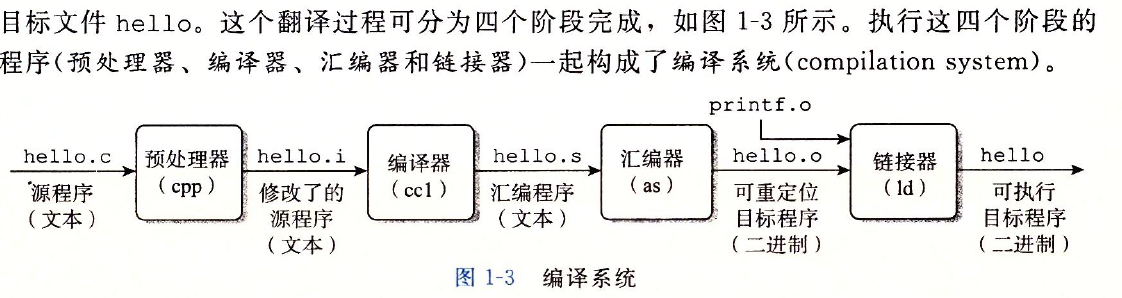
Compilation system:programs which executing 4 phase.

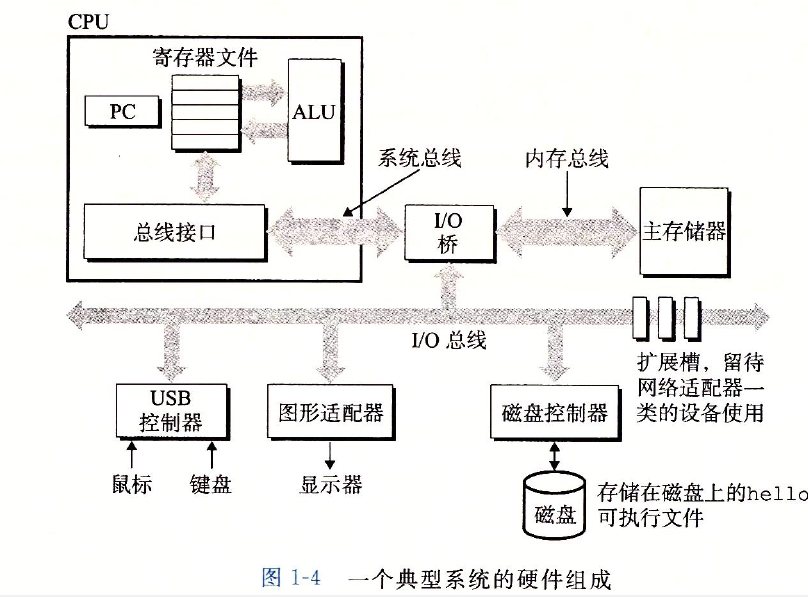
Cpp:还是C

Ccl:.c>.s----------C到汇编

As:汇编到二进制

Ld:二进制文件的组合





主存:内存DRAM芯片----处理器执行程序时存放程序和程序处理的数据.

CPU:核心由指令集架构(区别实际的微体系结构)+一个字大小的寄存器(PC)

PC执行程序,自然PC指向的是主存的逻辑地址.

Registe file:一组寄存器设备

ALU:能进行算术运算和逻辑运算的结构单元.

CPU在指令集架构下的元操作:

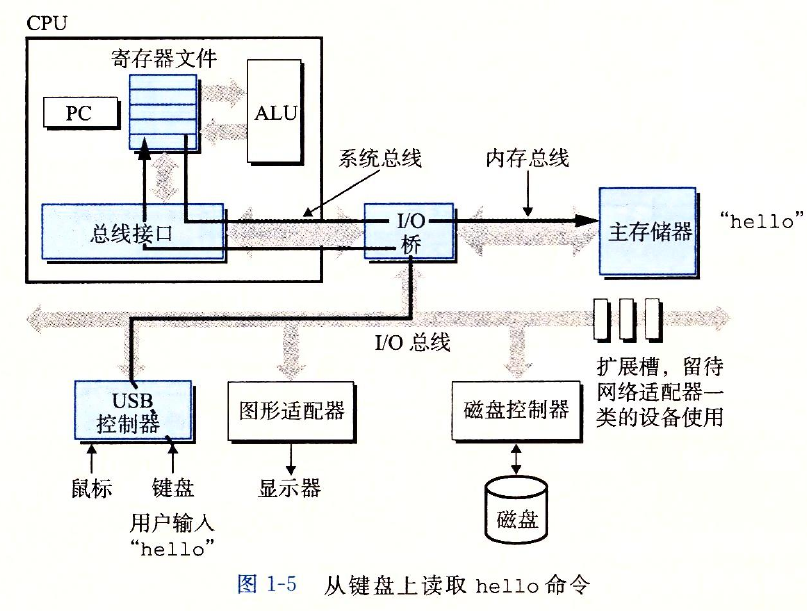
加载:主存->register file

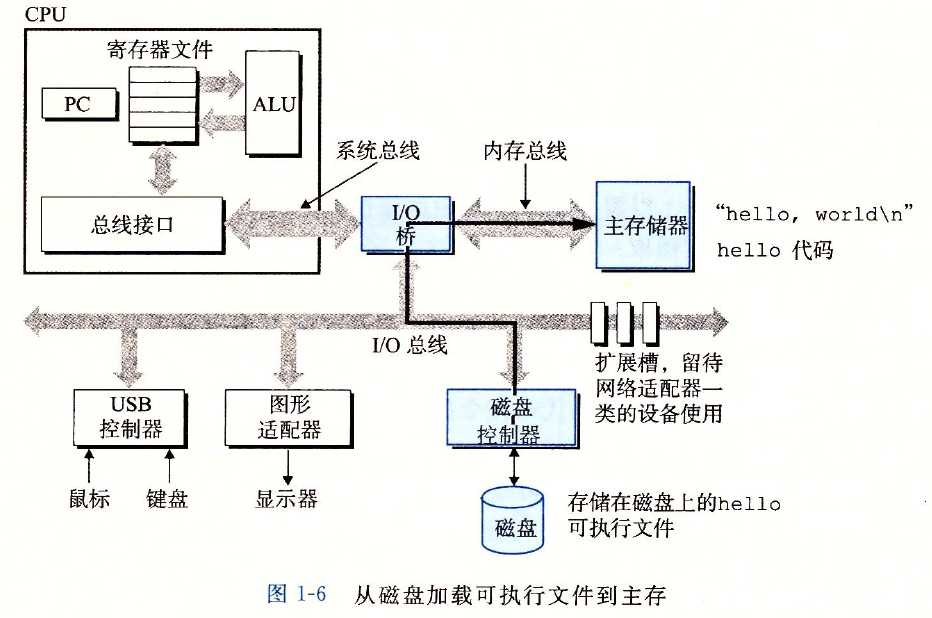
存储:register file->主存

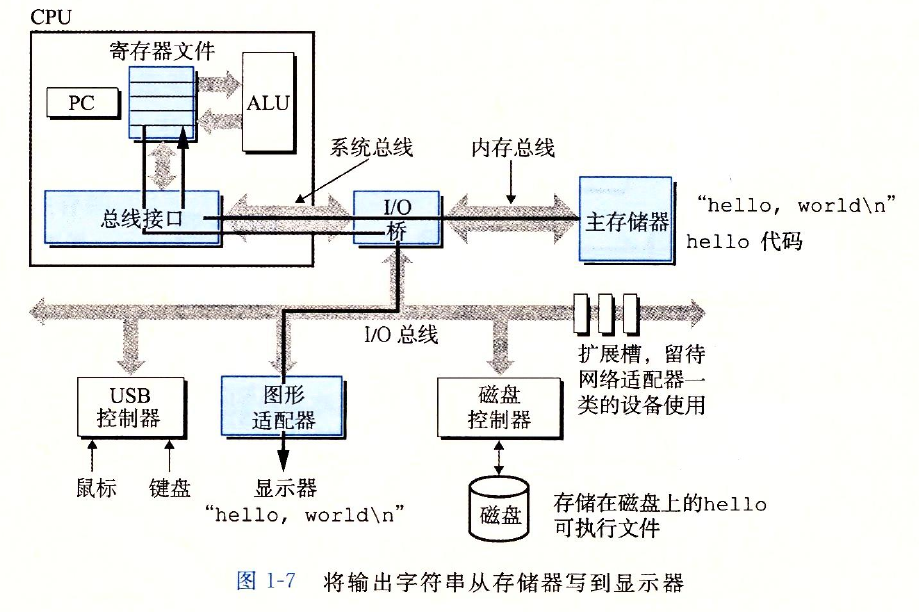
操作:register file->ALU->register file.

跳转:register file->PC

实例:







一些问题:

机械原理表面:存储设备大小与数据传输快慢成反比;

磁盘大1000倍,慢1000万倍(与主存比)

主存大几十亿/几百字节=1000万倍,慢100倍.(与register file相比)

趋势是处理器更容易快,主存更容易大,他们的存储大小和数据传输快慢差距将越来越大.

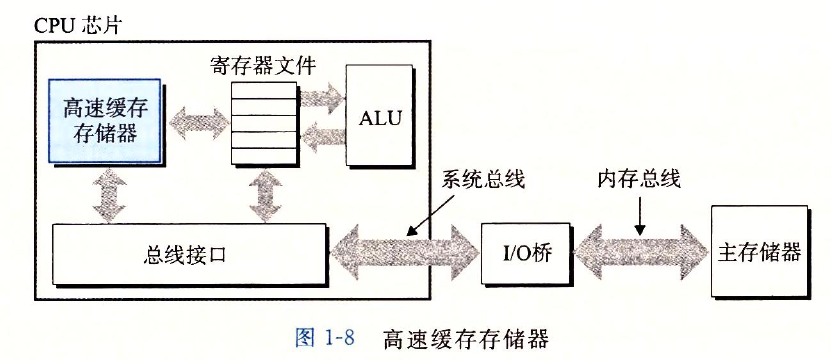
解决办法:

更小更快的cache memory:大一百倍(几万字节),慢一丢丢

结合概率论:高速缓存的局部性原理(程序具有访问局部区域里的数据和代码的趋势)

L2数十万/数百万字节,进程访问L2要比L1慢5~10倍.

有些甚至有L3.



举例说明高速缓存的作用:

主存到CPU的100个字节(a-z0-9共100个,数字占大头有74个)传输,每个要1秒,共100秒

主存到缓存每个要1s,,我们让缓存只能存10个字节,就是0-9,那么主存到CPU传输字母要26秒,主存到缓存传输数字要10秒,缓存到CPU时间贼快可忽略不计,总计36秒.36<100.

存储设备都形成存储结构实际上:

