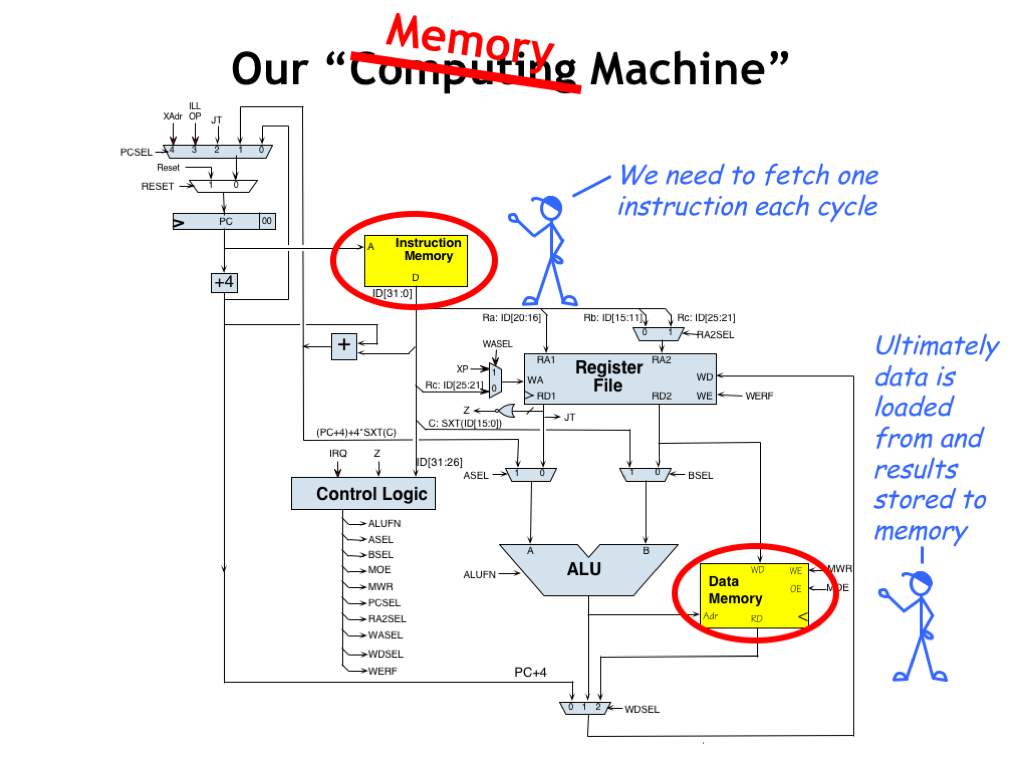
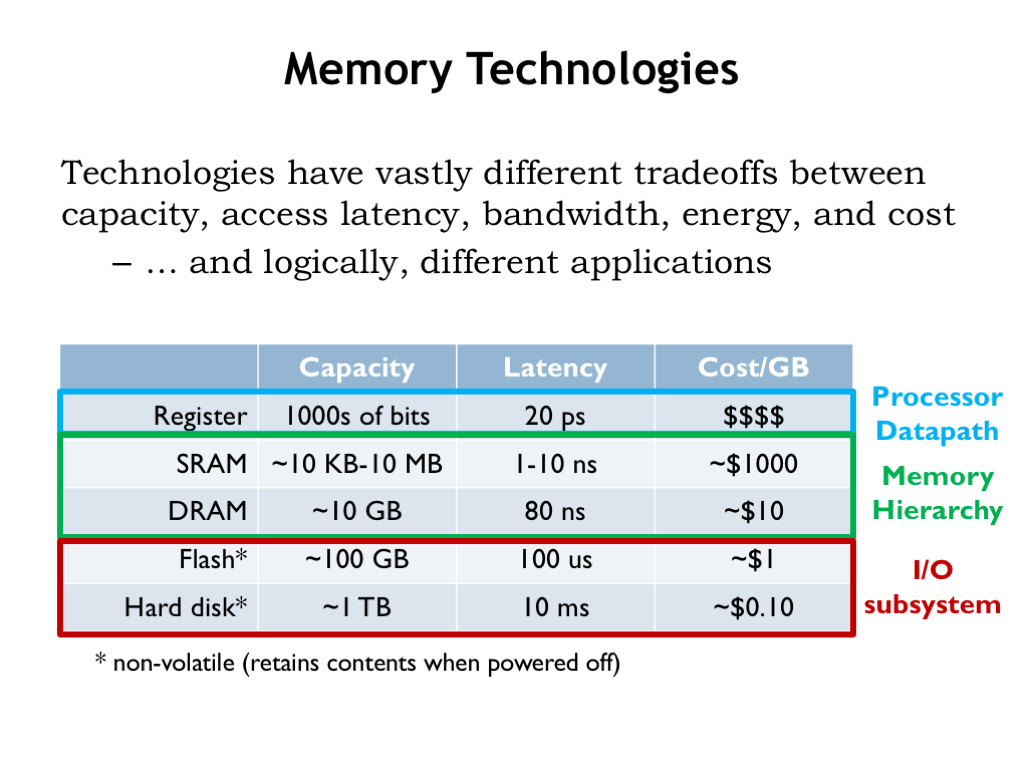
# 课件

