的指令集架构和处理器的微体系结构区分开来：指令集架构描述的是每条机器代码指令的效果；而微体系结构描述的是处理器实际上是如何实现的。在第 3 章研究机器代码时，我们考虑的是机器的指令集架构所提供的抽象性。第 4 章将更详细地介绍处理器实际上是如何实现的。第 5 章用一个模型说明现代处理器是如何工作的，从而能预测和优化机器语言程序的性能。

1.4.2　运行 hello 程序

前面简单描述了系统的硬件组成和操作，现在开始介绍当我们运行示例程序时到底发生了些什么。在这里必须省略很多细节，稍后会做补充，但是现在我们将很满意于这种整体上的描述。

初始时，shell 程序执行它的指令，等待我们输入一个命令。当我们在键盘上输入字符串“./hello”后，shell 程序将字符逐一读入寄存器，再把它存放到内存中，如图 1-5 所示。

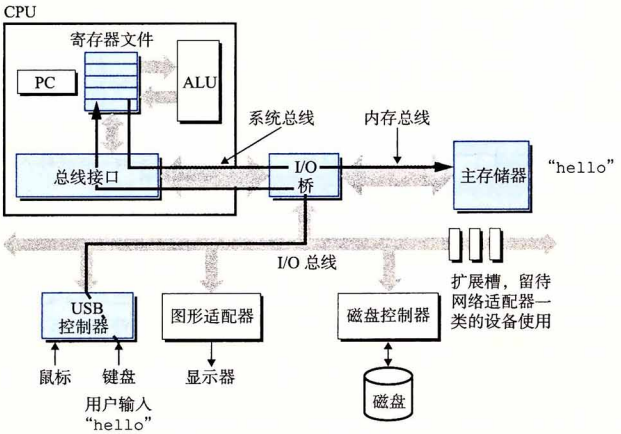

图 1-5　从键盘上读取 hello 命令