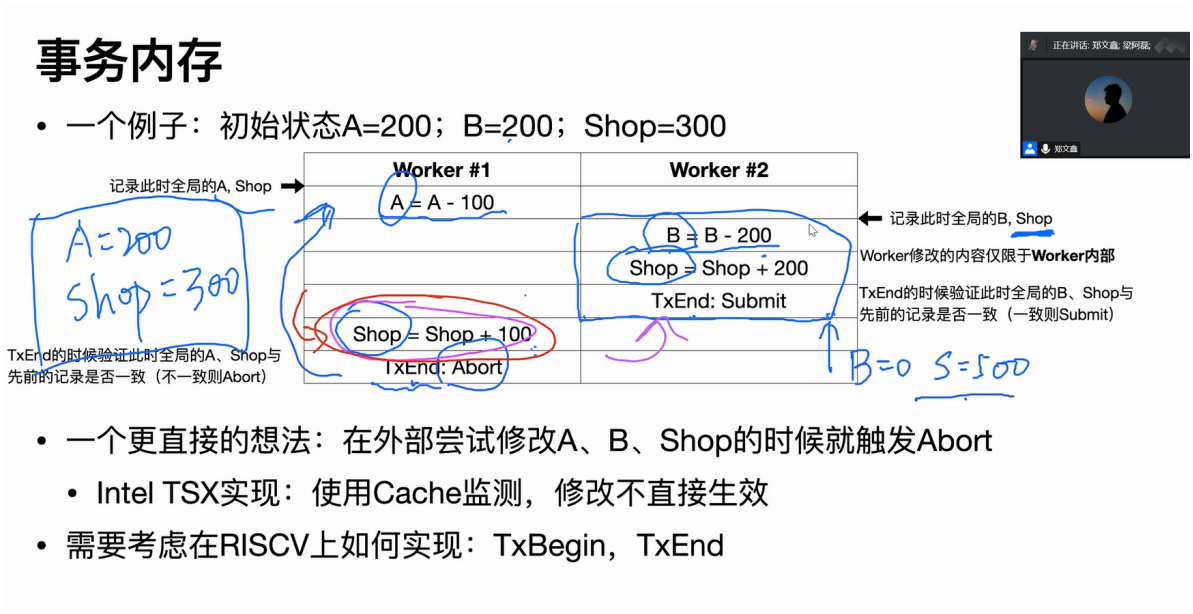



cache line 要加 tag，一旦被换，直接abort

最好把相关的东西放在一个 cache line

加个tag，别的核修改时，改tag，然后检查tag 就可以判断abort

snoopy协议，检查被abort的line有没有tx标记，有就abort

abort之后跳到abort_handler然后跑，该程序是手动指定的

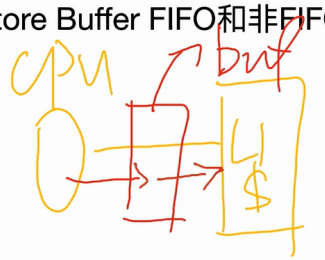
| 姓名 | 问题 | 回答 |
| --- | --- | --- |
| 周秉霖 | （1）在riscv上实现TxBegin，TxEnd这句话是什么意思，是指我用riscv已有的指令来实现，还是我可以进行riscv指令集的扩展设计，可以自己进行硬件设计使我可以自己创造新的指令？（2）另一个问题是，我是只需要写出riscv指令的组合就足够了，还是说对于我自己设计的指令，我需要明确指出它每一个阶段在做什么？ | 1. 期望你自己创造新的指令，如果要用已有的也可以； 2. 不论哪种方案，都需要写出每个阶段 |

addm，xbegin，xend

细想一下，是不是不用别的核改内存，只要在一个事务里连续访问数组，当访问到一定大小，abort的时候就是cache的大小，虽然这个时候要求别的core不能有干扰（用到时load进cache，之后监测，同样地，如果发生cache冲突，就会产生abort）

# 弱内存模型

- 引入Store Buffer带来的问题
  - 处理器的内存写操作会直接写入Buffer中，**此时CPU会认为该写入已经生效**。
  - 讨论：
    - 加入Store Buffer之后加速的原因是什么？
    - 会带来什么问题？核心原因是什么？
    - 如果Store Buffer FIFO和非FIFO的区别是什么？




