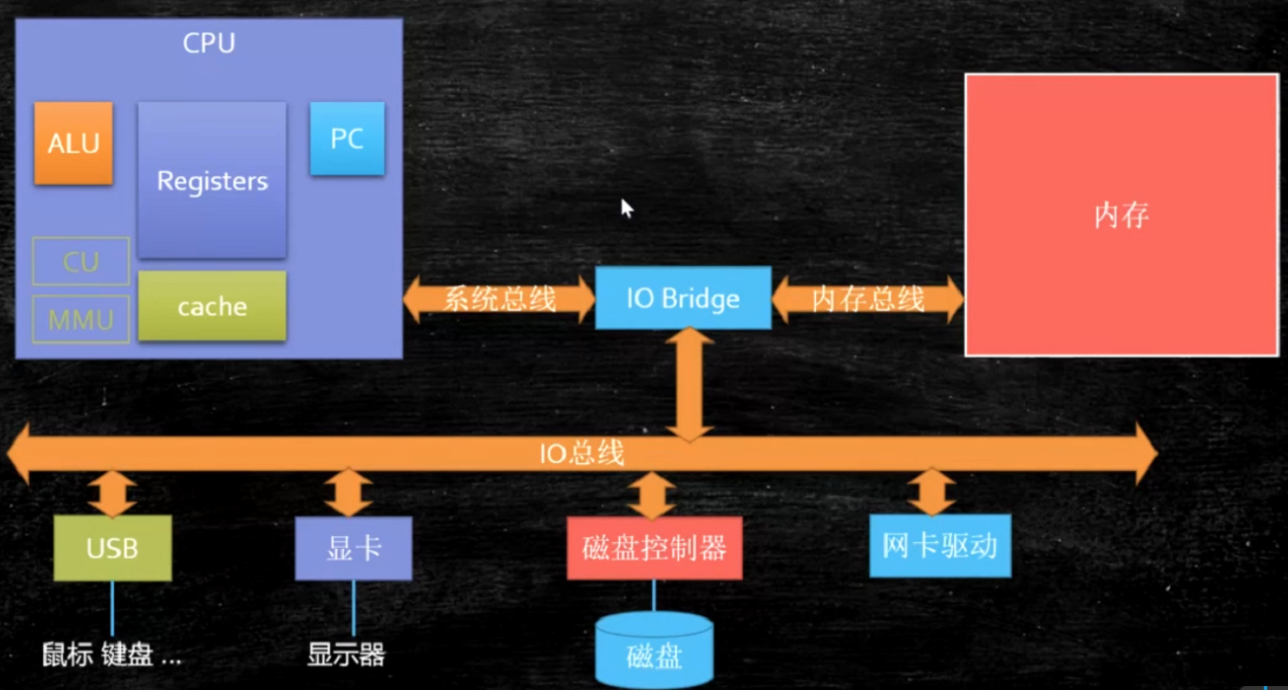
# 计算机组成

# 计算机基本结构：



1:CPU：计算机的大脑，负责计算的地方，32位64位指的就是CPU一次能处理的最大位数，同时位数也限制了内存的寻址空间。

2:内存：进程运行时数据存储的地方

3:总线：CPU和内存交互，和外设交互，发送命令的道路

系统总线：CPU发送控制指令的道路

内存总线：CPU和内存交互的道路

IO总线：CPU和IO设备交互的道路

4:外设：其他I/O设备，如磁盘，显示器，鼠标，键盘，打印机等

# CPU结构详解（可以有多个core）

## 1:寄存器：

存储操作数的地方

## 2:ALU：

使用指令 将寄存器数据进行计算的 计算单元

## 3:PC计算器：

存储ALU下一条该执行的命令

## 4:cache：

缓存内存数据的地方

## 5:AU：

控制单元，处理中断信号

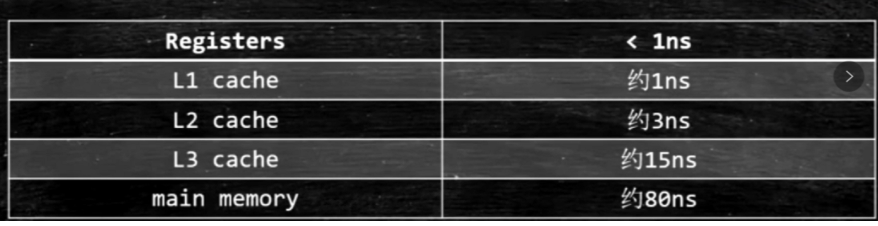
## 6:MMU

内存管理单元

## 注意：一个Core：ALU+寄存器组（超线程是多组）+ 一个PC（超线程是多个） + L1/L2 Cache

# CPU多级缓存结构





CPU的缓存一共三级，L1和L2是单个Core级别，L3是CPU级别共享。

## 缓存工作原理：

1:缓存的最小的单位是CacheLine，一般为64Byte

2:CPU读取数据顺序是，先L1，找不到去L2，再L3，最后去主存中找

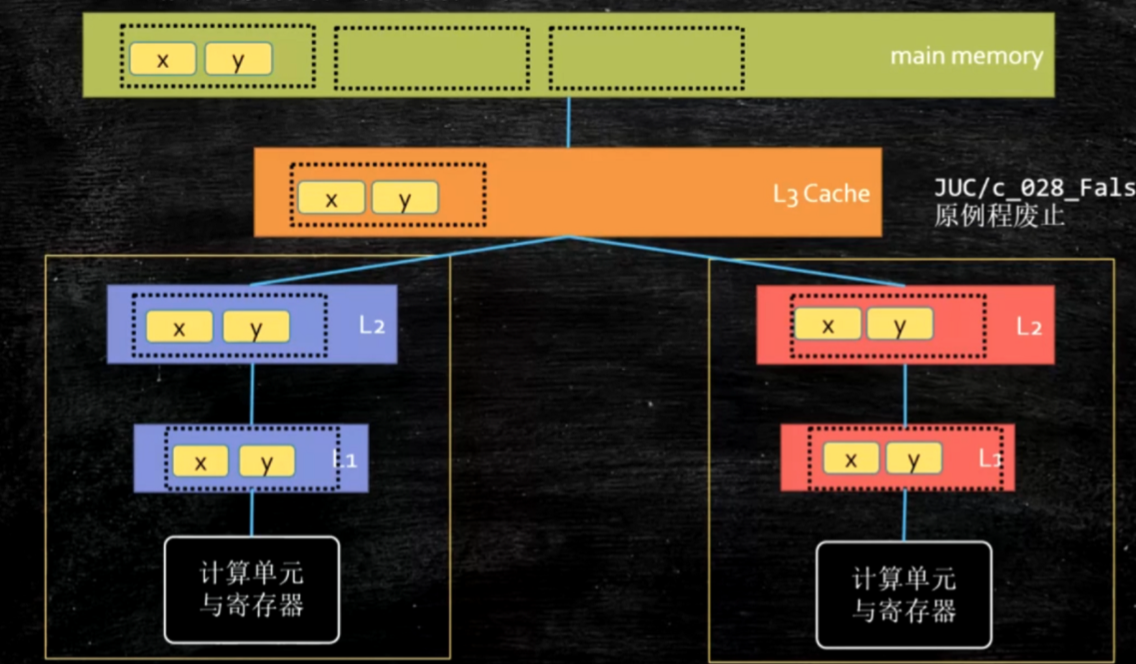
## 缓存的作用：

1：加快CPU的执行速度，更快的读取到数据

2: 原子语句的缓存锁，没有缓存ChacheLine的话，CPU通过锁内存总线来保障原子语句

## 缓存带来的伪共享问题：

如下图，如果两个线程，一个只会访问x，另一个只会访问y，由于缓存一致性原理，xy会被同时缓存到这两个core的L1,L2缓存中去，因为缓存一致性性质的存在， 只要修改了x/y，另一个线程的缓存就会失效，需要保正缓存一直之后，才能继续工作。这就是伪共享的问题。



#### 解决办法：

1：使用@Contented来修饰字段，被修饰的字段独立存在于一个cacheline中

2：在不想被共享的数据前后各放64个字节的 假数据

## 缓存带来的可见性问题：

因为不同的线程工作在不同的core上，不同的core有不同的缓存，如果一个线程修改了共享变量，其他线程的缓存上是否可以同步到？这就是可见性问题；

