冯诺依曼体系结构

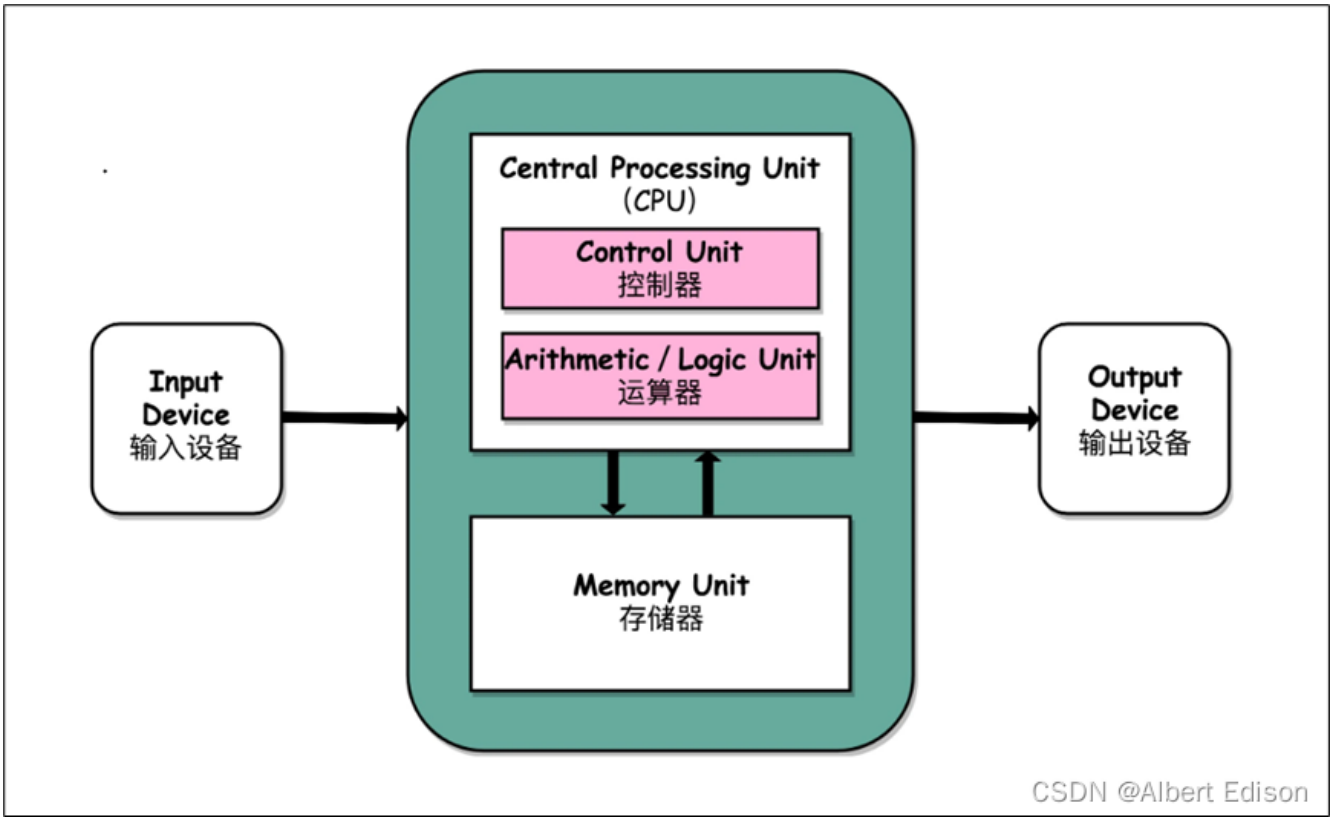
原文链接

文章目录

1.冯诺依曼体系结构

冯 • 诺伊曼结构，也称冯 • 诺伊曼模型或普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的电脑设计概念结构。本结构隐约指导了将存储设备与中央处理器分开的概念，因此依本结构设计出的计算机又称存储程序计算机。