1. 如果CPU与L1直接通信，则需要等L1实现核间同步，现在直接放到buffer里，然后CPU继续跑，就不会被卡住
2. 原因是这个CPU以为已经写入L1实现同步，但实际上可能还没写，别的核用到的可能还是旧值
3. FIFO可以保证 WAW，非FIFO可能顺序乱掉

solution：对于自己的buffer和L1 cache，取值要先找自己的buffer

store buffer不对外

## 弱内存模型（Weak Memory Models）

在弱内存模型中，四种类型的内存重排序都有可能经历。只要不改变单线程的行为，弱内存模型可随意对代码进行重排序。

像C++11这种编程语言，暴露的是弱内存模型，但当在这些语言中使用底层原子操作时，不需要关心更底层的是否**强硬件内存模型**。

例如起初有x=y=0，

CPU1 进行 x=1,y=2的操作

CPU2 进行$m = x, n = y$

则符合若内存模型规定的是：

m=0, n=0

m=1,n=0

m=1,n=2

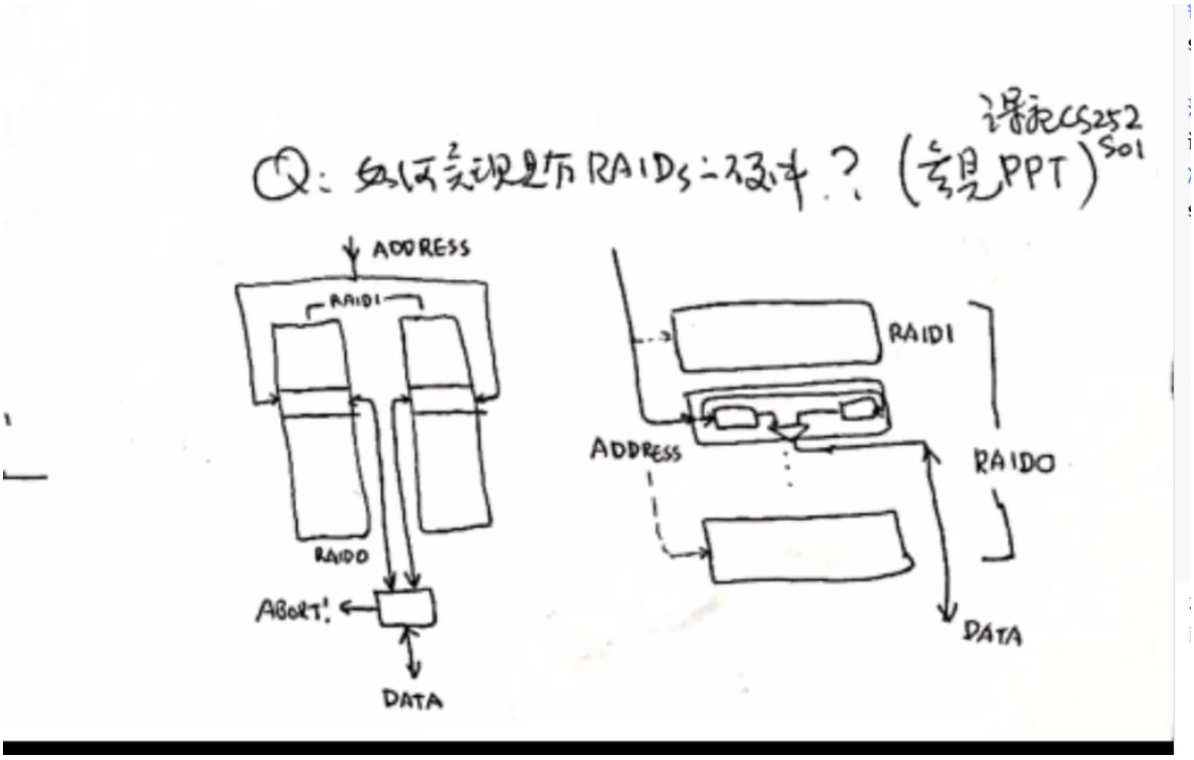
# 冗余存储

Raid 0：一块硬盘或者以上就可做
raid0优势：数据读取写入最快，最大优势提高硬盘容量，比如3快80G的硬盘做raid0 可用总容量为240G。速度是一样。
缺点：无冗余能力，一块硬盘损坏，数据全无。
建议：做raid0 可以提供更好的容量以及性能，推荐对数据安全性要求不高的使用。

Raid 1：至少2快硬盘可做
raid1优势：镜像，数据安全强，2快硬盘做raid一块正常运行，另外一块镜像备份数据，保障数据的安全。一块坏了，另外一块硬盘也有完整的数据，保障运行。
缺点：性能提示不明显，做raid1之后硬盘使用率为50%.
建议：对数据安全性比较看着，性能没有太高要求的人使用。