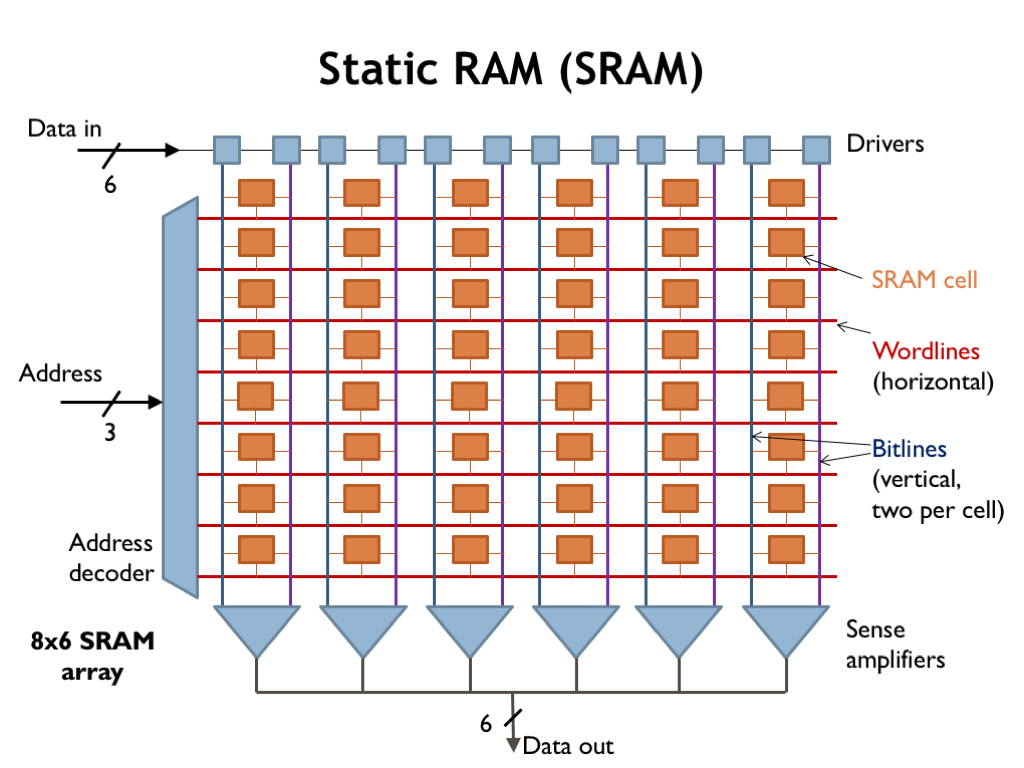
## 我们的内存机器



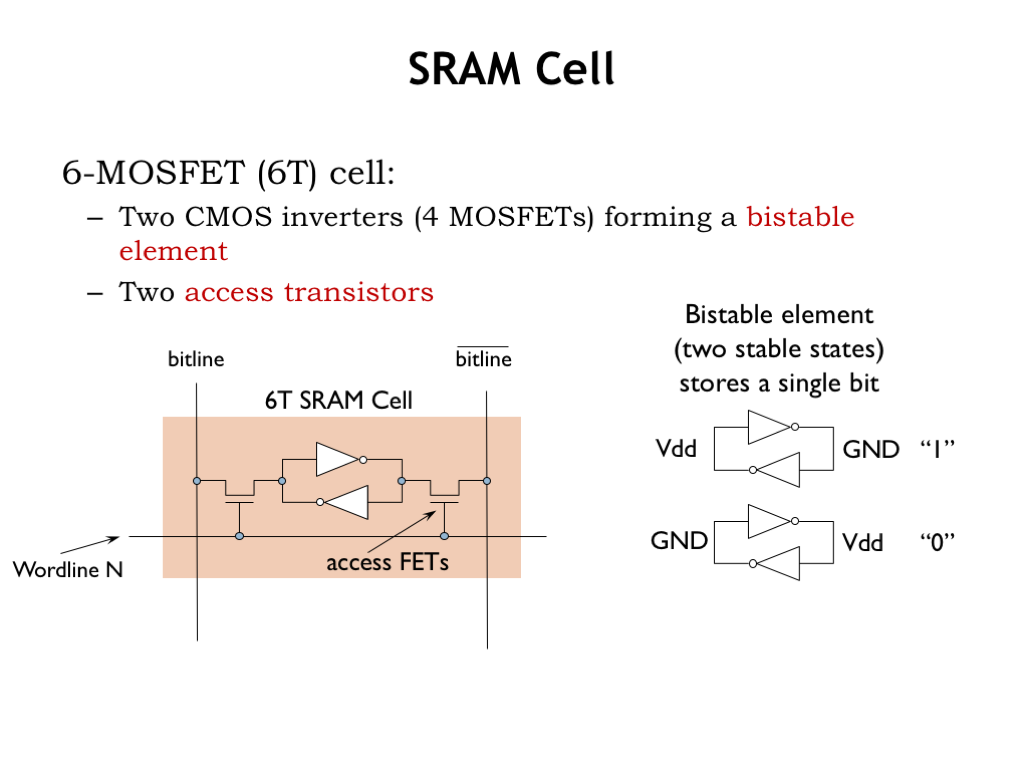
## 内存技术



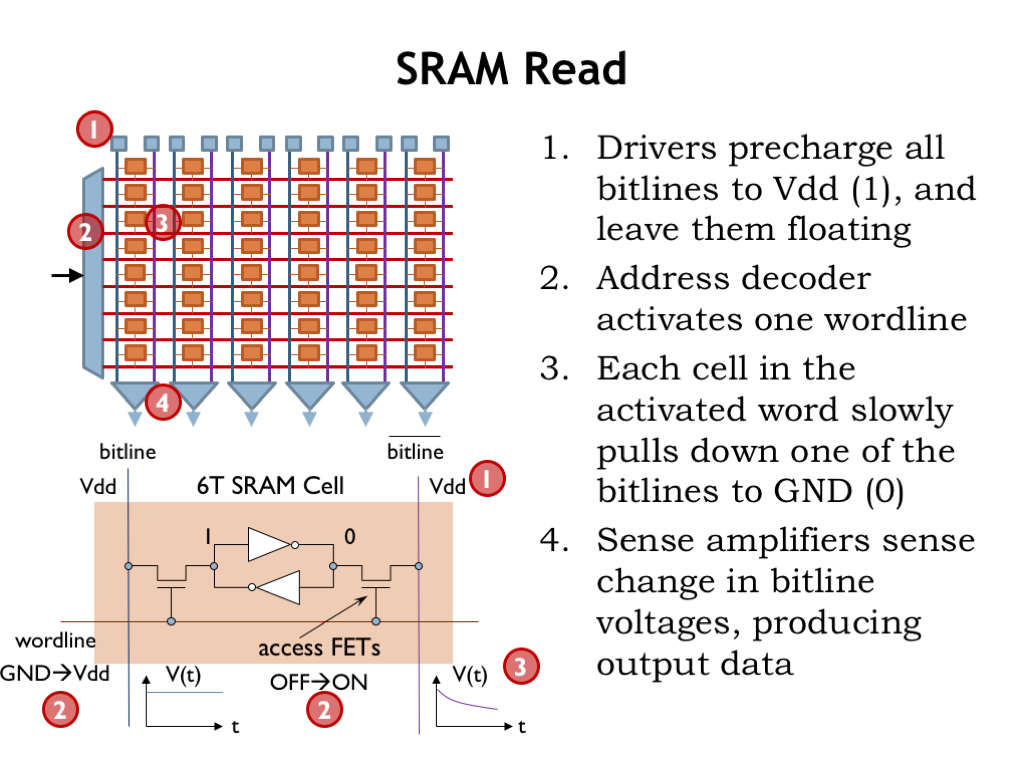
## SRAM



## SRAM单元



## SRAM读



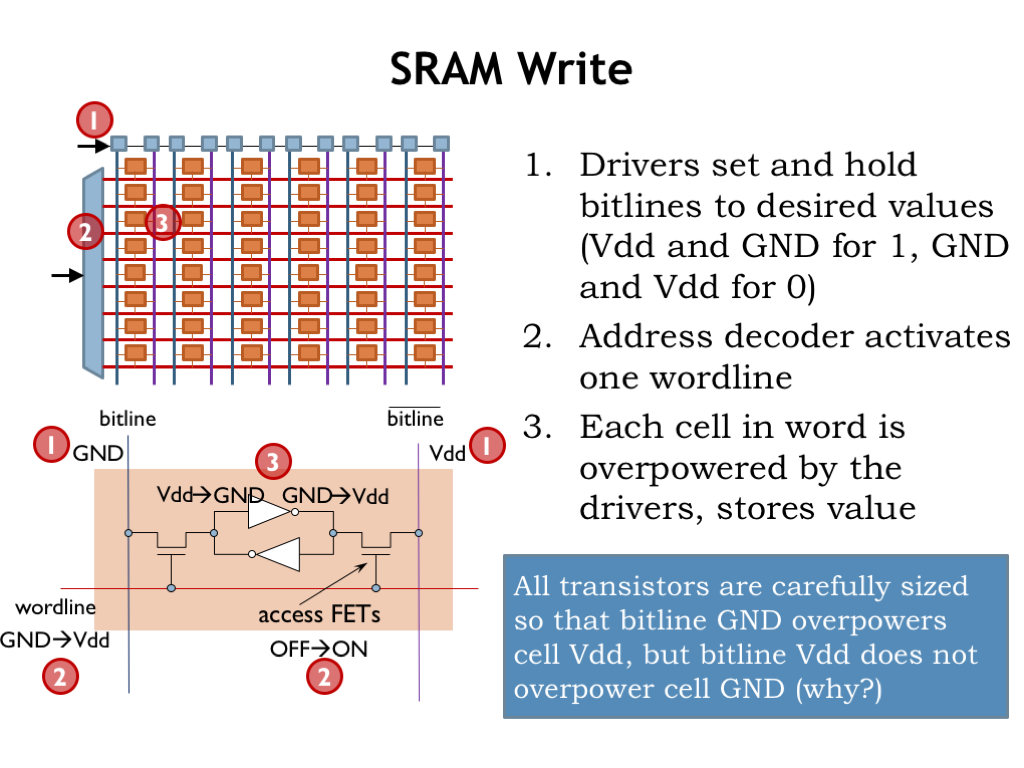
1、所有bitline电压都设置为1

2、某个wordline电压设为高电平，让对应cell的两个mos管接通

3、cell两边bitline中的某一个肯定会主键降为低电平

4、无需等到某个bitline降低到gnd，感应放大器感应到变化即可有相应输出

## SRAM写

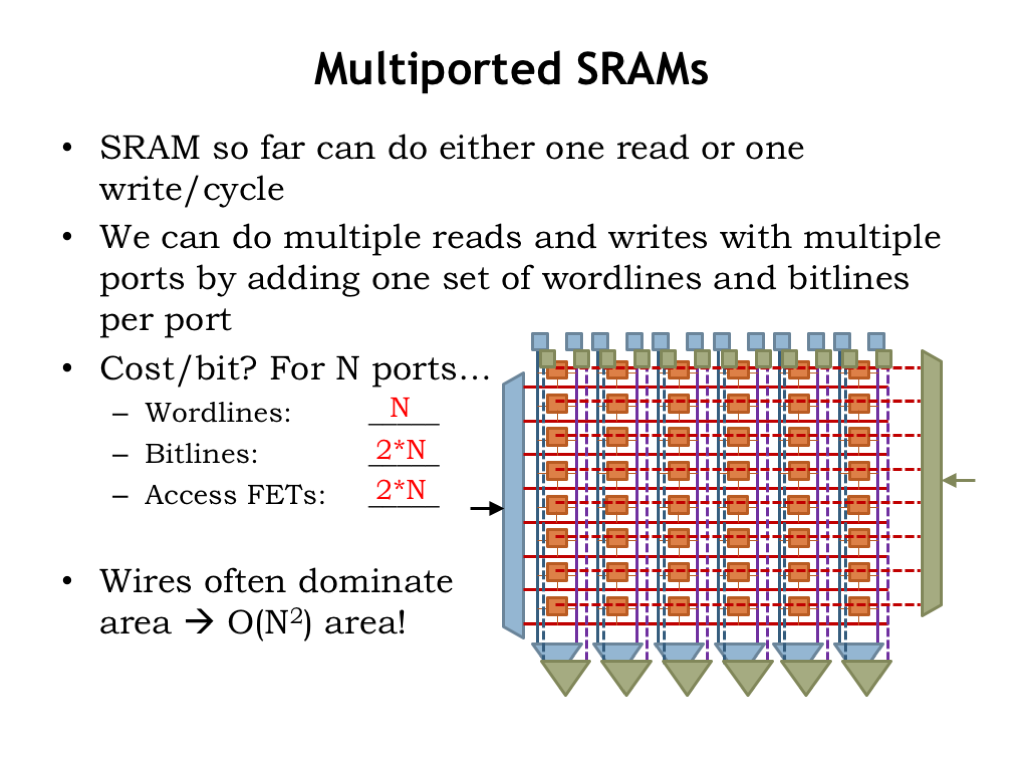


1、驱动将bitline设置为要写入的电压

