2020/12/24_16位汇编_第1课_汇编语言简介、计算机的发展及组成

笔记本: 16位汇编
创建时间: 2020/12/24 星期四 10:28
作者: ileemi

什么是汇编语言

汇编语言早期称为**助记符语言**，原始编程：将二进制代码在纸带、上进行打孔后，机器在进行读取。

> 汇编语言（Assembly Language）是任何一种用于电子计算机、微处理器、微控制器或其他可编程器件的低级语言，亦称为符号语言。在汇编语言中，用助记符代替机器指令的操作码，用地址符号或标号代替指令或操作数的地址。在不同的设备中，汇编语言对应着不同的机器语言指令集，通过汇编过程转换成机器指令。特定的汇编语言和特定的机器语言指令集是一一对应的，不同平台之间不可直接移植。

机器码语言编程：

- 前0001 后0010 左0011 右0100 ...
- 0001 0001 0010 0010 0011 0011 0100 0100

助记符语言编程（类似于使用一个单词代表一个功能（一串二进制））：

- 0001 left -> LEF
- 0010 right -> RHT
- 0011 front -> FRT
- 0100 back -> BCK
- left left right right front front --> 不够简洁
- LEF LEF RHT RHT FRT FRT BCK BCK --> 定义一个规范，使指令更加整洁，可读性高使用简写

早期可以认为，每一种硬件有一种汇编语言（内部功能实现，功能代码风格等由厂商决定，也就是汇编指令不同，电脑硬件，由 "CPU" 厂商提供）。

"CPU" 不认识汇编指令，可以通过 "翻译器" 将汇编指令翻译成 "CPU" 认识的二进制代码（翻译器：由硬件厂商提供）。

反翻译器：将二进制代码转换成对应的汇编指令

- LEF LEF RHT RHT FRT FRT BCK BCK ==》翻译器 ==》0001 0001 0010 0010 0011 0011 0100 0100
- LEF LEF RHT RHT FRT FRT BCK BCK 《== 反翻译器《== 0001 0001 0010 0010 0011 0011 0100 0100

汇编语言执行效率在一定程度上比高级语言的执行效率要高

汇编语言的缺点：通用性差（同样的功能需要重复编写，硬件不同）
高级语言：通用性较强，定义的语言需要适用于不同的机器，只适合定义共性的语法（高级语言就是解决通用性的）。

高级语言通过翻译器生成对应公司指定程序的汇编代码，之后在使用这家公司指定的翻译器将汇编指令翻译成对应的二进制代码。

位数设计一般为2的多次方：2 4 8 16 32 等

Windows 上主流的汇编为：x86（8086 -- 16位CPU）汇编，每个CPU都有自己的指令集。

高级语言会降低效率解决问题，存在效率低的问题（软件运行速度上），现在逆向就是看汇编，将汇编语言还原成对应的高级语言。

现在汇编还没有淘汰：

- 游戏上的优化：核心代码由汇编编写
- 懂硬件才能更好的用好汇编

汇编语言的主要特点：

- 汇编语言程序与处理器指令系统密切相关
- 程序员可直接、有效地控制系统硬件
- 形成的可执行文件运行速度快、占用主存容量少

汇编语言和高级语言：

- 汇编语言与处理其密切相关，汇编语言程序的通用性、可移植性较差。汇编语言功能有限、设计硬件细节，编写程序比较繁琐，调试起来也比较困难。汇编语言本质上就是机器语言，可以直接、有效地控制计算机硬件，易于产生速度快、容量小的高效率目标程序。
- 高级语言与具体计算机无关，高级语言程序可以在多中计算机上编译后执行。高级语言提供了强大的功能，不必关心琐碎的问题，类似自然语言的语法，易于掌握和应用。高级语言不针对具体计算机系统，不易直接控制计算机的各种操作，目标程序比较庞大、运行速度较慢。

脚本语言在机器上其汇编指令已经确定，可通过翻译器直接翻译成二进制。

汇编语言和高级语言的优缺点

汇编语言的优点：

- 直接控制计算机硬件部件
- 可以编写在 "时间" 和 "空间" 两方面最有效的程序

汇编语言的缺点：

- 与处理器密切有关
- 需要熟悉计算机硬件系统、考虑许多细节
- 编写繁琐，调试、维护、交流和移植困难

> 汇编语言的优点使得它在程序设计中占有重要的位置，是不可被取代的。汇编语言的缺点使得人们主要采用高级语言进行程序开发工作。有时需要采用高级语言和汇编语言混合编程的方法，互相取长补短，更好地解决实际问题。

