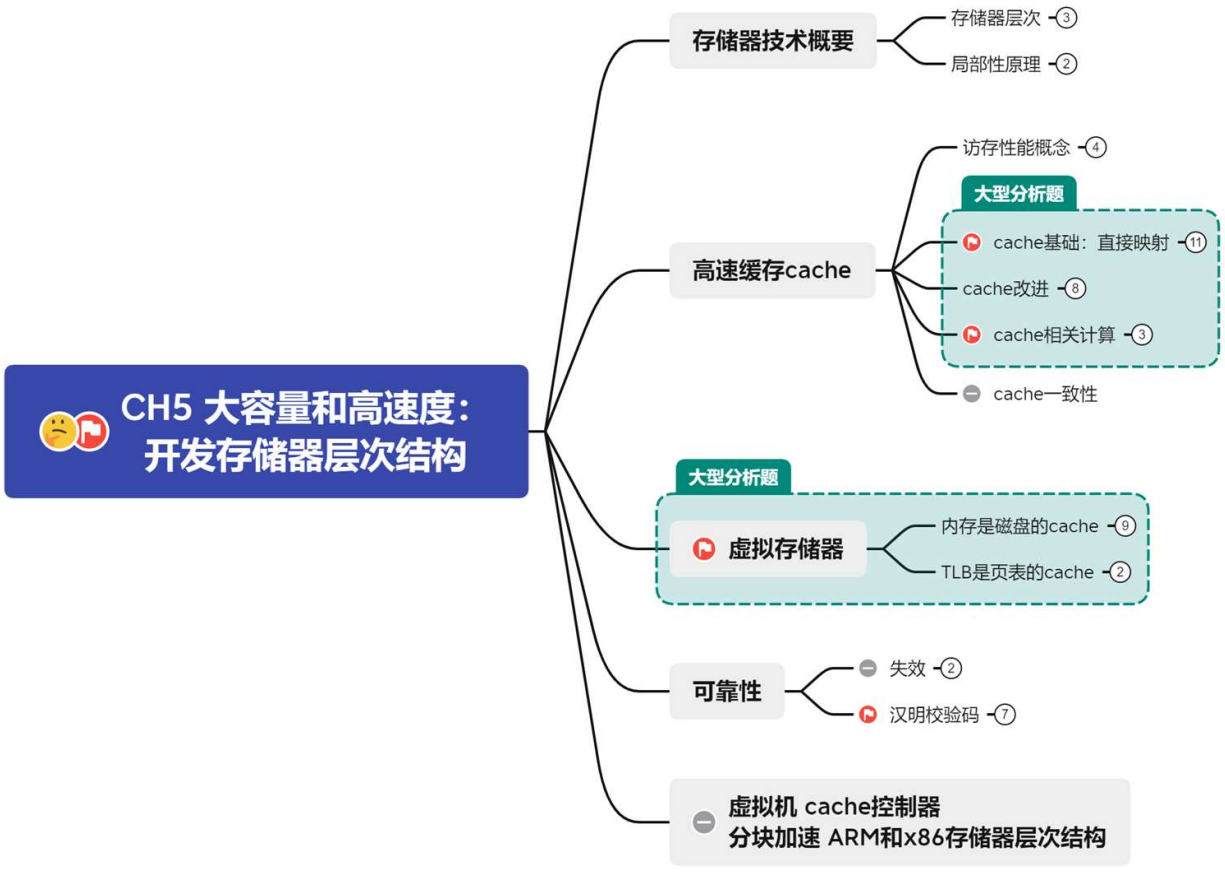
CH5 存储器层次结构

全名 大容量与高速度：开发存储器层次结构

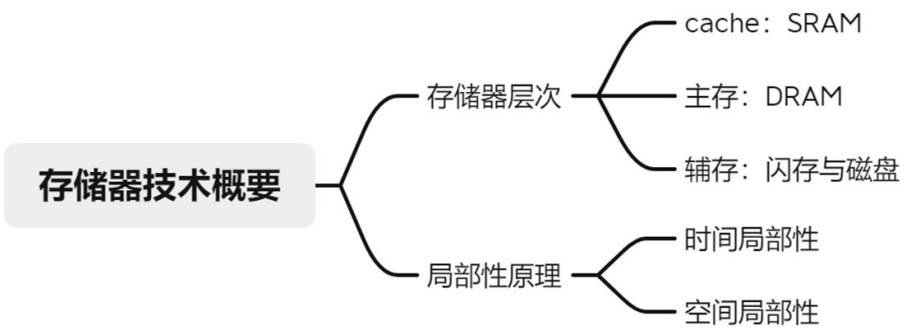
课程基于
《计算机组成与设计：硬件/软件接口》5e
Patterson & Hennesy 著

章节导图


CH5 大容量和高速度：
开发存储器层次结构
存储器技术概要
存储器层次 3
局部性原理 2
高速缓存cache
访存性能概念 4
大型分析题
cache基础：直接映射 11
cache改进 8
cache相关计算 3
cache一致性
大型分析题
虚拟存储器
内存是磁盘的cache 9
TLB是页表的cache 2
可靠性
失效 2
汉明校验码 7
虚拟机 cache控制器
分块加速 ARM和x86存储器层次结构

