什么是汇编语言？

CPU只负责计算，本身不具备智能。你输入一条指令，他就运行一次，然后停下来，等待下一次指令。

这些指令都是二进制的，称为操作码（opcode），比如加法指令就是00000011.

编译器的作用就是，把高级语言写好的程序，翻译成一条条的操作码。

因为二进制的语言是不可读的，所以，产生了汇编语言。

汇编语言是二进制指令的文本形式，与二进制指令是一一对应的关系。比如，加法指令00000011写成汇编语言就是ADD。只要还原成二进制，汇编语言就可以被CPU直接执行，所以它是最底层的低级语言。

汇编语言的来历：

最早，编写程序就是手写指令，然后通过各种开关输入计算机。

后来，发明了纸带打孔机，将二进制指令自动输入计算机。

为了解决可读性问题，将二进制改为八进制，然而八进制可读性也不行，最后变成用文字表达，内存地址也不再直接引用，而是用标签表示。

将文字指令翻译成二进制步骤称为assembling，完成这个步骤的程序就叫做assembler、。他处理的文本就叫做assembly code。标准化以后，称为assembly language，缩写为asm，中文译为汇编语言。

每一种CPU的机器指令都是不一样的，因此，对应的汇编语言也不一样。

寄存器：

CPU本身只负责计算，不负责存储数据。数据一般都是存储在内存中，CPU要用的时候就会去内存读写数据。但是CPU的运算速度要远高于内存的读写速度，为了提高效率，CPU都自带一级缓存和二级缓存。基本上，CPU缓存可以看作是读写较快的内存。

但是，CPU缓存还是不够快，另外，数据在缓存中的地址是不固定的，CPU每次读写都要寻址，会拖慢速度。因此，CPU还自带了寄存器，用来存储最常用的数据。也就是说，读写最频繁的数据（循环变量）会被存储在寄存器中，CPU优先读写寄存器，再由寄存器跟内存交换数据。

寄存器不依靠地址区分数据，而是依靠名称。每一个寄存器都有自己的名称，我们告诉CPU去具体的哪一个寄存器拿数据，这样的速度是最快的。

为什么寄存器比内存快？

计算机的存储层次之中，寄存器最快，内存次之，最后是硬盘。

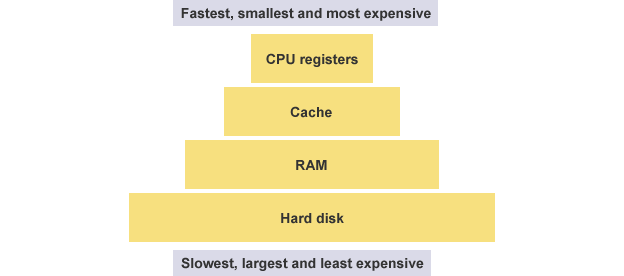
为什么寄存器比内存快？

距离不同（不是主要因素），内存离CPU比较远

设计不同，内存的设计相对简单，就是一个电容加上一个晶体管，而寄存器要多几个电子元件。另外，通电以后，寄存器的晶体管一直有电，而内存的晶体管只有用到的才有电，没用到的就没电，这样有利于省电。这些设计上的因素决定了寄存器读取速度比内存要快。

工作方式不同：寄存器的工作方式：找到相关的位，并读取这些位。

内存的工作方式：找到数据的指针（指针可能放在寄存器内，所以这一步就包括了寄存器的所有工作了），将指针送往内存管理单元（MMU），由MMU将虚拟的内存地址翻译成实际的物理地址，将物理地址送往内存的控制器，由内存的控制器找出该地址应该插在哪一根内存插槽上，确定数据在哪一个内存块上，从该块读取数据，最后，数据先送回内存控制器，再送回CPU，然后开始使用。



寄存器的种类：

早期的 x86 CPU 只有8个寄存器，而且每个都有不同的用途。现在的寄存器已经有100多个了，都变成通用寄存器，不特别指定用途了，但是早期寄存器的名字都被保存了下来。

八个寄存器分别是：EAX、EBX、ECX、EDX、EDI、ESI、EBP、ESP

前七个都是通用的，ESP寄存器有特定用途，保存当前Stack的地址。

常常看到的32位CPU、64位CPU这样的名称，其实就是寄存器的大小。32位CPU寄存器的大小就是4个字节。

内存模型：Heap（堆）

寄存器只能放少量数据，大多数时候，CPU要指挥寄存器，直接跟内存交换数据。

程序运行的时候，操作系统会给它分配一段内存，用来存储程序和运行时产生的数据。这段内存有起始地址和结束地址，比如从0x1000到0x8000，起始地址是比较小的那个地址，结束地址是比较大的那个地址。程序运行过程中，对于动态的内存占用请求（比如新建对象、或者使用macllo命令），系统就会从预先分配好的那段内存之中，划出一部分给用户，具体规则是从起始地址开始划分。举例来说，用户要求得到10个字节内存，那么起始地址0x100开始分配，一直分配到地址0x100A。

这些因为用户主动请求而划分出来的内存区域，叫做堆。它由起始地址开始，从低位向高位增长。Heap的一个重要特点就是不会自动消失，必须手动释放，或者由垃圾回收机制回收。