> 汇编语言是学习逆向的基础。

微机系统的组成

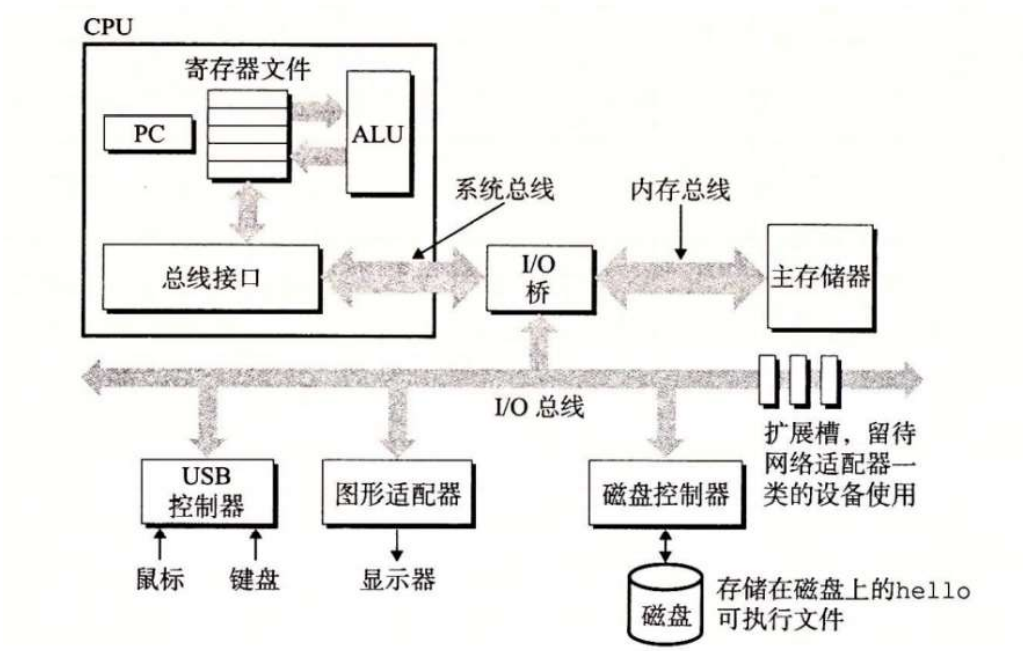
硬件：

- 控制器、运算器（CPU）
- 存储器（硬盘、内存）
- 输入设备、输出设备（键盘、鼠标、显示器、打印机等）

软件：

- 系统软件
- 应用软件

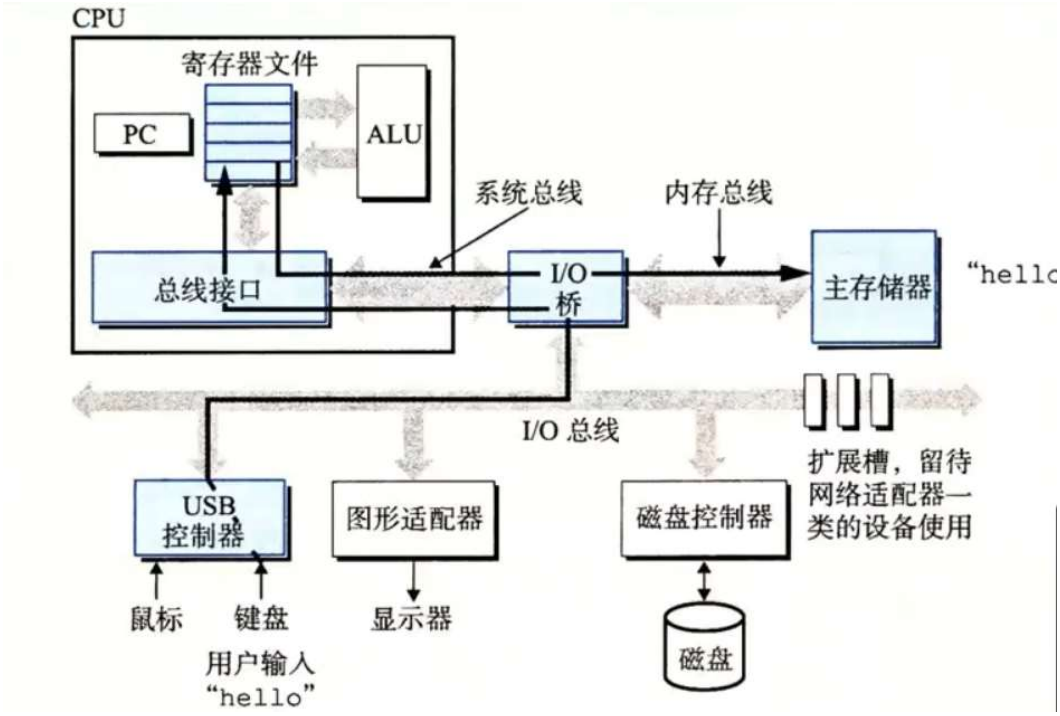
微机系统组成图



- I/O 总线（等价与主板）：连接CPU和各个硬件（CPU通过 "I/O桥(南北桥芯片") 和 "I/O总线" 进行连接的）
- 主存储器：内存（内存的缺点：存在易失性（存在断电可能），优点：读写数据速度快），存储空间一般不大
- 磁盘：优点：非易失性，缺点：读写数据速度慢