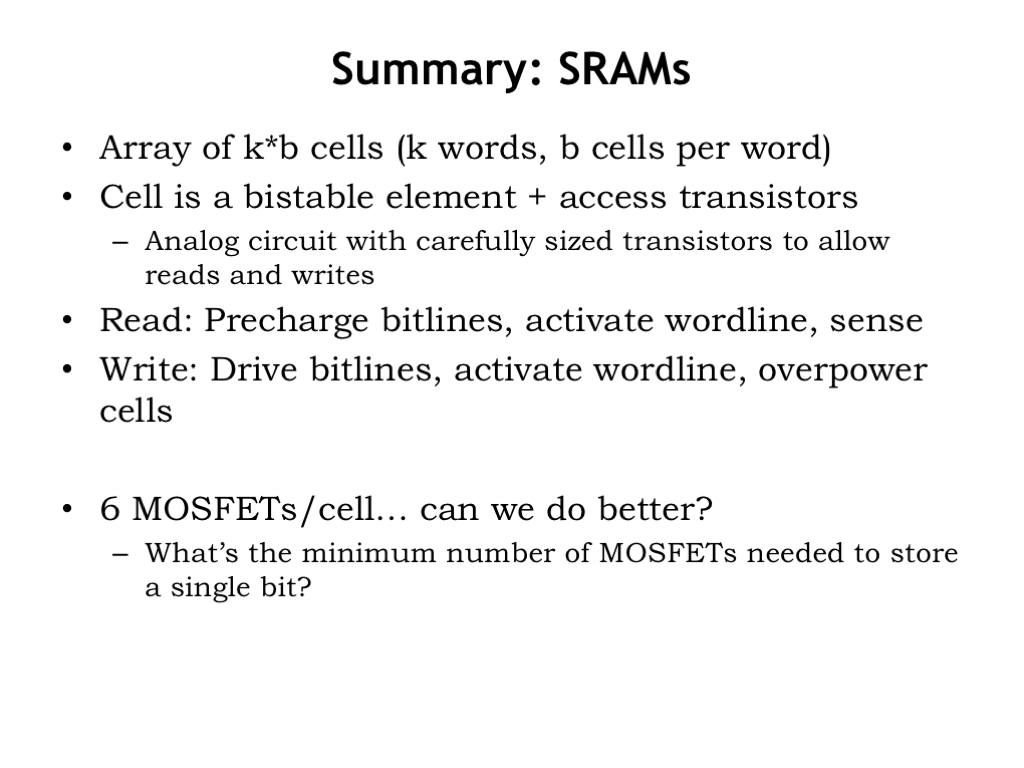
2、地址解码器选中某一wordline

3、bitline中的值覆盖掉cell中的值

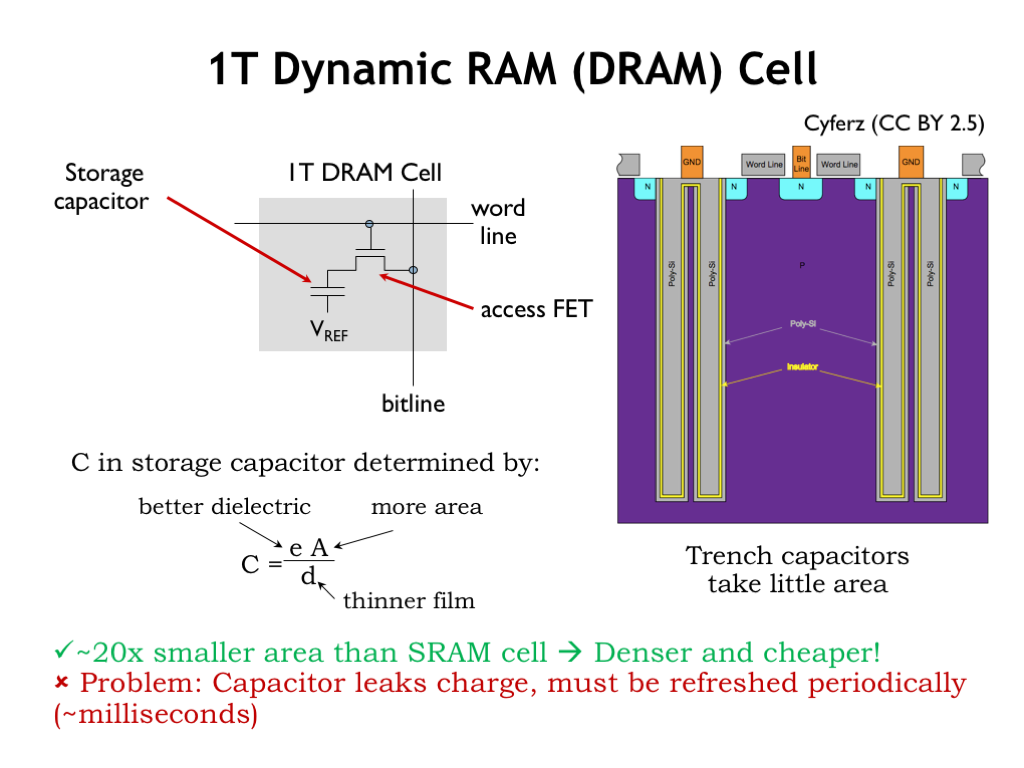
## 多端口SRAM



## SRAM总结

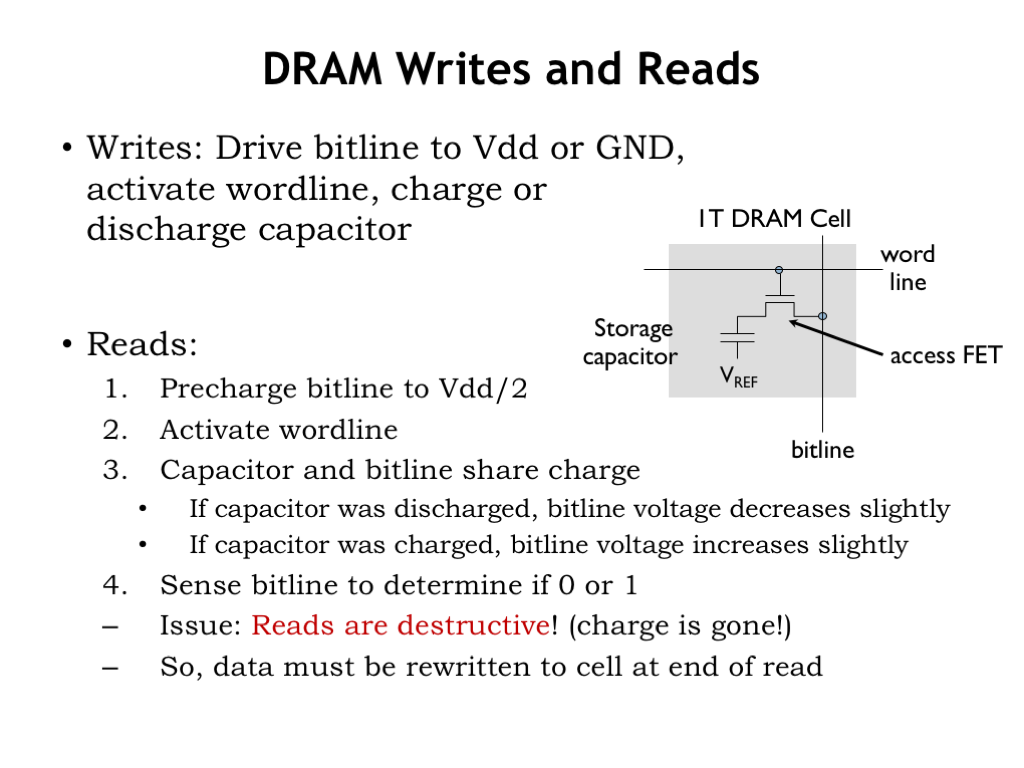


## DRAM Cell



因为电容器、mos管漏电，不能稳定保持一定的电荷量，所以需要周期性地读取、重写cell的值，保证cell中的值有效

## DRAM写和读

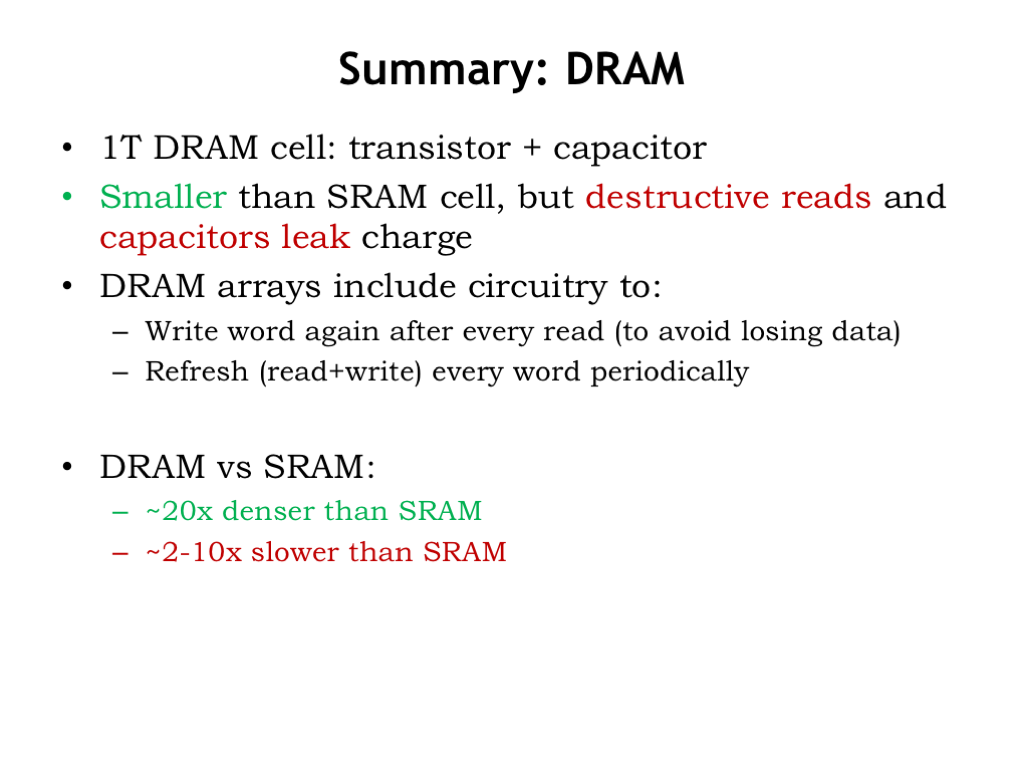


电容存放电荷用来代表0、1，wordline选中某行，读时，bitline电压设为Vdd/2，根据bitline的充放电判断cell的值，若cell为高电平，那么每次读会导致cell电荷量减少，需要重写；

