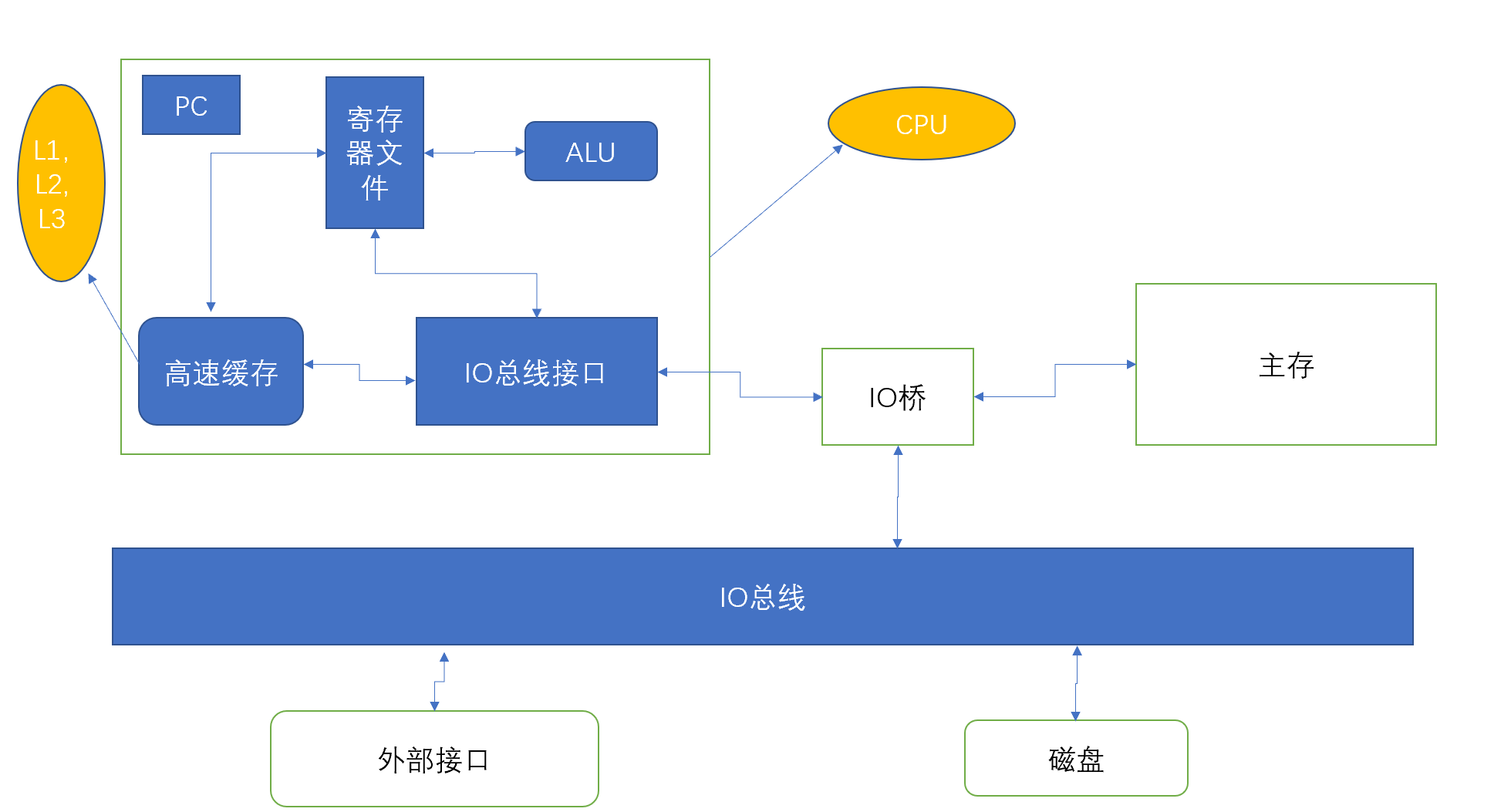
第一章：计算机系统漫游

源程序文件的处理流程：

1. 预处理器（cpp）:根据#开头的命令修改原始的.c文件，生成.i文件
2. 编译器（ccl）：将.i文件转换成包含汇编语言的.s文件
3. 汇编器（as）：将.s文件转换成二进制的机器语言(.o文件)
4. 链接器（ld）：合并程序中引用到的一些外部的文件库，生成可执行的目标程序