- I/O桥（南北桥芯片）：负责 "I/O" 总线之间的通讯，解决通讯效率问题（提高通信性能而缩短传输距离）
- 扩展插槽（PCI）：提供网卡，声卡的接口（网络适配器一类的设配使用）
- 磁盘控制器：将硬盘中的数据直接传入到内存，内存中的数据不在需要通过寄存器。

> CPU 从内存中读取数据，需要哪些数据就将磁盘（程序也在磁盘中）中的数据放入到内存中供 CPU 进行读取。

早期微机系统运行程序流程图



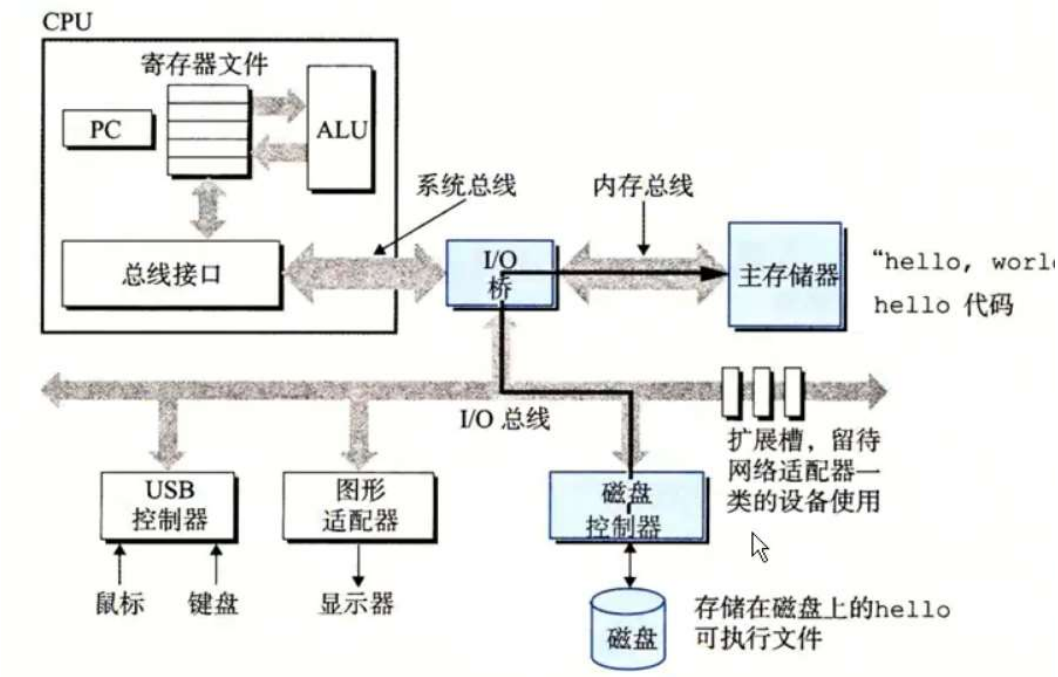
> 键盘内部有一个键盘存储器，由 CPU 进行读取，将键盘存储器中的字母发送到寄存器中，之后在从寄存器发送到内存中。硬盘中的数据也是如此（通过寄存器进行中转），寄存器的存储速度要比内存存储速度快100倍还要多。