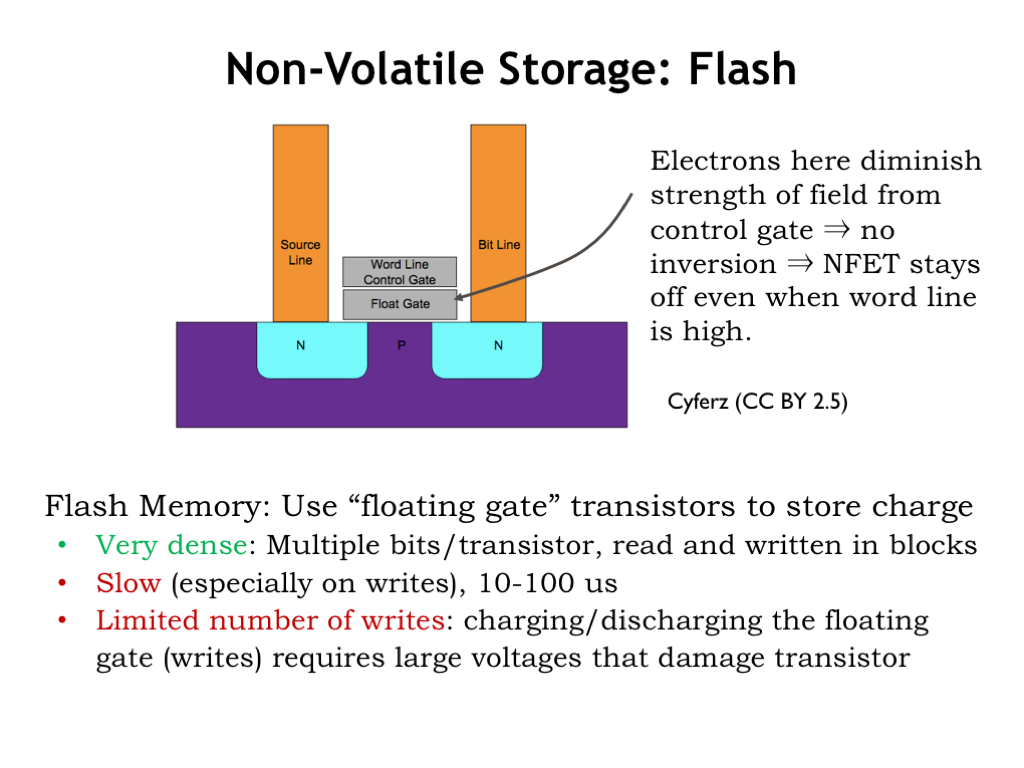
写时，bitline设高低电平，来对电容充放电。

行地址选中（延时较大）后，同一行内，选择不同列时延迟较低

## DRAM总结



## 稳定存储：Flash



floating gate无电荷时：wordline施加V1，即可让NMOS导通。

floating gate有电荷时：wordline施加V2（一个大于V1的电压），才可以让NMOS导通。

那么施加一个V1到V2之间的电压，通过观测NMOS是否导通，得知floating gate处于哪种状态。

写有次数限制。

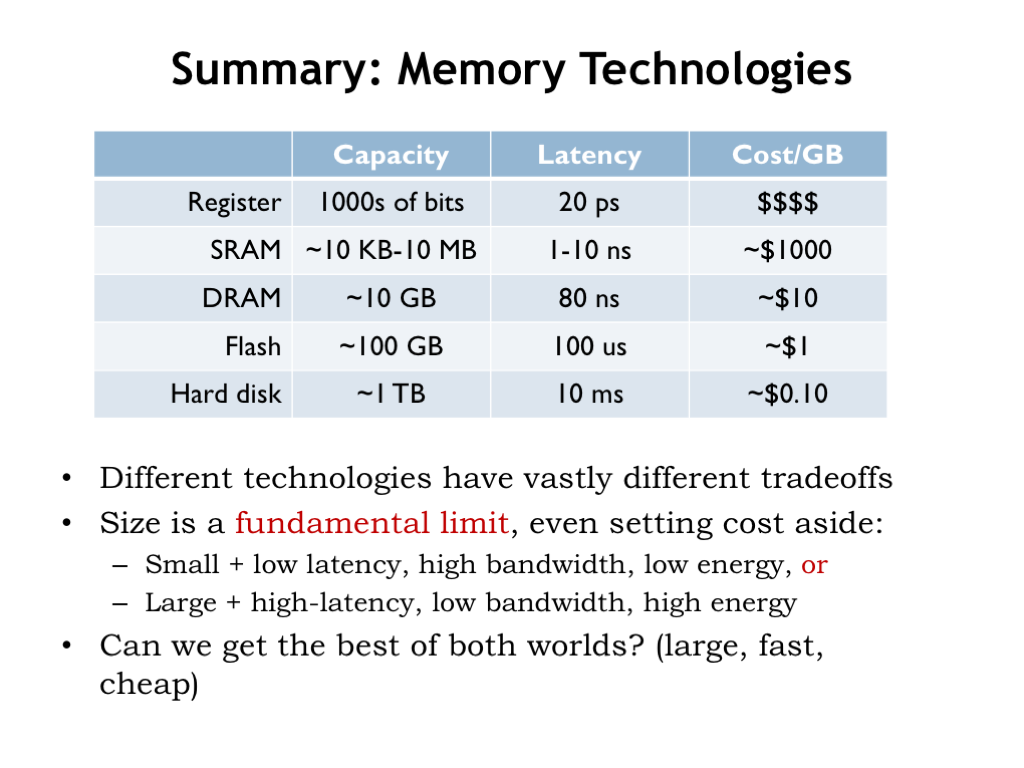
## 稳定存储：硬盘



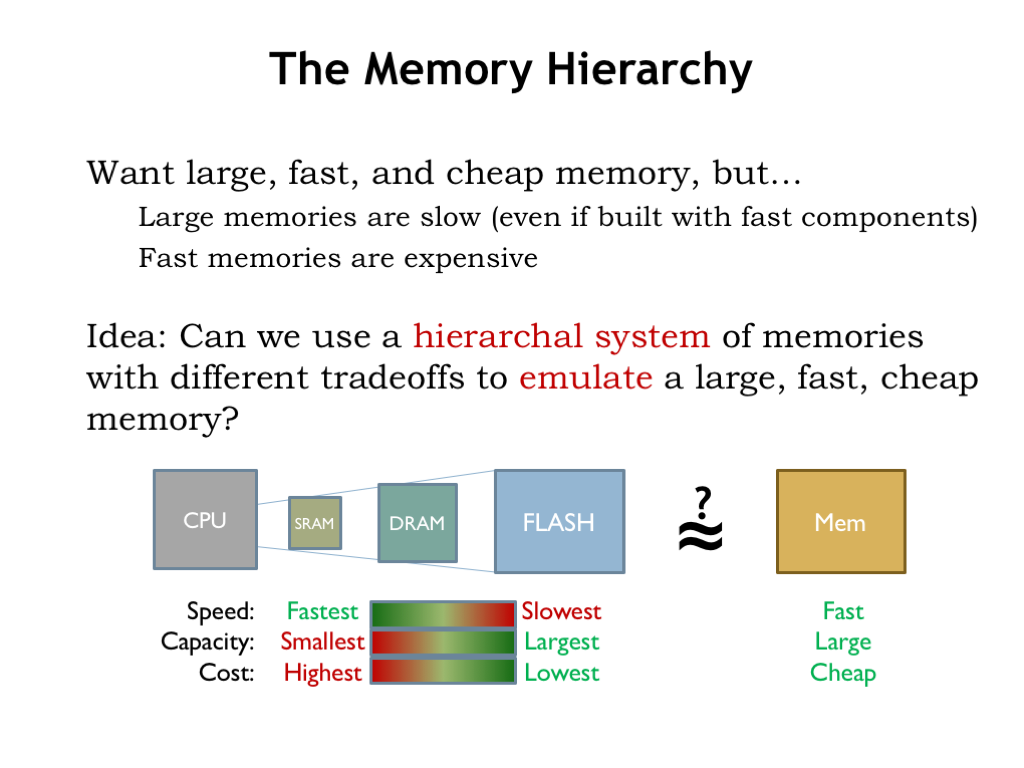
确定哪个扇面，再确定哪个圆圈，再确定圆圈哪一段。

串行读/写100MB/s，随机读写100KB/s

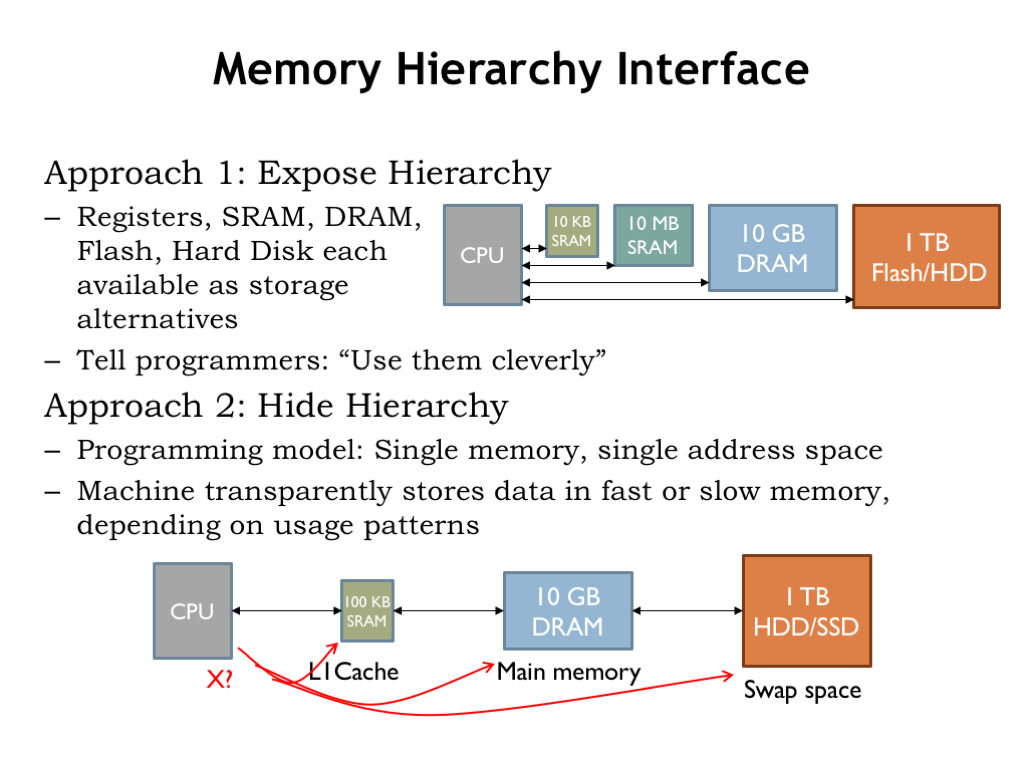
## 总结：内存技术



## 内存分级



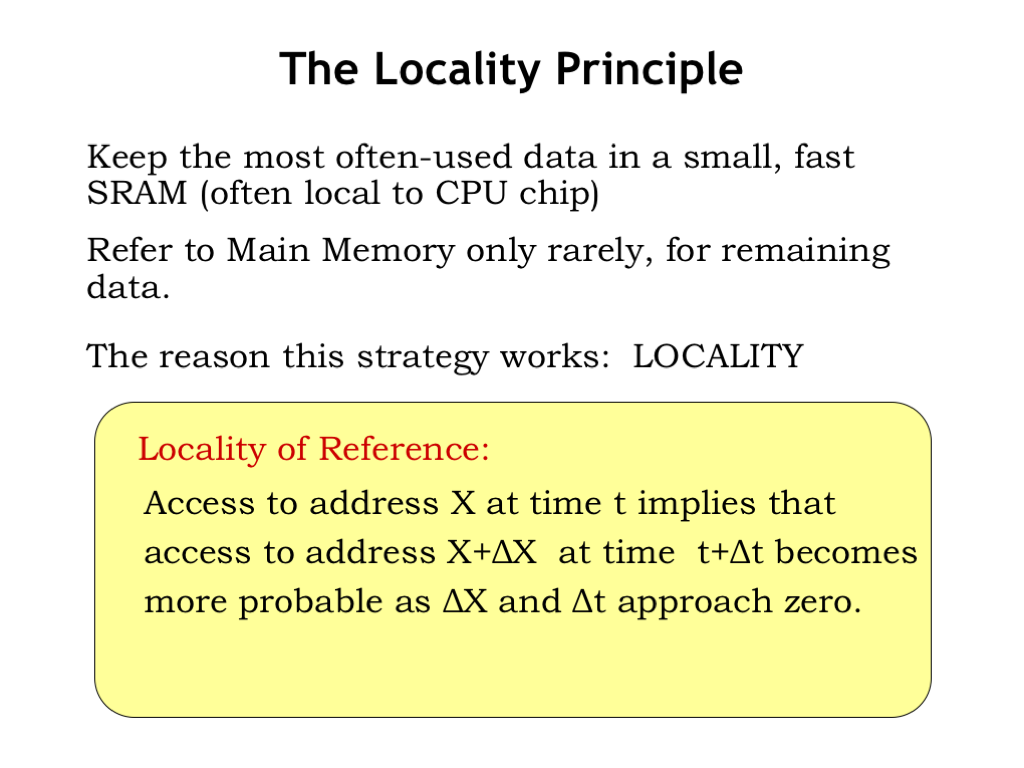
## 内存层级连接



程序直接管理各个层级内存：超级计算机Seymour Cray就是个典型例子。

计算机内存系统管理各级内存：程序员只知道有一个大内存，有统一的地址可以访问。

## 局部原则



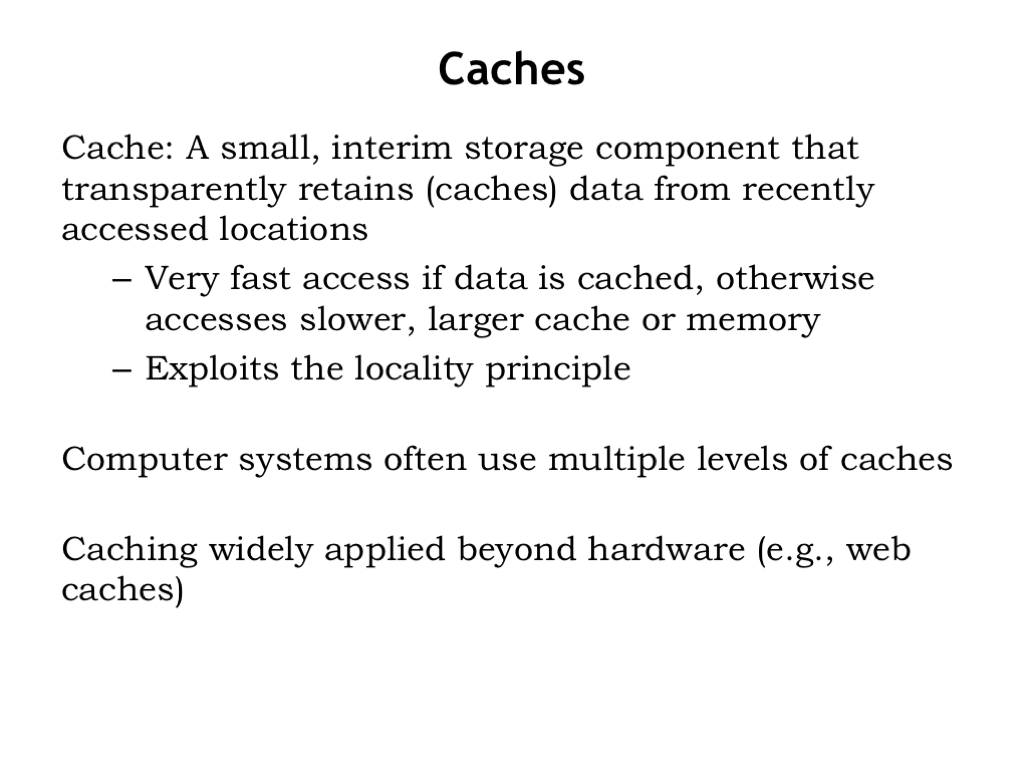
局部性访问，空间局部性：访问地址a后，地址a附近的其他内存也更可能被访问；时间局部性：访问地址a后，接下来的时间里，a也很可能再次被访问。

访问DRAM中的数据后，内存系统会把DRAM中该数据附近的一块数据，放到SRAM中缓存。

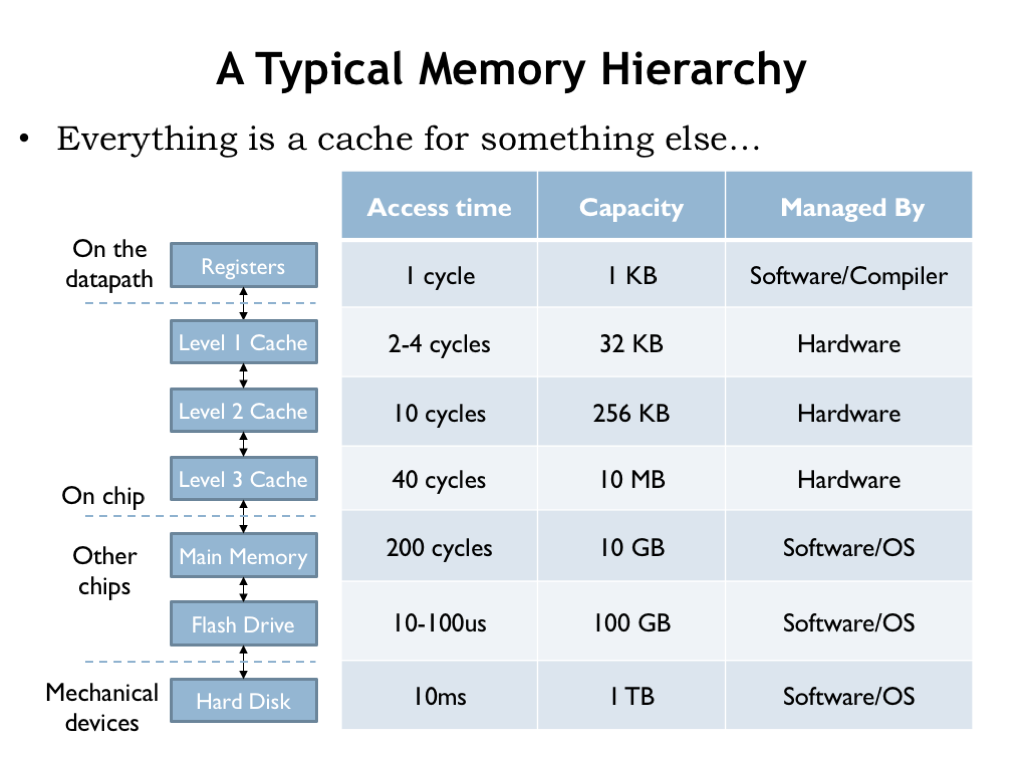
## 内存引用模式



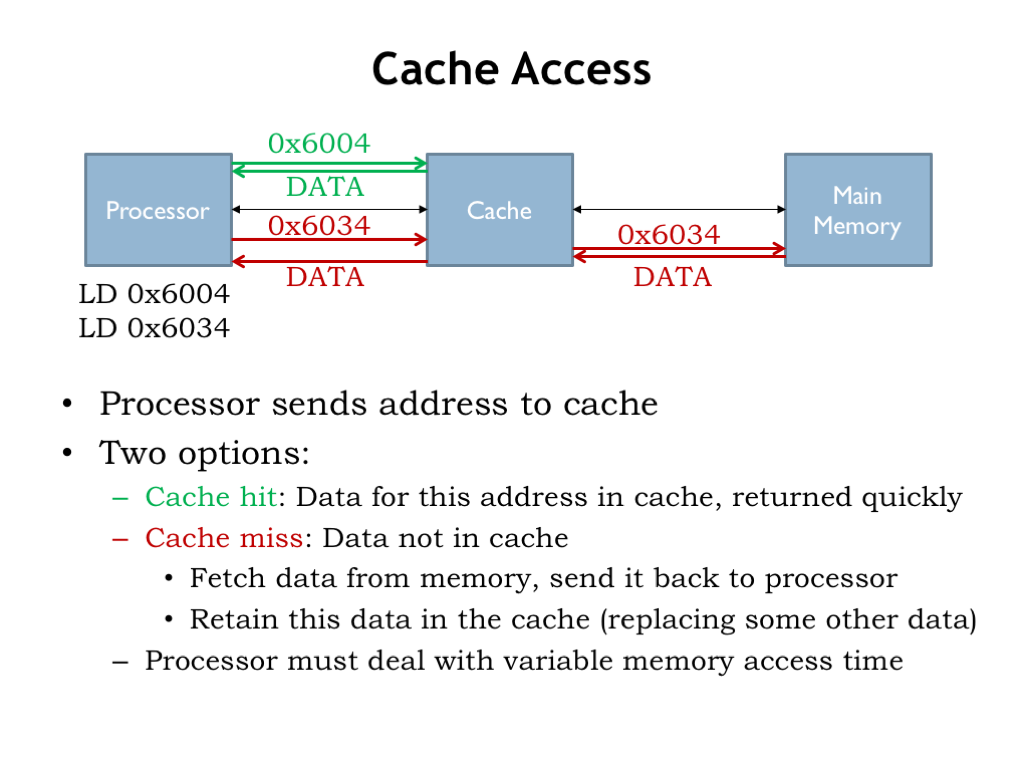
## 缓存



## 一个典型的内存分级

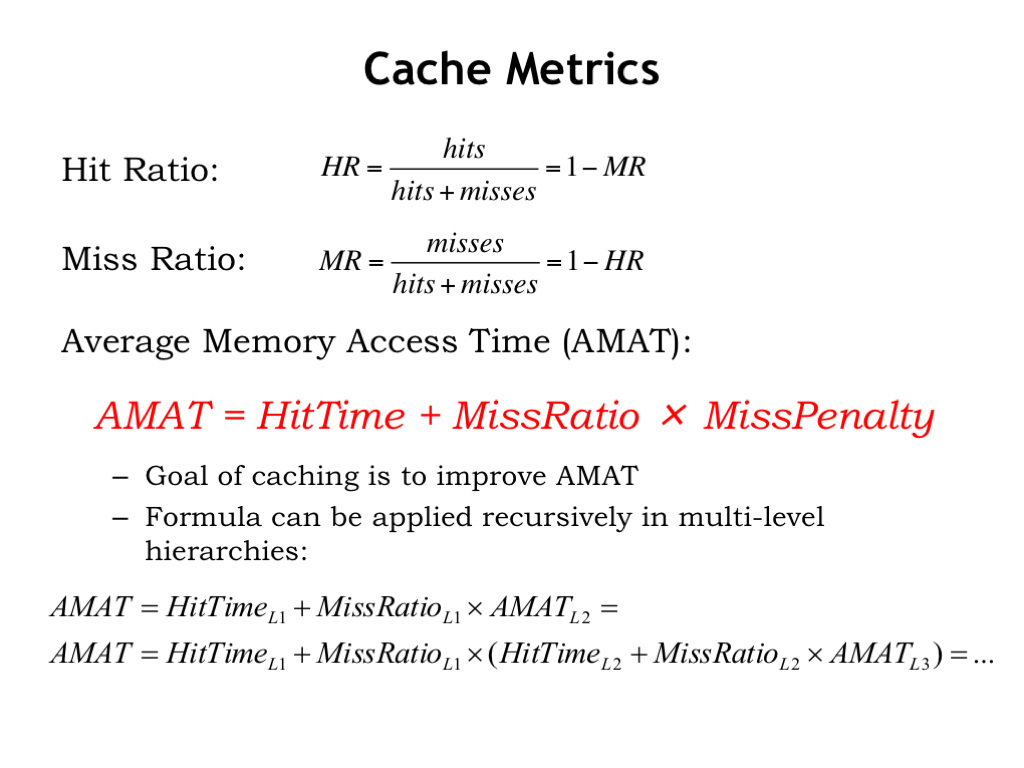


## 缓存访问

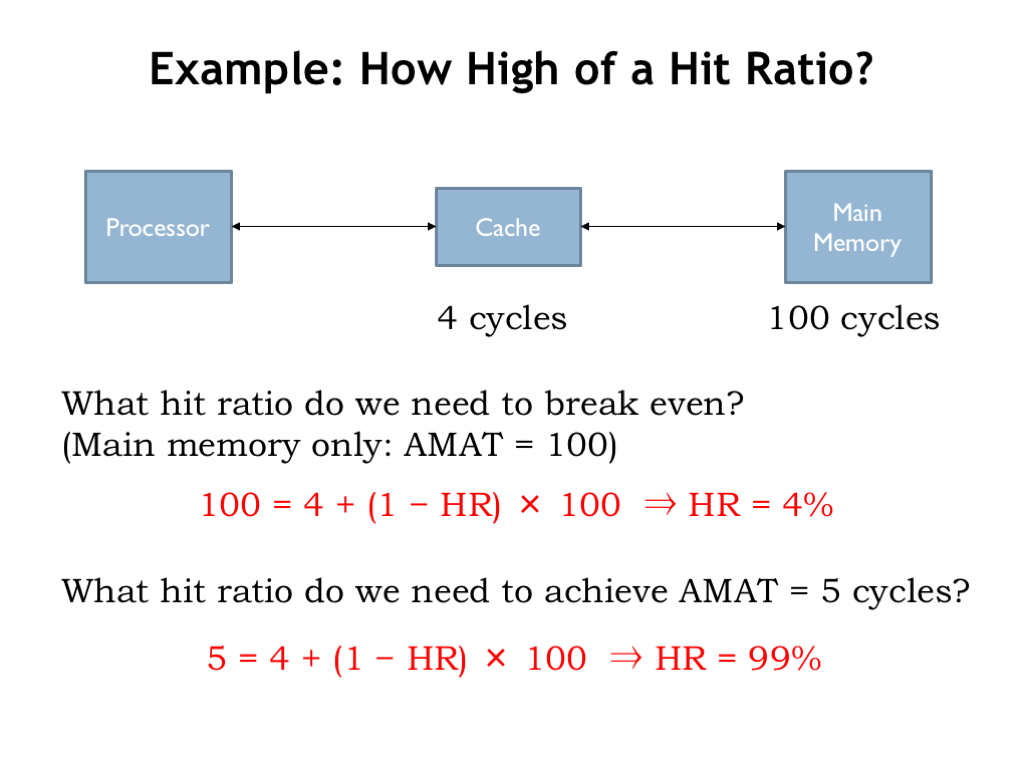


cpu从cache拿数据4ns，从主存拿数据44ns，cpu如何应对不同的内存访问时间。要么等着，现代处理器超线程技术可以在这段时间执行其他程序的指令。

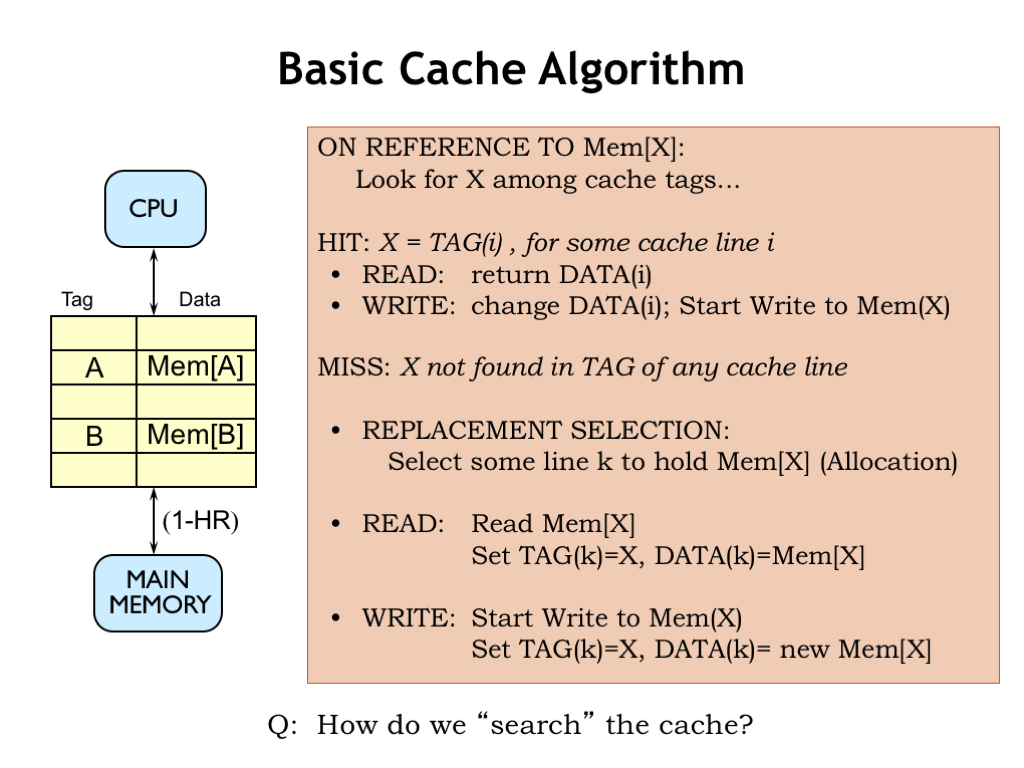