我们常见的计算机，如笔记本；我们不常见的计算机，如服务器，大部分都遵守冯诺依曼体系。如下图所示：



截至目前，我们所认识的计算机，都是由一个个的硬件组件组成，那么主要可以分为四大类：

- 输入设备
- 输出设备
- 存储器（内存）
- 运算器 && 控制器（中央处理器，也被称为 CPU）

输入和输出设备

输入设备向计算机输入数据，计算机经过计算后，把数据输出给输出设备。

常见的输入和输出设备有：

- 输入设备：键盘、磁盘、网卡、显卡、话筒、摄像头、鼠标、扫描仪等等
- 输出设备：显示器、磁盘、网卡、显卡、音响、打印机等等

注意： 同种设备在不同场景下可能属于输入设备，也可能属于输入设备。

中央处理器

中央处理器也就是我们常说的 CPU，它是由运算器和控制器组成。

CPU 内部还有一些组件，常见的有**寄存器**、**控制单元**和**逻辑运算单元**等。其中，控制单元负责控制 CPU 工作，逻辑运算单元负责计算，而寄存器可以分为多种类，每种寄存器的功能又不尽相同。

CPU 中的寄存器主要作用是存储计算时的数据，你可能好奇为什么有了内存还需要寄存器？原因很简单，因为内存离 CPU 太远了，而寄存器就在 CPU 里，还紧挨着控制单元和逻辑运算单元，自然计算时速度会很快。

常见的寄存器种类：

- 通用寄存器，用来存放需要进行运算的数据，比如需要进行加和运算的两个数据。
- 程序计数器，用来存储 CPU 要执行下一条指令「所在的内存地址」，注意不是存储了下一条要执行的指令，此时指令还在内存中，程序计数器只是存储了下一条指令「的地址」。
- 指令寄存器，用来存放当前正在执行的指令，也就是指令本身，指令被执行完成之前，指令都存储在这里。