寄存器的大小按照字节计算的，16位CPU，寄存器的大小为 2 字节，32位CPU，寄存器的大小为 4 字节。

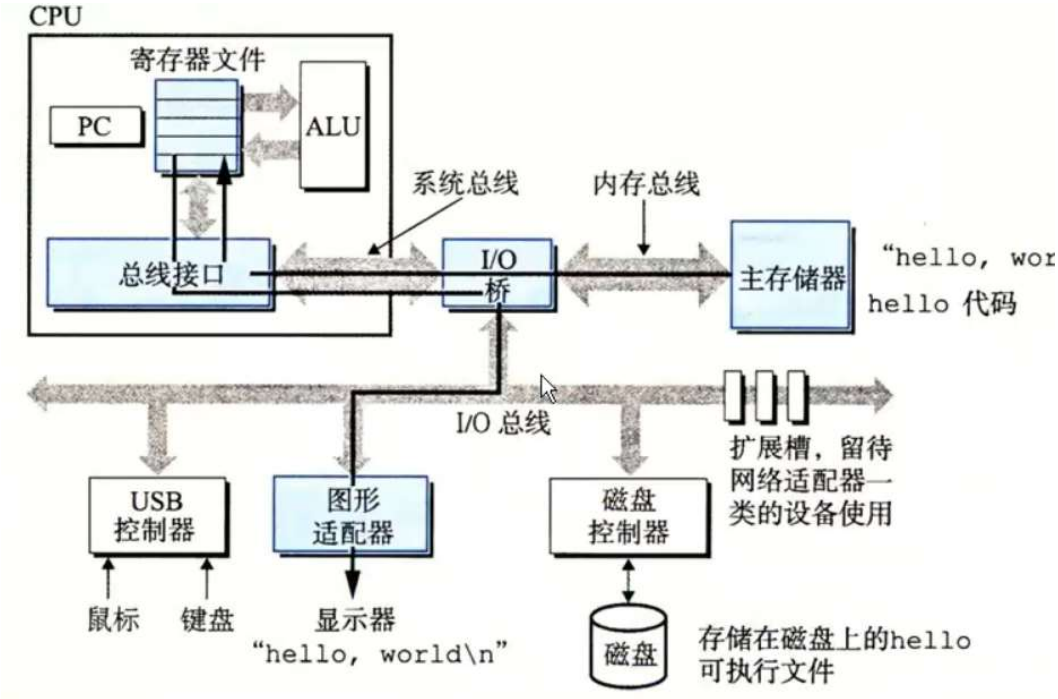
CPU 内部的运算器只和寄存器进行通信，运算器没有内存概念。C语言不可以操作寄存器。寄存器的数据多来源于内存。