第一部分

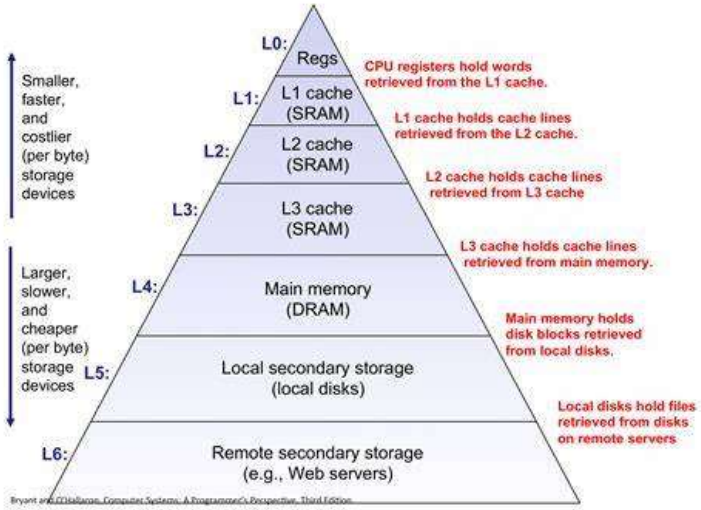
存储器技术概要



存储器层次：cache和内存

在存储器层次结构中

顶端：快贵小 → 底端：慢廉大 为解决什么问题而产生？



L1~L3：高速缓存cache通常集成在CPU中

采用静态随机访问存储器（SRAM）集成电路

由双稳态触发器制造

每B由6~8个晶体管组成，硬件规模较大

L4：内存，采用动态随机访问存储器（DRAM）集成电路

使用电容保存电荷，进而存储数据

每B仅使用1个晶体管，硬件规模远远小于SRAM（密度大于SRAM）

由于电荷只能短暂留存，需要周期性地将一行上的数据读出后重新写入，完成刷新

移动设备中的LPDDR5内存，全称为第五代低功耗双数据速率同步DRAM

SRAM、DRAM在断电后很快丢失数据，称为易失性存储器（volatile memory）

存储器层次：辅存

L5：二级存储器或辅存（内存则称为一级存储器或主存）

过去常使用磁盘，由一个覆盖着磁性材料的金属/玻璃盘片保存数据

读写磁头紧挨着盘片，磁头中的电磁线圈通过感应磁性材料的磁场方向进行读取

通过扭转磁性材料的磁场方向进行写入

每个盘片由数万条磁道组成，每个磁道又被分为几千个扇区

关于磁盘的更多术语和访问时间计算，请参考操作系统课程，这里不再深入

现在个人设备中更多使用闪存（flash）

闪存是一种集成电路制造的电可擦除可编程只读存储器（EEPROM）

与磁盘相比速度更快、更坚固、功耗更低，但写入次数过多会产生损耗

机械硬盘属于磁盘，而U盘、固态硬盘都属于闪存

一些系统还会使用光驱或网络服务器作为更低一级的辅存

磁盘和闪存断电后不丢失数据，称为非易失性存储器（nonvolatile memory）

局部性原理