## 缓存指标



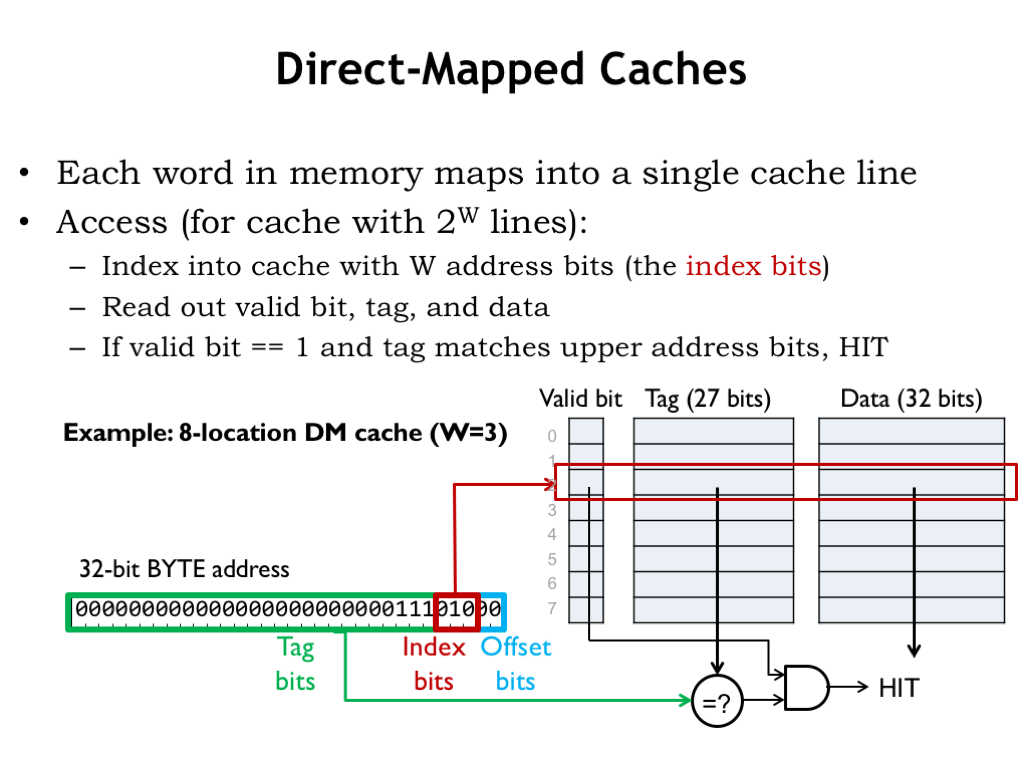
## 例子：命中率要有多高



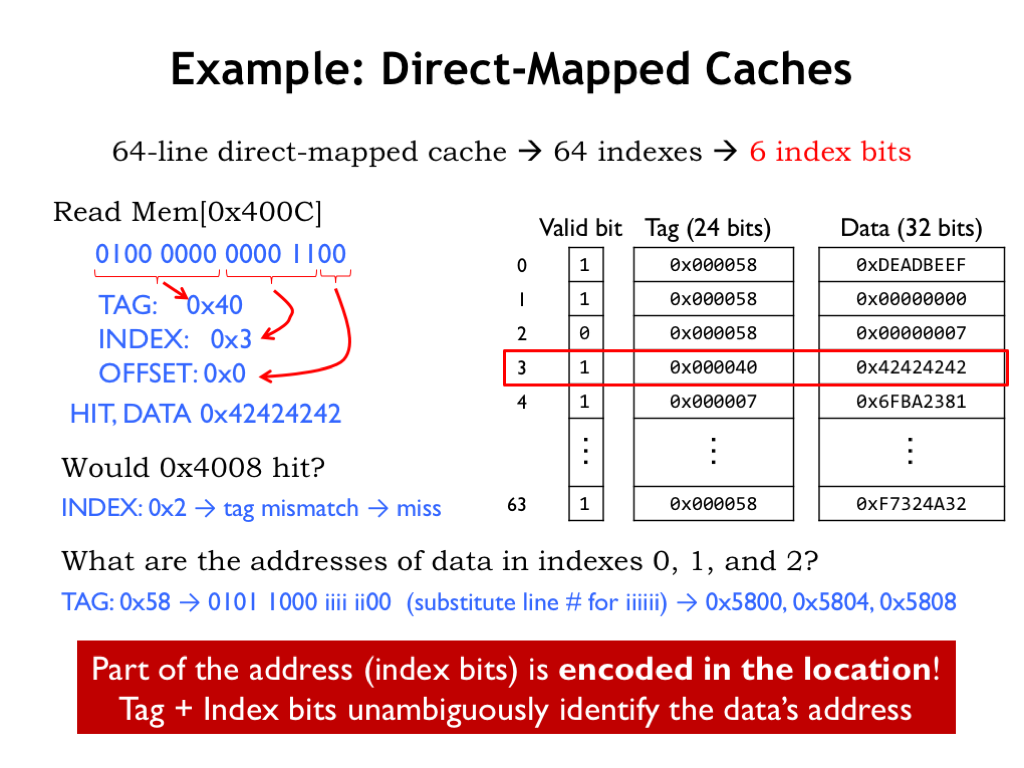
## 基本缓存算法



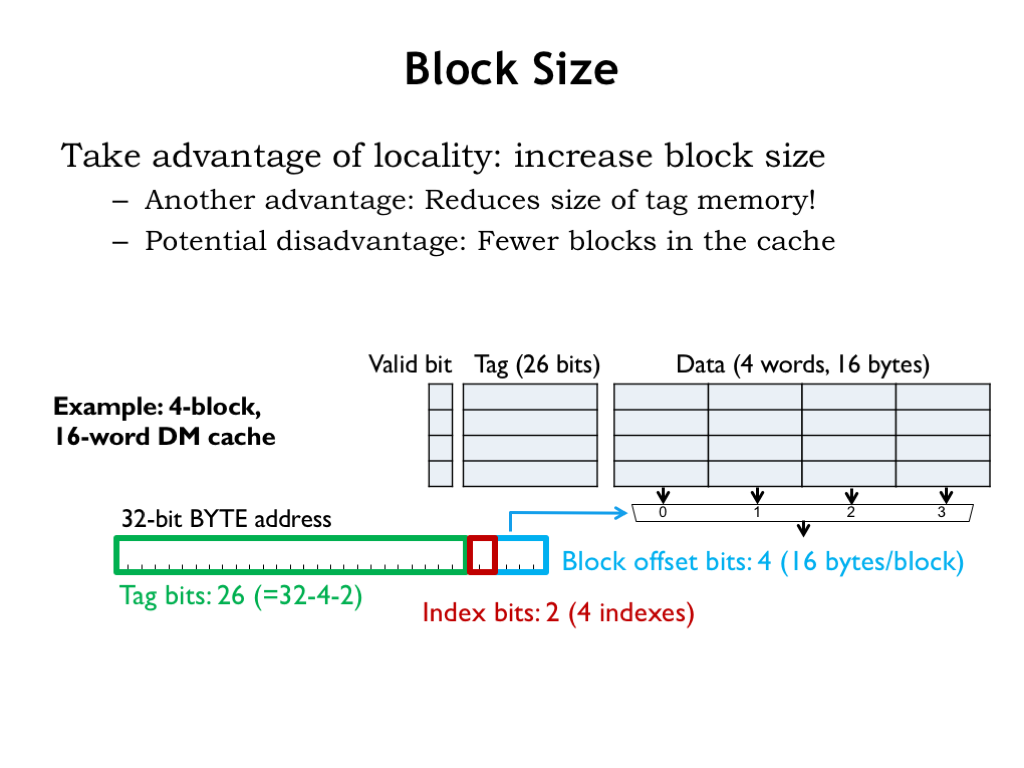
## 直接映射缓存



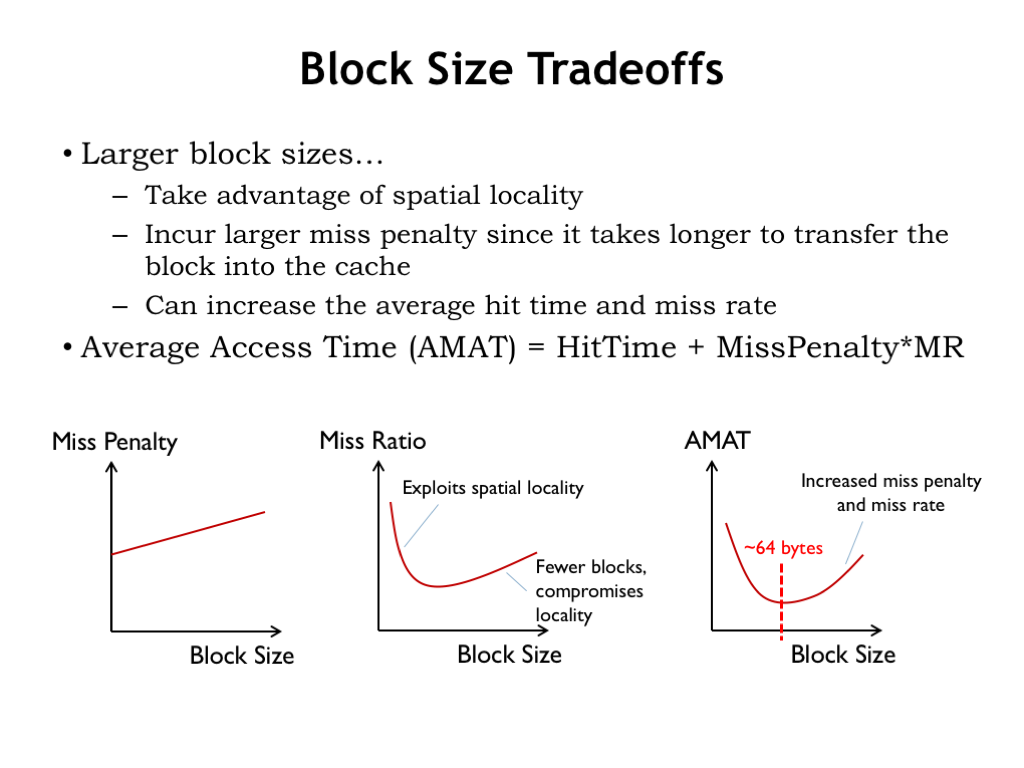
## 例子：直接映射缓存



## 块尺寸



## 块尺寸权衡

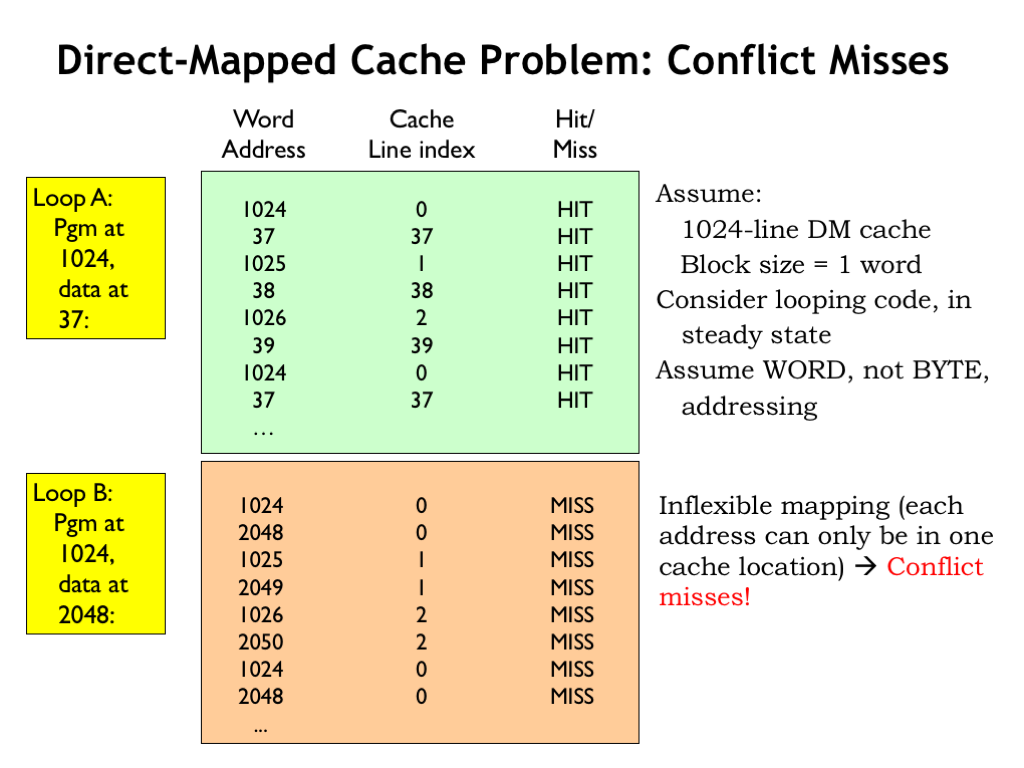


miss处罚：随着块尺寸增大而增大；

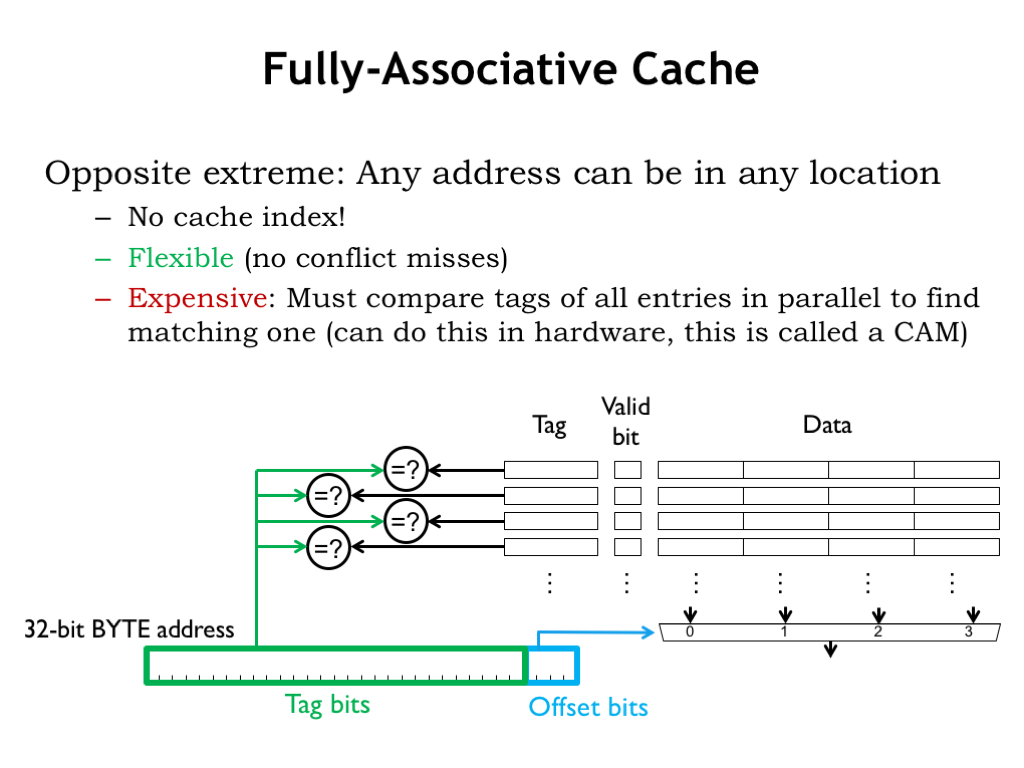