通过汇编指令可以控制寄存器，通过二进制告诉 CPU 要操作哪些硬件，怎么传输数据。

通过磁盘控制器将所需的数据直接发送到内存



执行程序

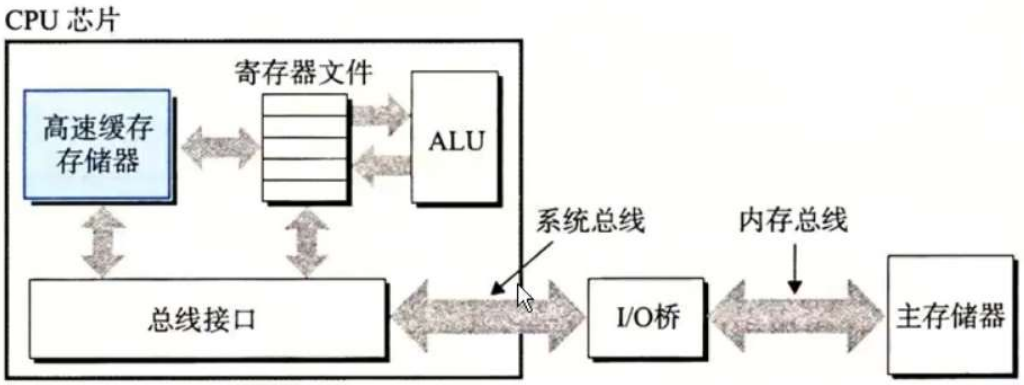


二进制代码从内存到寄存器，在通过运算器执行，将执行的结果该显示就显示，该存储就进行存储。CPU 会自动解释二进制。

程序计数器（PC）：用于存放下一条指令所在单元的地址的地方（不能存放数据）。

高速缓存

> 提供内存数据访问效率



缓存的字节单位：kb

4 x 32：

- 4核心的CPU其有四个高速缓存器，每个高速缓存器（32kb）都有对应的寄存器和运算器。

访问内存中的变量时，优先访问内存中相邻的两个变量（这样效率更高）。频繁切换线程会降低效率。

缓存级别越低，效率越快。三级缓存是所有 CPU 共享的。在汇编中控制寄存器和缓存对程序的运行速度有很大的影响（对硬件了解越深，写的程序效率就会越高效）。

存储器缓存结构

当今的计算机系统当中，基本上全部都置入了各种各样的存储设备（ Cache、内存、HDD(SSD) 硬盘），这些存储设备呈明显的层次结构， 它们的特点是容量越大，速度越慢，越靠近 CPU 速度越快，容量越小，价格越贵。按照容量和速度将它们以图示的方式呈现的话，则看起来 就像是一个金字塔，如下图所示：

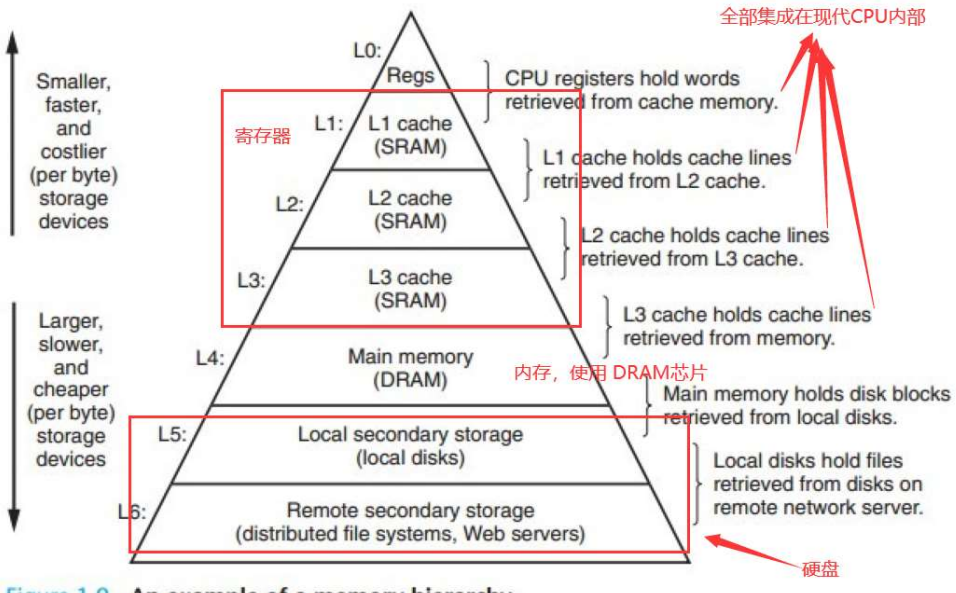


Figure 1.9 **An example of a memory hierarchy.**

- 寄存器（Register）：寄存器与其说是存储器，其实更像是 CPU 本身的一部分，只能存放极其有限的信息，但是速度非常快，和 CPU 同步。
- 高速缓存（CPU Cache）：使用 SRAM（Static Random-Access Memory，静态随机存取存储器）的芯片。

- 内存（DRAM）：使用 DRAM（Dynamic Random Access Memory，动态随机存取存储器）的芯片，比起 SRAM 来说，它的密度更高，有更大的容量，而且它也比 SRAM 芯片便宜不少。
- 硬盘：如 SSD(Solid-state drive 或 Solid-state disk，固态硬盘)、HDD(Hard Disk Drive，硬盘)。

不同层次存储器设备特点：

- 越靠近 CPU 速度越快，容量越小，价格越贵
- 每一种存储器设备只和它相邻的存储设备打交道。比如，CPU cache 是从内存里加载而来的，或者需要写回内存，并不会直接写回数据到硬盘，也不会直接从硬盘加载数据到 CPU cache 中，而是先加载到内存，再从内存加载到 cache 中。