计算机系统的硬件模型如上图所示CPU模块包含了PC，寄存器文件，ALU,高速缓存这些模块，通过总线接口和外部进行通信。

PC：程序计数器，总是指向下一行指令的内存地址（类似于java虚拟机中的程序计数器）

寄存器文件：小型的存储设备，存取速度非常快，用于存放临时数据（类似于java虚拟机中的临时变量表的概念）

ALU：逻辑运算单元，运算速度非常快（类似于java虚拟机的栈的功能）

高速缓存：内存比寄存器大几个数量级，但是存取速度比寄存器慢5到10倍

主存：存放数据的模块，用来存储执行进程的数据（有点类似java的堆的概念）

磁盘：存放持久化的数据

IO总线：传输数据的通道

外部接口：接一些外部设备，比如鼠标，U盘，显示器，键盘等等

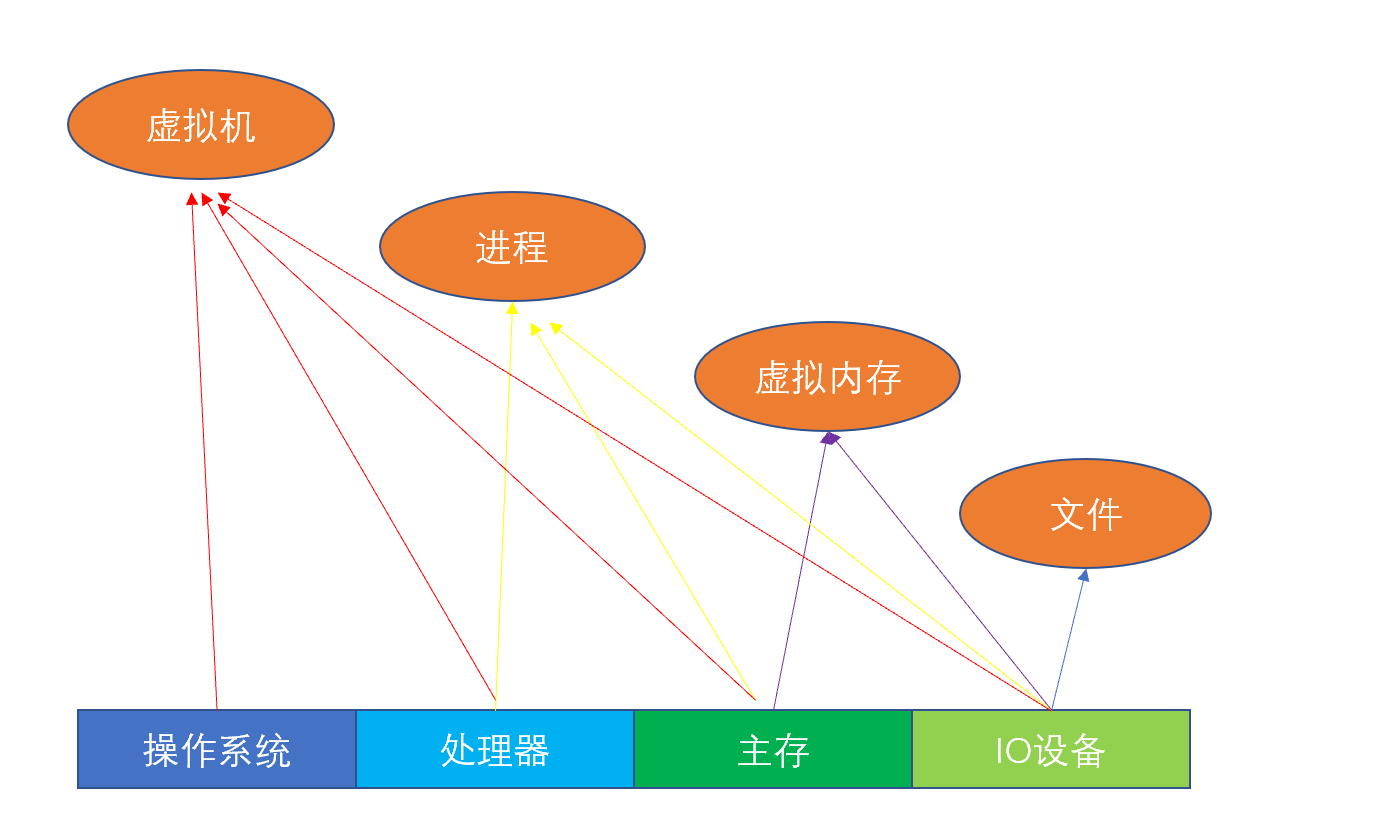
Amdahl定律

说明在做程序优化时，应当更加注重程序的主要耗时部分的性能优化

计算加速比公式为：

S = Told/Tnew = 1/[(1 -a) +a/k](a为优化前该部分耗时所占的比重，k为该部分性能提升的比例，即优化前该部分耗时和优化后该部分耗时的比值)