循环和顺序控制流在程序中广泛存在

程序正在执行某条指令

如果这条指令位于循环体中，那么这条指令很可能不久后被再次访问——时间局部性

如果在循环体或顺序指令流中，那么将要执行的指令往往地址相近——空间局部性

时间局部性与空间局部性统称为局部性原理

数据是否和指令一样满足局部性原理？

程序在某一时间真正需要的指令/数据往往只占整个程序内存空间的一小部分

并且程序在使用一个指令/数据字的时候，即将使用的指令/数据通常在内存中相近的位置

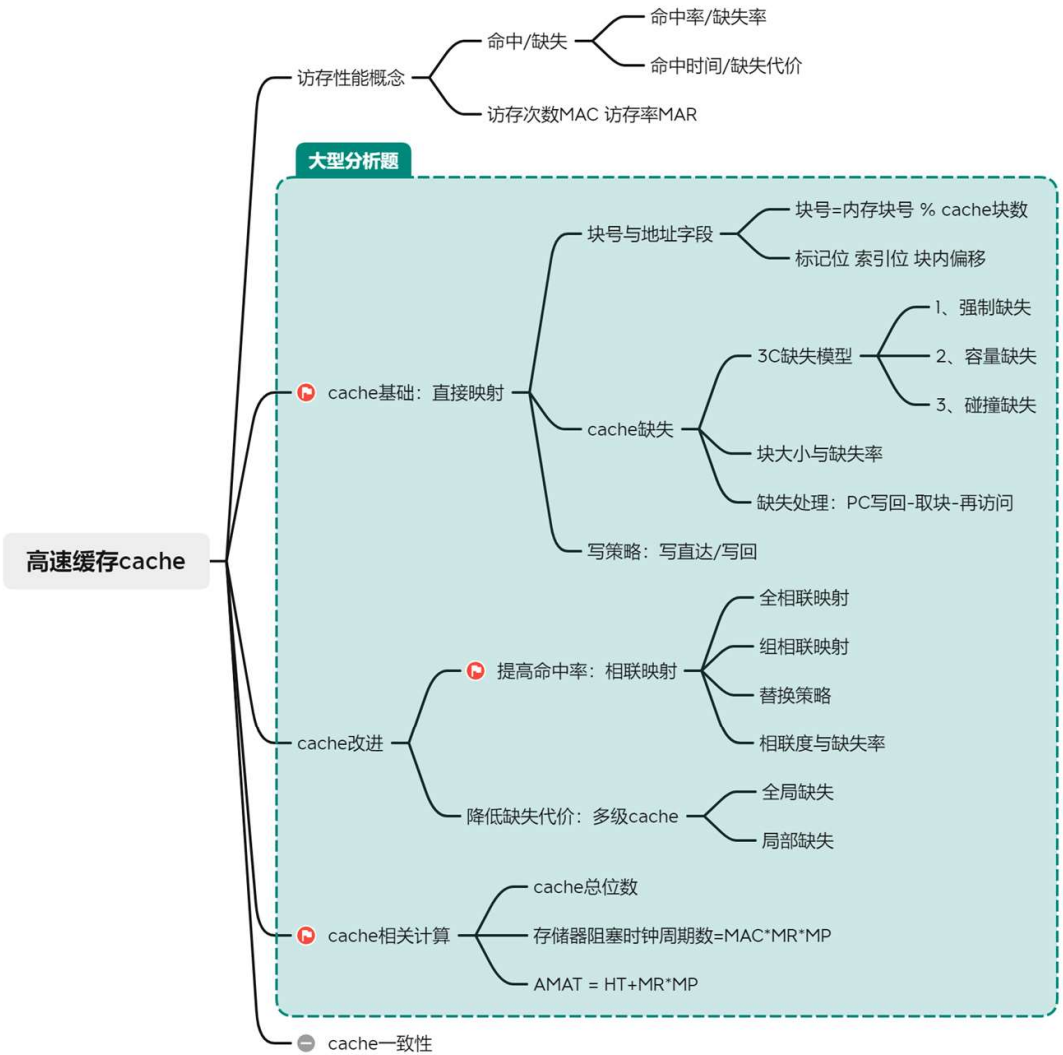
因此，可以将内存中的一小块放入更小、更快的上级存储器——cache中

让CPU快速取用数据

cache集成在CPU中且容量远远小于内存，到CPU的电路更短、更简单，从而更快

第二部分

高速缓存cache



访存性能概念：命中与缺失

访存即访问内存，分为读取内存和写入内存

访存指令 = MEM-reg数据传送指令 = L-S指令，即lw和sw指令

CPU访问内存时，都会优先询问cache是否保存着所需数据

如果访问数据在cache中，就称为一次cache命中（hit）

但是这种预测不会永远满足

例如程序跳转到很远的距离之外，cache并不包含所需数据

CPU则需要访问慢得多的内存

此时产生一次cache缺失（miss）

命中/缺失占访存次数的比例叫做命中率（hit rate）/缺失率（miss rate）

CPU访问内存数据，要么在cache中访问命中，要么缺失，因此

缺失率=1–命中率 MR=1-HR

访存阻塞周期数

CPU访问cache的时间称为命中时间（hit time），通常只有1T

CPU访问内存比访问cache多出来的时间称为缺失代价（miss penalty），长达数百T