内存

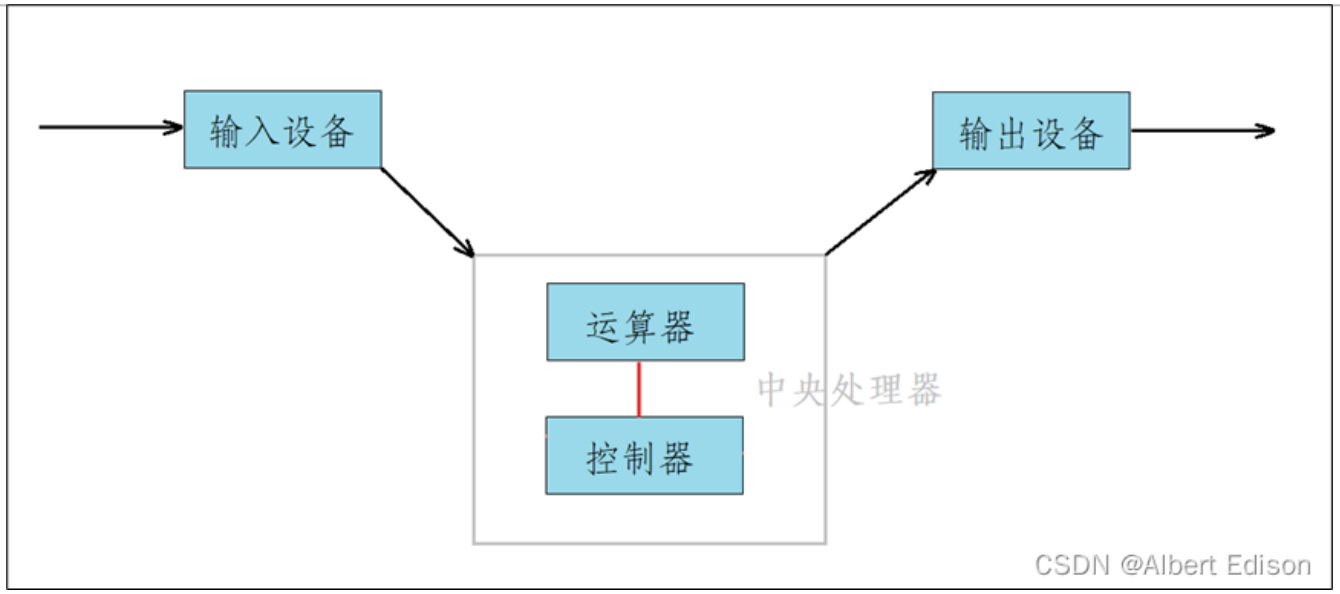
内存，也就是所谓的存储器。

我们的程序和数据都是存储在内存，存储的区域是线性的。

在计算机数据存储中，存储数据的基本单位是字节（byte），1 字节等于 8 位（8 bit）。每一个字节都对应一个内存地址。

内存的地址是从 0 开始编号的，然后自增排列，最后一个地址为内存总字节数减 1，这种结构好似我们程序里的数组，所以内存的读写任何一个数据的速度都是一样的。

思考一个问题：当我们的体系结构中，有了输入、输出设备和 CPU 以后，就能正常工作了，那么为什么还需要内存呢？



（1）从技术角度来说

CPU的运算速度 > 寄存器的速度 > L1~L3Cache > 内存 > 外设（磁盘） > 光盘磁带

也就是说，输入设备和输出设备相对于 CPU 来说是非常慢的。

如果没有内存的话，那么当前这个体系整体呈现出来的就是：输入设备和输出设备很慢，而 CPU 很快。

相信大家知道**木桶原理**吧，那么最终整个体系结构所呈现出来的速度将会是很慢的。

所以，从数据角度出发，外设几乎不和 CPU 打交道，它是直接和内存打交道，CPU 也同样如此。

进言之，内存在我们看来，就是体系结构的一个大的缓存，用来适配外设和 CPU 速度不均的问题！

（2）从成本角度来说

既然上面说了内存是用来适配外设和 CPU 速度不均的问题，那么为什么不直接在 CPU 里面开发一个类似于内存的东西呢？

这个想法可以，但是如果真要去实现的话，那么一台计算机的成本起码得 10W+，而计算机它是蔓延全世界的，也就是说人人都能用得起的！

寄存器的价格 > 内存 > 外设(磁盘)

所以内存就是方便我们使用较低的成本，获得较高的性能。

总线

总线是用于 CPU 和内存以及其他设备之间的通信，总线可分为 3 种：

- 地址总线，用于指定 CPU 将要操作的内存地址；
- 数据总线，用于读写内存的数据；
- 控制总线，用于发送和接收信号，比如中断、设备复位等信号，CPU 收到信号后自然进行响应，这时也需要控制总线；

当 CPU 要读写内存数据的时候，一般需要通过下面这三个总线：

- 首先要通过「地址总线」来指定内存的地址；
- 然后通过「控制总线」控制是读或写命令；
- 最后通过「数据总线」来传输数据；

局部性原理

我相信大家应该还有个疑惑：就是，先将输入设备的数据交给内存，再由内存将数据交给 CPU，这个过程真的比 CPU 直接从输入设备获取数据更快吗？

说明这个问题之前，我们首先需要知道：内存具有数据存储的能力。**虽然内存的大小只有 4G/8G，但是既然内存有大小，那么它就有预装数据的能力，而这就是提高该体系结构效率的秘诀。**

这里不得不说到的就是**局部性原理**：根据统计学原理，当一个数据正在被访问时，那么下一次有很大可能会访问其周围的数据。所以当 CPU 需要获取某一行数据时，内存可以将该行数据之后的数据一同加载进来，而 CPU 处理数据和内存加载数据是可以同时进行的，这样下次 CPU 就可以直接从内存当中获取数据。

输出数据的时候也一样，CPU 处理完数据后直接将数据放到内存当中，当输出设备需要时再在内存当中获取即可，这也就有了我们平常所说的缓冲区的概念。

例如，缓冲区满了才将数据打印到屏幕上，使用 fflush 函数将缓冲区当中的数据直接输出之类的，都是将内存当中的数据直接拿到输出设备当中进行显示输出。

总结

冯 • 诺依曼体系结构核心原理为：用户输入的数据先放到内存当中，CPU 读取数据的时候就直接从内存当中读取，CPU 处理完数据后又写回内存当中，然后内存再将数据输出到输出设备当中，最后由输出设备进行输出显示。

我们可以知道，站在硬件角度或是数据层面上，CPU 和外设不能直接交互，而是通过内存，也就是说，所有设备都只能和内存打交道。

由此可以说明一个问题：为什么程序运行之前必须先加载到内存？

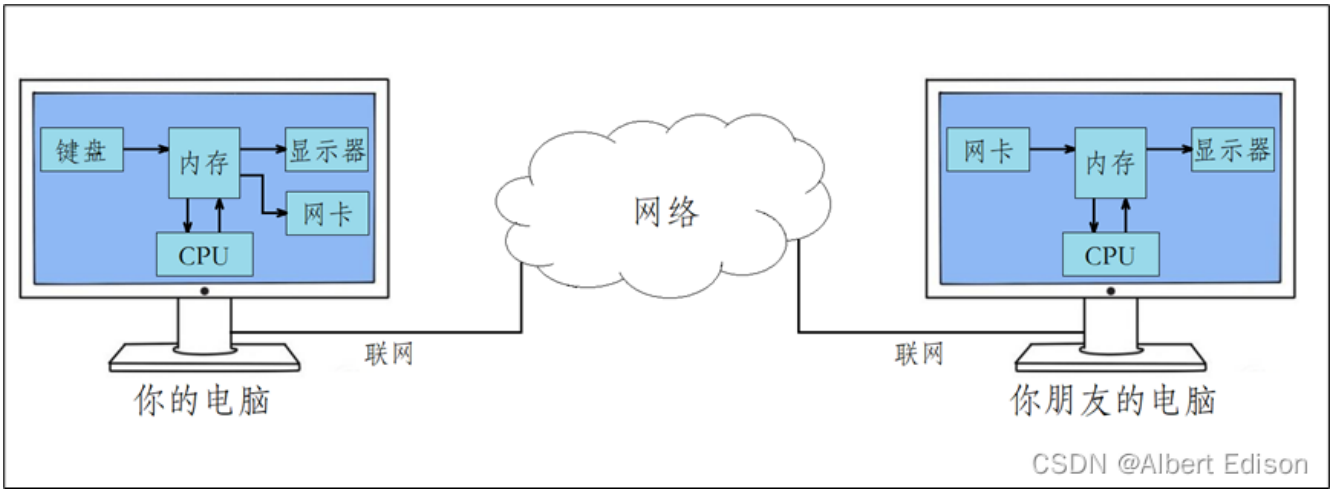
因为可执行程序（文件）是在硬盘（外设）上的，而 CPU 只能从内存当中获取数据，所以必须先将硬盘上的数据加载到内存，也就是必须先将程序加载到内存。