计算机系统的抽象



本章存疑：

1.对一些概念还是比较模糊，比如指令级并行部分，还有虚拟内存的实现

2.关于多线程还有一个疑惑，一个CPU是不是在一个时间点只能执行一个程序？书上说有的处理器可以让每个核执行两个程序。这和我理解的有偏差。

3.为什么二进制程序在不同计算机上面是不兼容的，但是64位计算机系统可以运行32位计算机编译出来的程序？

第二章：信息的表示和处理

1.二进制，十进制，十六进制之间的转化

2.无符号数和有符号数的二进制表示（无符号数二进制的第一位表示正负），一个有符号数和一个无符号数进行比较或者计算时，会强制转换成无符号数之间的运算

3.指针的声明，它指向目标对象的首地址，类似于数组的第一个元素

4.大端法和小端法（小端法为主流）

5.布尔代数：与，或，异或，反

6.位移运算，逻辑右移和算数右移（算数右移是带符号的，取二进制的最高位填充移位后的高位）和java有所区别。当位移比较大时，会通过取模来进行位移

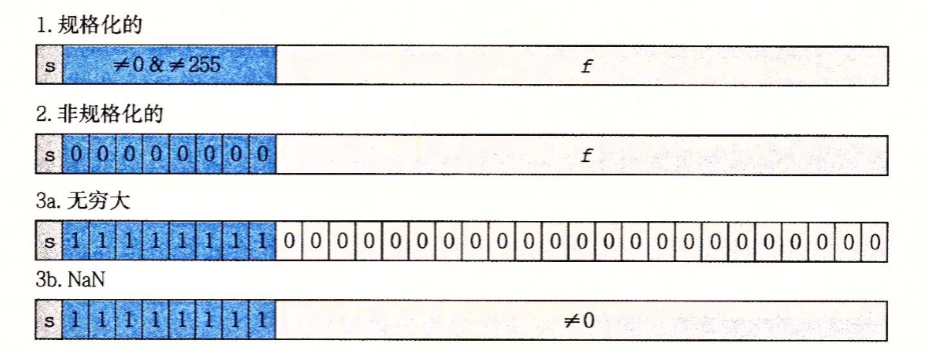
7.整数的二进制表示。绝大部分计算机用补码的编码表示负数，补码的取值范围与无符号数有别，且最小数的绝对值要比最大数的绝对值大1.

8.补码和无符号数之间的转化。补码要扩展长度，高位需要补充原高位数字。

9.无符号数加法和补码加法，从二进制角度看，就是各个数位相加，如果溢出，直接去掉无法表示的那个最高位。一个数的相反数 = 这个数各个位求补 +1

10. 乘法运算会使用分配率的方式进行处理。除法运算队除不尽的数据会向下舍入

11.浮点数的表示：（-1）^s \*M \*2^E,主要分三种情况讨论：



1. 规格化的值：这种情况exp部分（蓝色）不全为0且不全为1，这部分的值记做e，

阶码E = e-Bias,Bias是一个偏置的常量，值为（2^k-1） - 1;f（白色部分）的取值0<=f<1，M = 1 + f。

1. 非规格化的值：这种情况exp部分(蓝色)全为0，此时E = 1 – Bias,,M = f。这样当f = 0时，就能表示出0的值。（1）中的情况是没法表示0的。非规格化数提供了一种逐渐溢出的属性。
2. 特殊值：这种情况exp部分(蓝色)全为1，当f部分为0时，表示无穷值。当f部分部位0时，表示NaN（非数字）

12.浮点的舍入：向偶数舍入

13.浮点的运算：

（1）1/+0 = +无穷 ， 1/-0 = -无穷

（2）浮点运算不满足加法交换律

（3）浮点运算的乘法不具备结合律和分配率

（4）int 转float不会溢出，会舍入；int或float转double可以精确表示；double转float可能溢出，变成无穷，也可能被舍入；float或double转出int会向0舍入，也可能溢出

存疑：  
1.为什么int转float不会溢出？要知道，int的取值-2^31到2^31 -1，而float的表达式为（-1）^s \*M \*2^E，M的取值为0到2之间，E的最大值为2^22 -1，如何表示溢出的int整数？

2.浮点数的公式推导有些稀里糊涂的

3.关于逆元，加法逆元的定义不太清楚