miss比率：随着块尺寸增大先降低、后升高；

amat：随着块尺寸增大先降低、后升高；

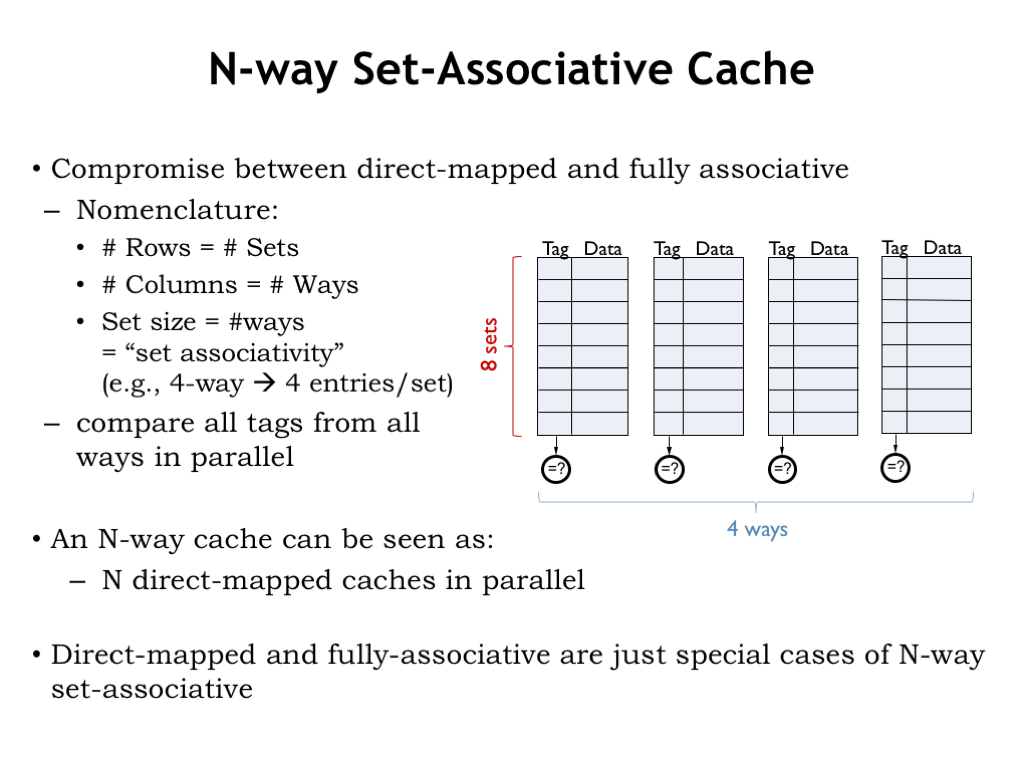
## 直接映射缓存问题：conflict miss



## 全相联缓存

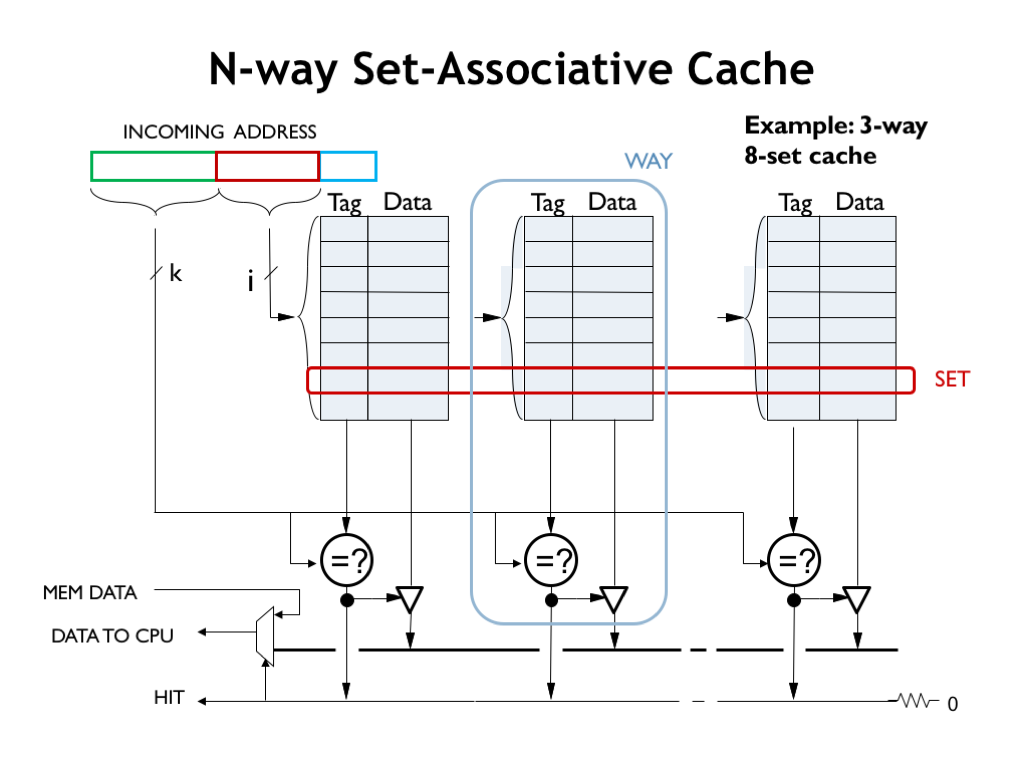


## N路组相联缓存1

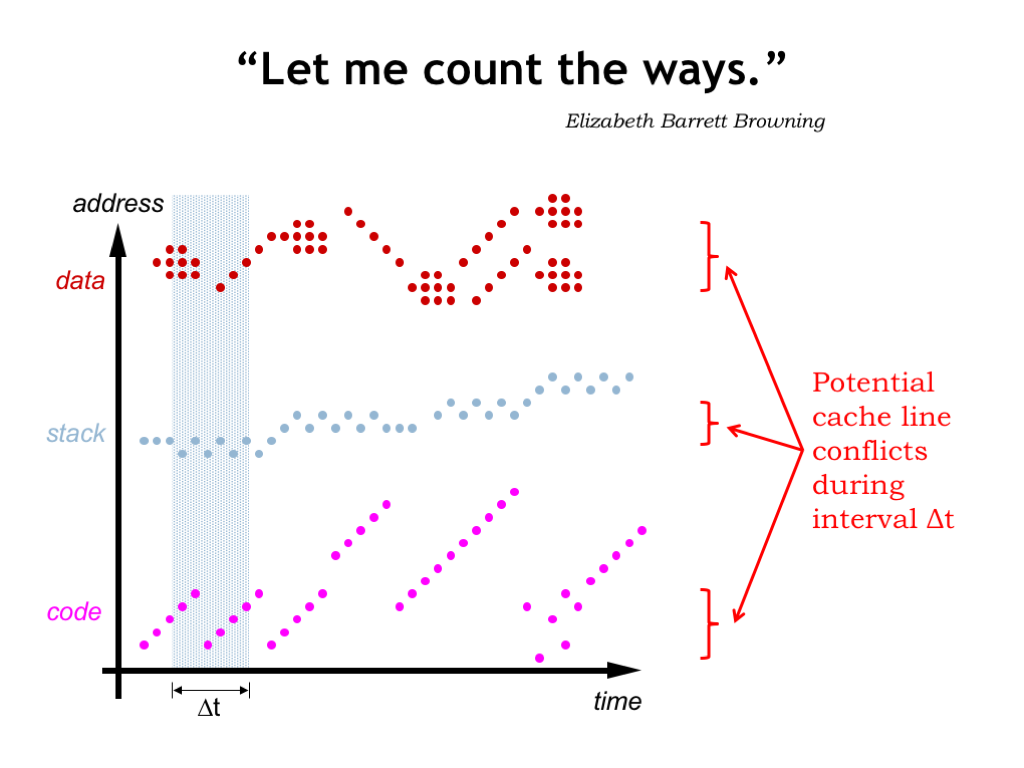


地址的index位定位到某个set（某行），再通过N个比较器，比较N个tag和地址的tag位。

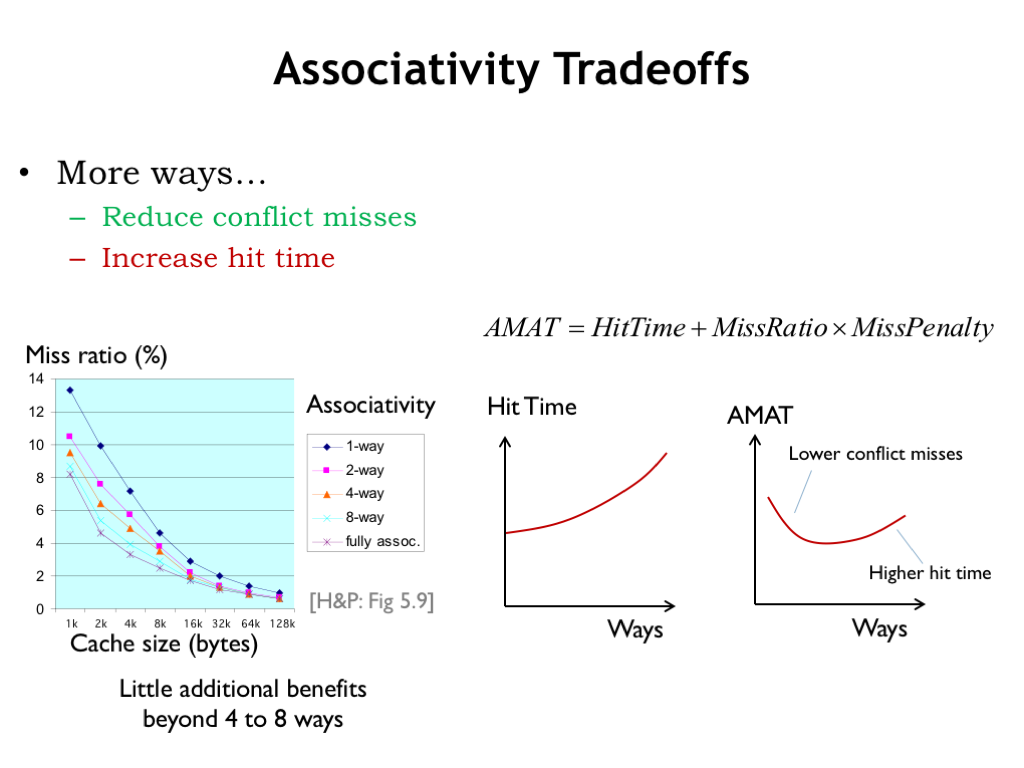
## N路组相联缓存2



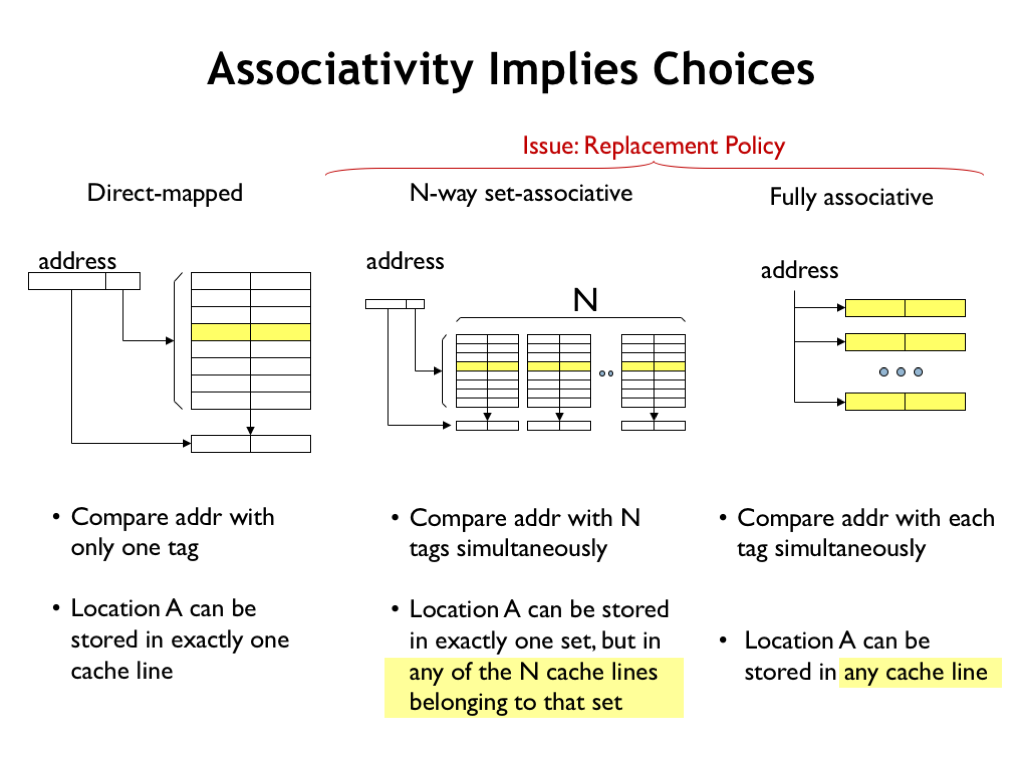