*注意，访问内存比直接访问cache多出来的额外开销，才是缺失代价

例如，CPU判断命中并从cache取得数据需要1T，判断缺失并从主存取得数据需要101T

缺失代价为他们的差值100T，不包含无论如何都要花费的判定是否命中的时间1T

为便于后续分析

定义一个程序的L-S指令数目为访存次数MAC（memory access count）

定义程序中L-S指令占总指令数的比重为访存率MAR（memory access rate）

为了评价存储器性能，我们将程序执行的周期数cycles

分为CPU执行周期数+访存阻塞周期数两部分，其中

访存阻塞周期数 = 访存次数 x 缺失率 x 缺失代价 $cycles_{mem}$ = MAC x MR x MP

直接映射：块号

内存和cache被划分为一些大小相同的块，内存中的块数远大于cache中的块数

假设cache拥有8个块（编号0~7），内存拥有256个块（编号0~255）

CPU访问的内存地址是内存中的24号块、35号块

那么，访问的内存块分别应放入几号cache块中？

cache块号 = 内存块号 % cache块数

32位地址支持4GB内存，此时内存拥有1G即2^{30}个块，假设cache拥有1K即2^{10}个块

对访存地址0000 0000 0000 0000 0000 0000 0011 1100

和0000 1111 1111 1111 1111 0000 0100 0000

二进制块号分别为多少？分别应放入几号cache块中？

对2^n块的cache，cache块号 = 内存二进制块号的后n位

直接映射：内存地址字段

根据上面的分析，我们可以将内存地址分为3段：

①cache块号之前；②cache块号；③cache块号之后