#### 解决办法：

CPU层面通过MESI缓存一致性原理保证可见性，每个CacheLine都有四种状态：

1:modify：缓存中已经修改，但是没有刷到主存

2:exclusive：独占cacheLine，而且数据和主存一致

3:share：共享cacheLine，而且和主存一致

4:invalid：失效，其他线程改了cacheline，但是我并没有看见

原理：在invalid的时候，必须同步新数据

# CPU乱序执行问题

## 背景：

CPU为了加快执行速度，有时会将指令执行的顺序打乱，来提升速度。单核CPU保证了指令排序之后的逻辑遵循as-if-serial，也就是乱序不会影响最终的结果。

## 问题：

as-if-serial不适用于多线程并发场景，比如java的DCL单例问题；

## CPU层面解决方式：

1:通过使用内存屏障来禁止指令重排序

比如，lfence sfence mfence

2:使用lock指令来锁内存总线/cacheline（相当于一个mfence）

## Java层面：

也是使用内存屏障（底层是使用Lock来实现的，**Lock会强制刷主存触发MESI，间接保证了可见性**）



volatile实现逻辑：



## Happends-Before：

JVM虚拟机层面自己也保证了一些JVM层面的禁止指令重排序

# 中断

## 背景：

中断是告诉CPU，有一些事情需要CPU做，CPU要对对应的中断作出响应，中断具体是由CPU的CU来控制的；

比如时钟振荡器 不停发哒哒哒信号（频率就是CPU的主频率），CPU接受到了，就是一个时钟中断，好几个时钟中断组成了一个时间片，一个进程的执行是按时间片来计算的。

## 原理：

CPU维护一个中断向量表（map：key中断号，value：回调函数），收到中断之后，会根据中断号找到回调函数，执行之。

## 中断分类：

#### 硬中断：

硬件发出的中断，比如时钟振荡器发出的时钟中断；或者网卡接受到数据包，发出的网卡中断，CPU收到之后，走回调函数，也就是网络协议栈解析数据。

#### 软中断：

软件发出的中断，进程需要调用系统调用，发出的中断，接受到中断之后，CPU需要将进程的执行等级从3升级到0（内核态），然后执行对应的系统调用方法，执行完成之后，CPU等级降到3。

注意：系统调用不涉及进程的切换，始终都是一个进程，但是涉及栈的上下文切换。

#### 中断命令：

int [0-255]：一个CPU可以支持255个中断

软中断为：int 0x80