2． 数据的流动过程

对冯诺依曼的理解，不能停留在概念上，要深入到对软件数据流理解上。

从你登录上 QQ 和某位朋友聊天开始，数据的流动过程是怎样的呢？从你打开窗口，开始给他发消息，到他的到消息之后的数据流动过程。

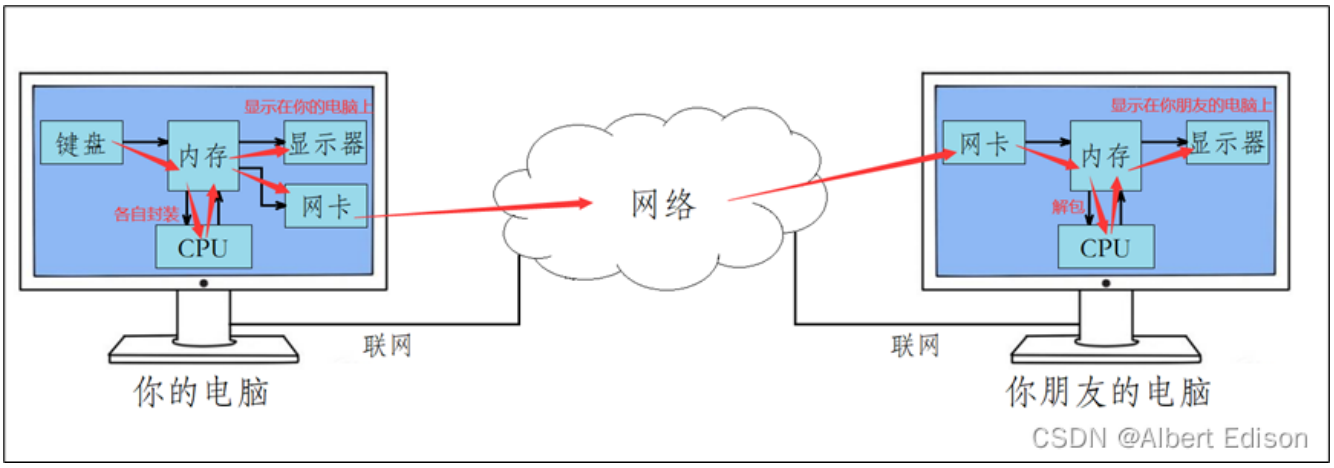
要使用 QQ，首先需要联网，而你和你的朋友的电脑都是冯诺依曼体系结构，在你向朋友发送消息这个过程中，你的电脑当中的键盘充当输入设备，显示器和网卡充当输出设备，你朋友的电脑当中的网卡充当输入设备，显示器充当输出设备。



刚开始你在键盘当中输入消息，键盘将消息加载到内存，此时你的显示器就可以从内存获取消息进而显示在你自己的显示器上，此时你就能在你自己的电脑上看到你所发的消息了。

在键盘将消息加载到内存后，CPU 从内存获取到消息后对消息进行各种封装，然后再将其写回内存，此时你的网卡就可以从内存获取已经封装好的消息，然后在网络当中经过一系列处理（这里忽略网络处理细节）。

之后你朋友的网卡从网络当中获取到你所发的消息后，将该消息加载到内存当中，你朋友的 CPU 再从内存当中获取消息并对消息进行解包操作，然后将解包好的消息写回内存，最后你朋友的显示器从内存当中获取消息并显示在他的电脑上。



那么如果是在 QQ 上发送文件呢?

首先你的文件最开始是在你本地的磁盘上的，先从磁盘上把文件读到内存中，文件里面的东西其实还是数据，把数据再经过 CPU 封装成报文，然后刷新到我们的内存中，定期再经过网卡，把数据刷新到网卡上，然后再发出去。

传文件的本质就是：两端的磁盘进行通信。

3. 总结

关于冯诺依曼，必须强调几点：

- 这里的存储器指的是内存
- 不考虑缓存情况，这里的 CPU 能且只能对内存进行读写，不能访问外设（输入或输出设备）
- 外设（输入或输出设备）要输入或者输出数据，也只能写入内存或者从内存中读取。

总结：任何外设，在数据层面，基本优先和内存打交道！CPU 在数据层面上，也直接和内存打交道！不管软件怎么写，操作系统怎么做，本质都脱离不开这样的硬件特性！也就是说：操作系统、CPU、各种各样的软硬件等等，都是围绕冯诺依曼体系结构展开的，而操作系统本身就是一款加载到内存中的软件罢了。