## RAID10

## RAID01




实现上的区别

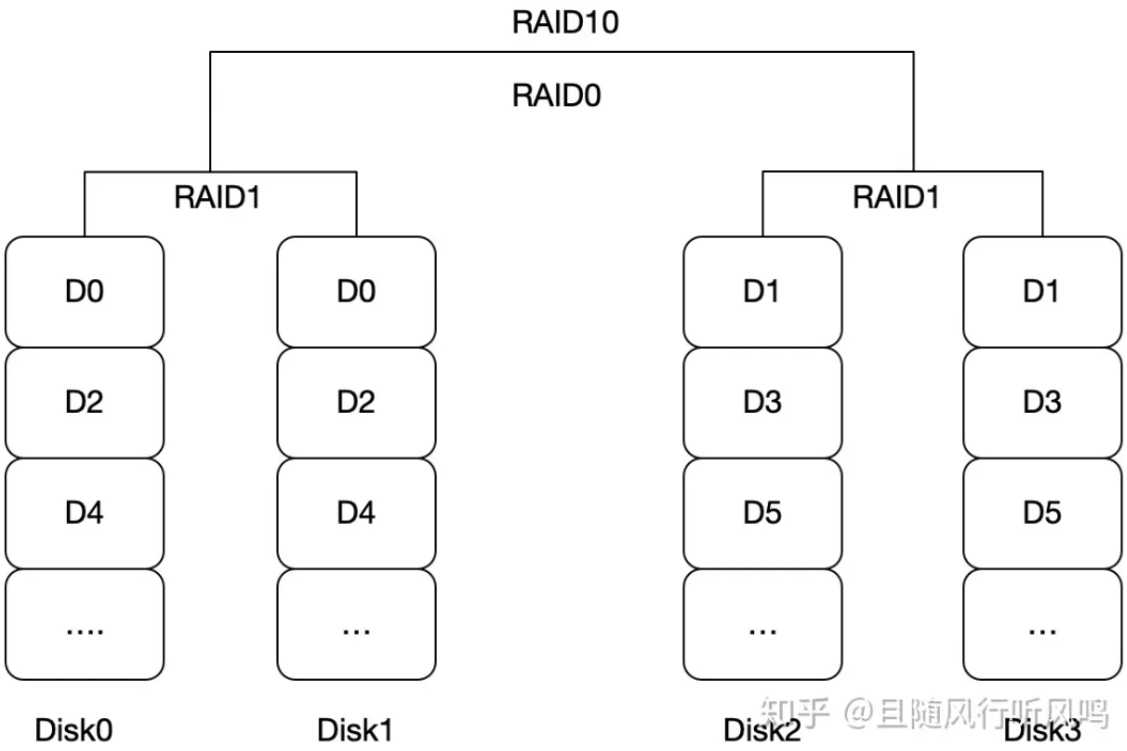
磁盘使用率

容错

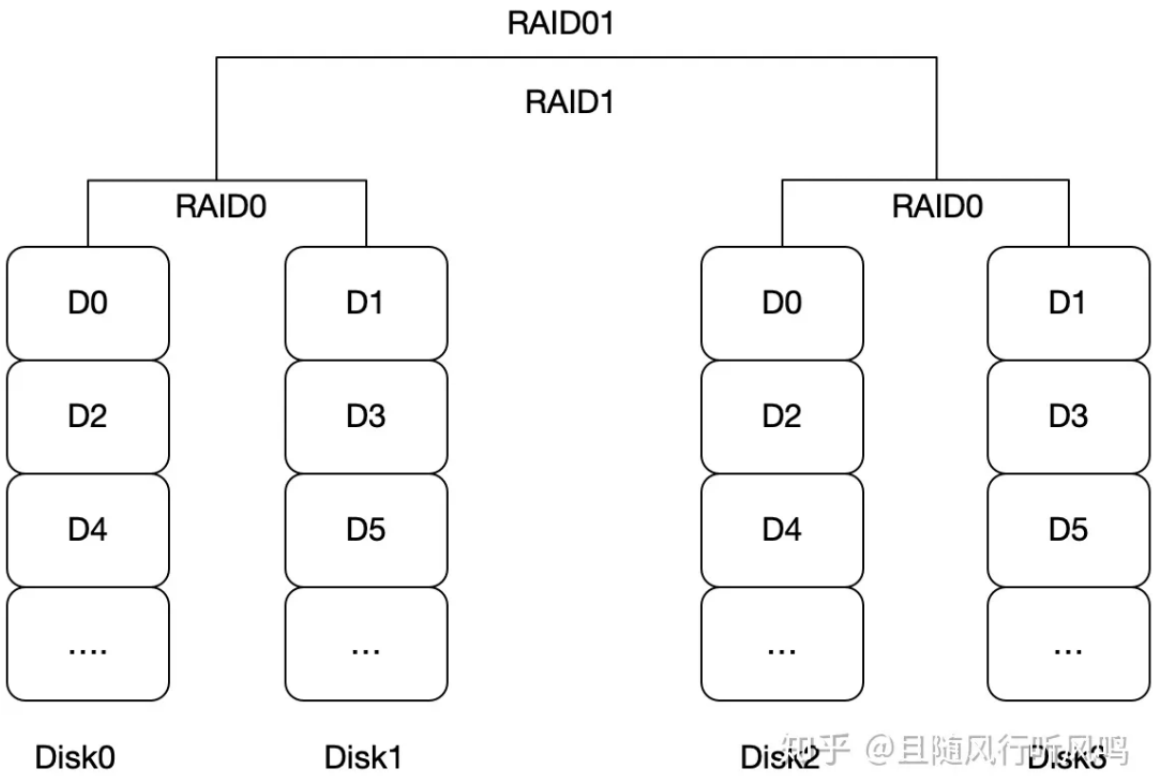
假设有四块磁盘做磁盘阵列：

此时假设Disk0损坏时

那么对于RAID10，当Disk0损坏时，同组的Disk1时可以正常工作的（因为Disk0与Disk1共同组成RAID1，RAID1的特性是同一组的磁盘互为镜像，其中一个损坏时，同组的另一个会代替损坏的继续提供服务），所以整个磁盘阵列可以正常工作，此时只有当Disk1也损坏时，整个磁盘阵列才会无法工作。简单的计算损坏率的话，RAID10的损坏率为 1/3



对于RAID01，当Disk0损坏时，Disk1也无法再提供工作（因为Disk0与 DIsk1共同组成RAID0，RAID0提供了并发写，但是没有做任何冗余功能，没有任何容错能力，当Disk0损坏时，Disk1也无法再进行工作），此时，当Disk2或Disk3损坏时，整个磁盘阵列就无法再工作.简单计算损坏率：RAID01为 2/3





初始化raid？处理读指令？

因此，当你把两块硬盘组建成一个RAID1——如果是你同时新买的两块硬盘当然最好，然而也不能排除其中一块是已经在使用的，上面满是各种数据的硬盘，另外一块是你为了组RAID1新买的硬盘。所以RAID卡/驱动需要把某一块硬盘（主硬盘）的数据原样写入到另外一块硬盘（副硬盘）上，这个过程就是初始化。前面说了，RAID卡/驱动并不知道哪些数据代表何种意义，因此这个初始化的过程，只能把整个主硬盘的全部扇区的数据读取出来，写入到副硬盘的对应扇区。