## 让我们算下路数？



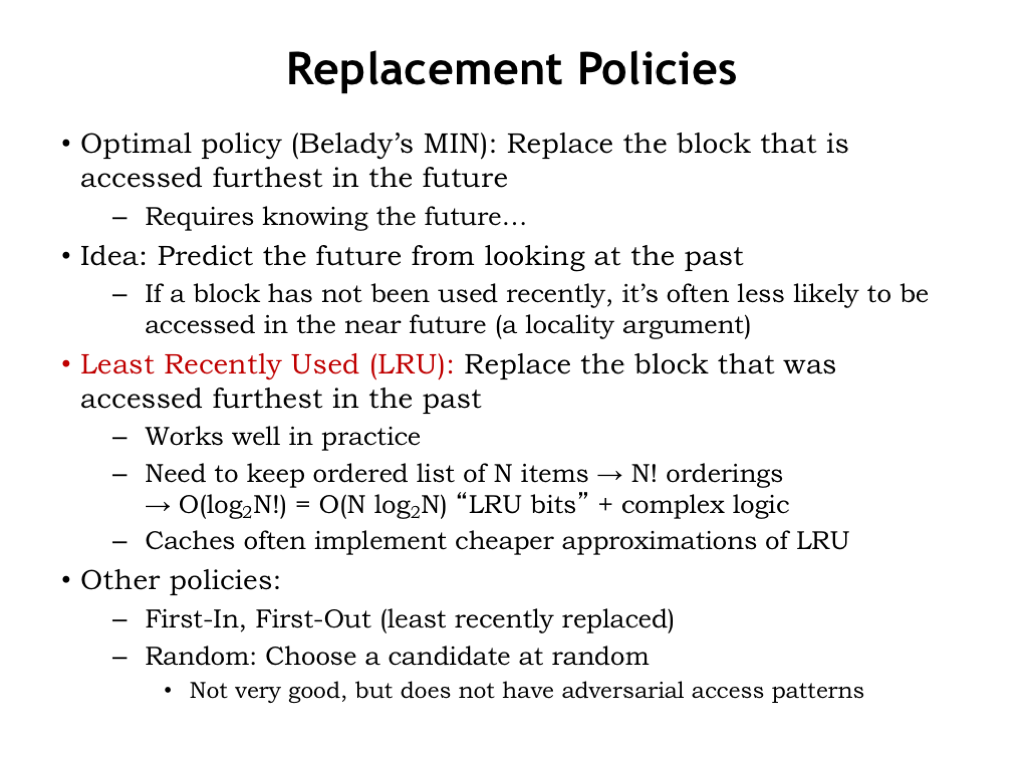
## 相联权衡



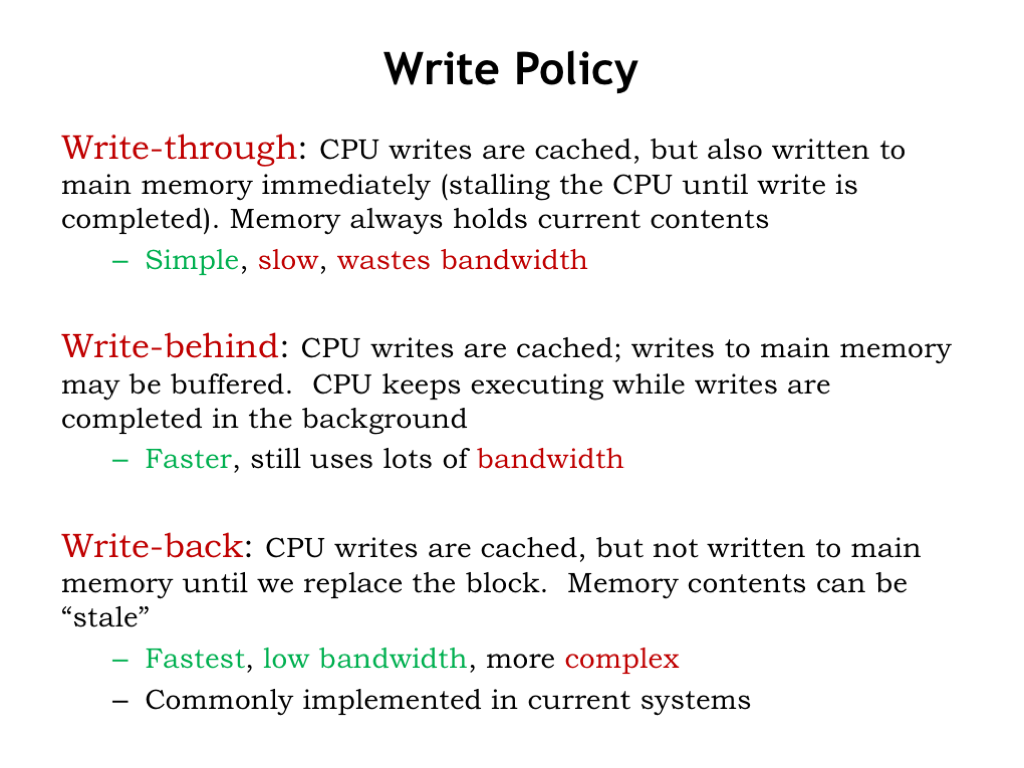
## associativity implies choices



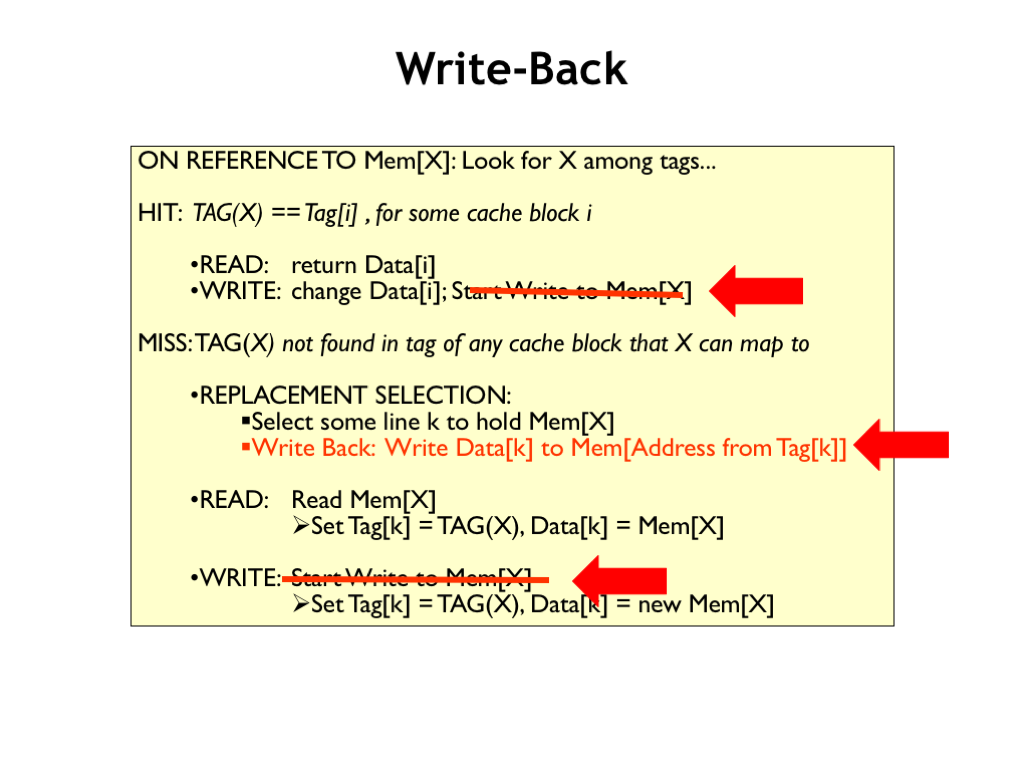
## 取代策略



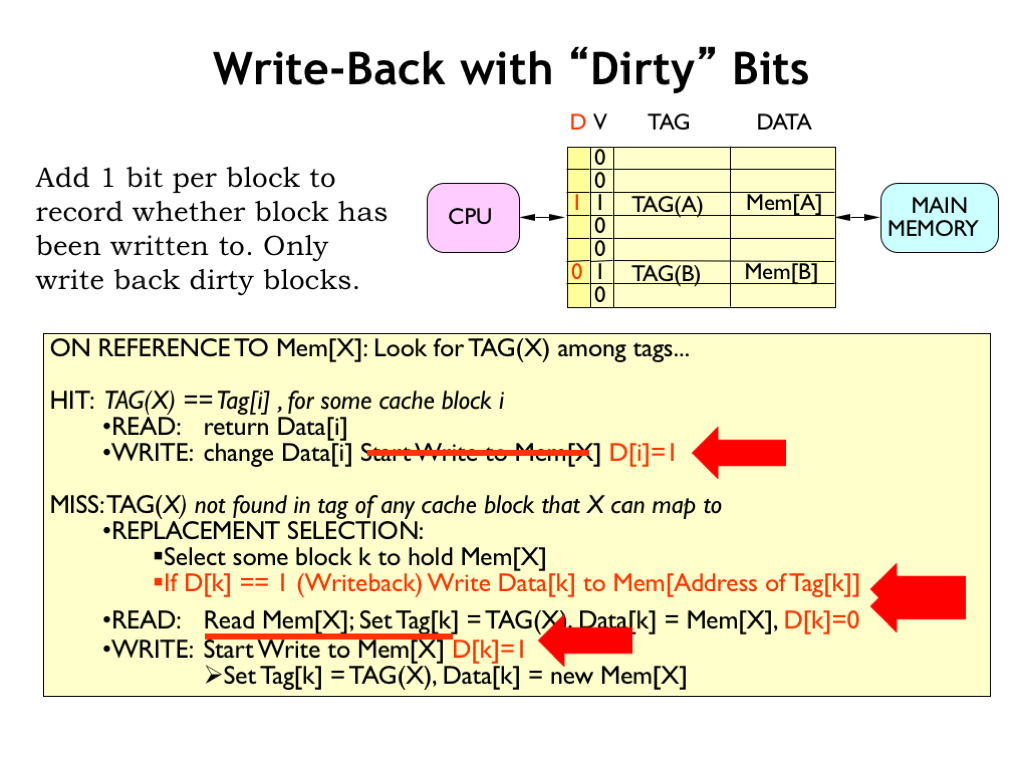
## 写策略



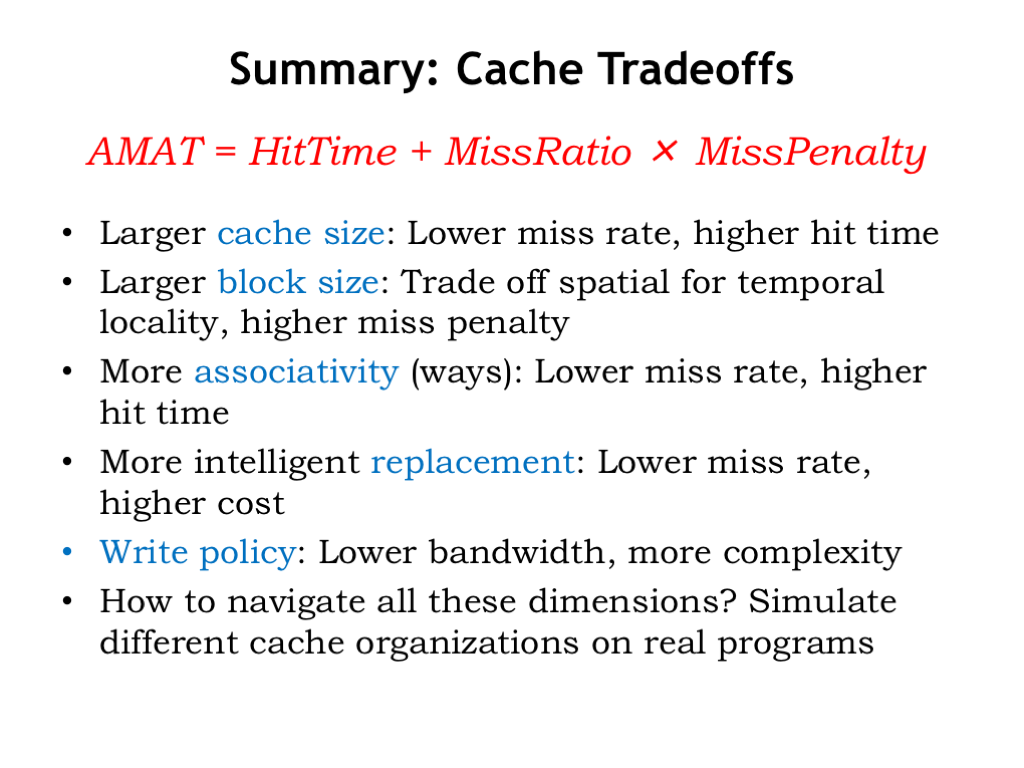
## 回写



## 带“脏”位的写回



## 总结：缓存权衡



# 问题

## 问题1

A：0.96

B：+1、unchanged、0.5

C：0x123414