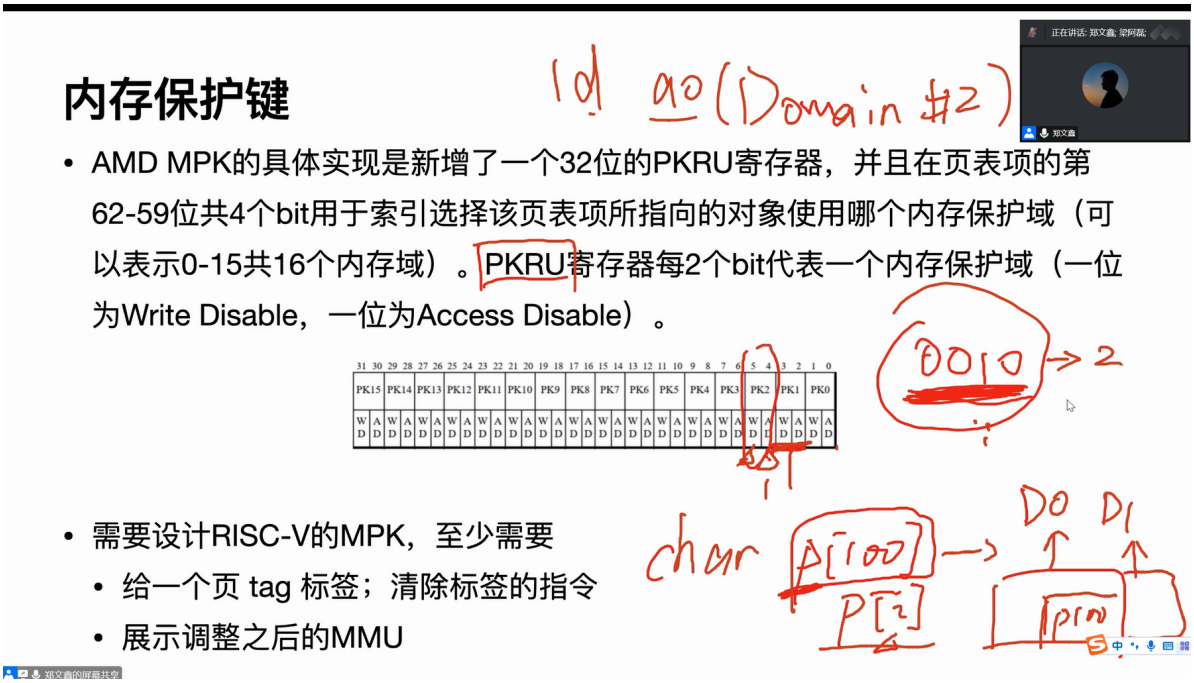
## Solution：

所以实际上的意思是我们要用RISCV的指令来实现 disc control 那个CPU的操作。
然后要做的实际上就是对于一个收到的地址，如果是RAID10，就先用地址取模找到对应磁盘对，
然后对两路同时load，再进行次按位check?

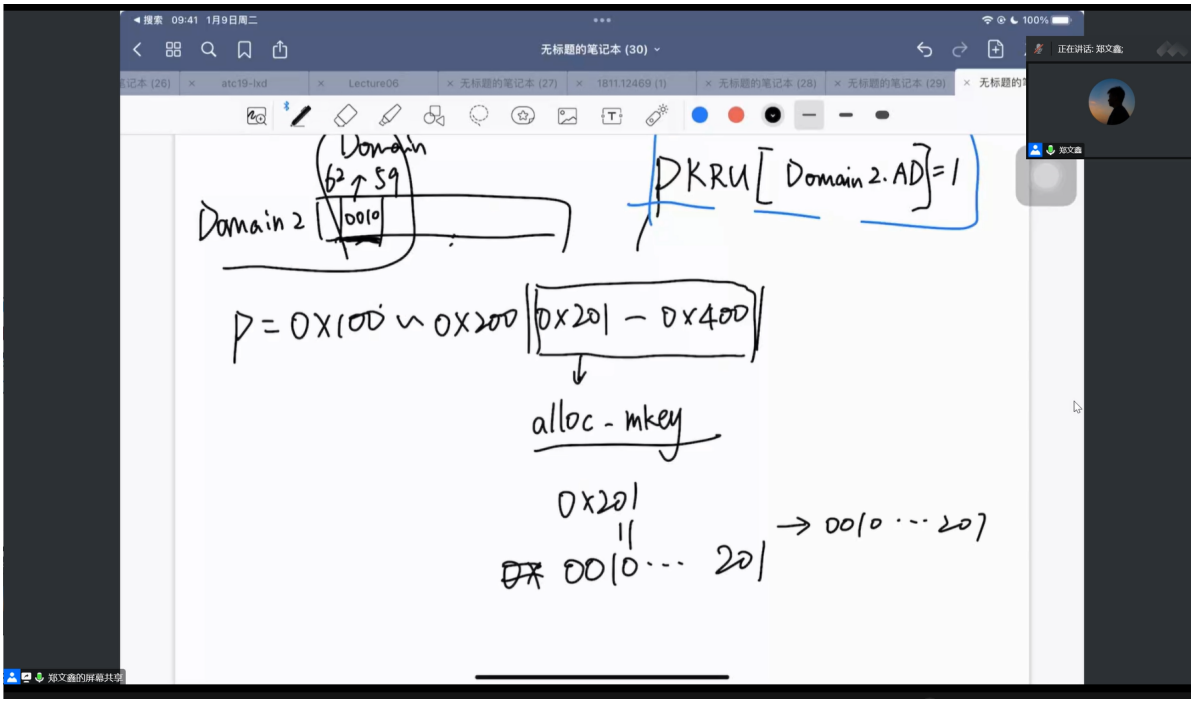
对于RAID01就对两路都发条load指令?
但是从哪个盘取到了RAID0这层是不是还得要个CPU来算?

# 内存保护键

# 内存保护键




标注domine是以页表为单位的，实现对P【100】的越界判定通过把P【100】放在页表最后，保证后项domine不一样




tlb内的地址也会带着 标志domine 的 0010

- asid作用在tlb上，目的是为了保留一部分tlb entry在context switch的时候不被flush掉，减少tlb miss率从而提速
  mpk作用在page table上，目的是为了 1. 通过pkru寄存器来更加方便地管理page table entry的权限（只需要修改寄存器的值，不需要访问多级页表，提速） 2. 更细粒度的page table entry权限管理。我们可以给不同的thread分配不同的pkru 寄存器的值，这样对于同一个page table, 不同thread的rwx权限是不同的，实现隔离