cache块号之后的2位用于指定访问块中具体哪个字节，称为③块内（字节）偏移

*书上称为块偏移，容易和第二章讲的字偏移/字节偏移混淆

cache块号用于决定放入哪个cache块，相当于一个标签，称为②索引位（index）

一个2^n块的cache，对应2^n个索引号，内存块号的低n位来表示

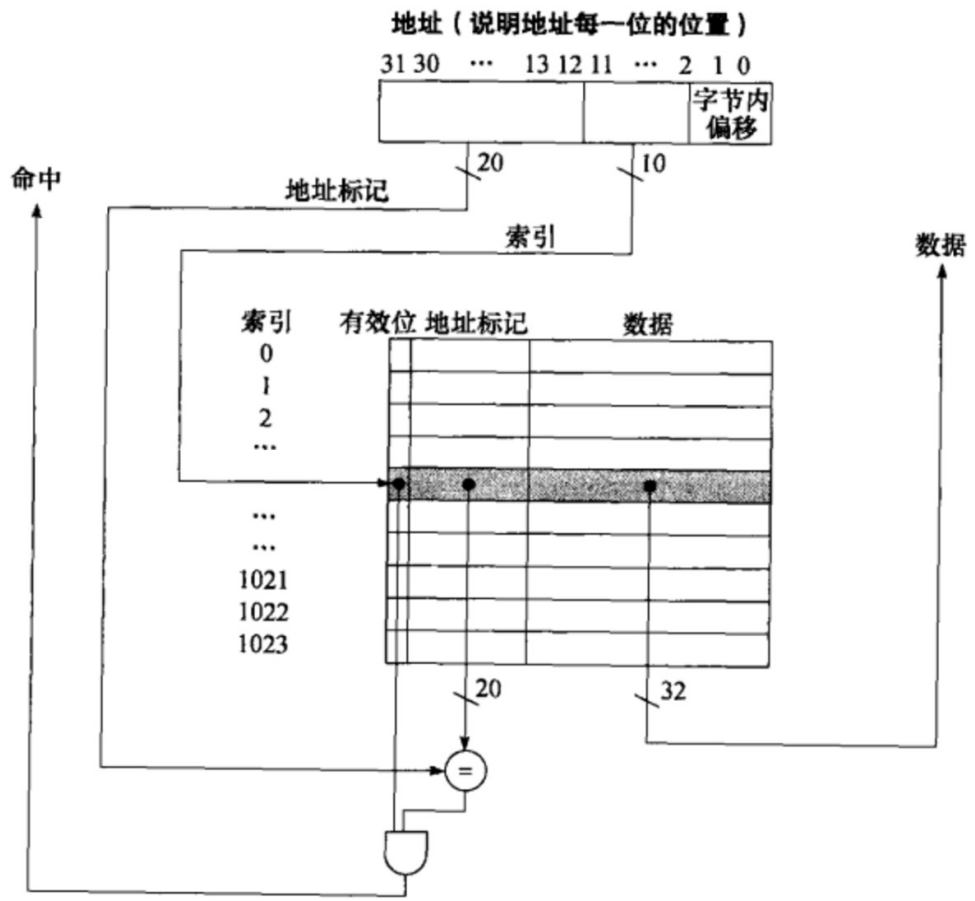
但是，cache为了确定一个块具体是哪个内存块

除了索引位，还需要索引位之前的高位地址

这个高位地址结合索引号，可以唯一标记一个内存块，故称为①标记位（tag）

0000 1111 1111 1111 1111 0000 0100 0000

直接映射硬件 有效位



假设计算机刚刚切换运行程序
此时，cache中的所有数据
都是上个程序的数据
对新程序而言是无效数据
cache清零后也是无效数据
只有新程序自己从内存
取到cache的数据
才是真正有效有用的数据

因此，cache中需要添加一位
有效位（valid）
仅当有效位为1且标记位对应时
cache才能访问命中

直接映射：cache位数计算

理清内存地址地段（题目没给地址位数则默认32位），cache位数计算迎刃而解！

*处理器访存读写的数据通常为一个32位字，一字一块根本不能利用空间局部性

一个有16KB数据的cache，块大小为4个字，地址为32位

该cache总共需要多少字节？

cache总容量是实际数据容量的多少倍？

$16\times2^{10}=2^{14}$ B

$\frac{16\times2^{10}}{16}=n$

Tag 18 | index 10 | offset（定位到字节） 4

$\frac{18.375}{16}=1.148$

$\frac{1+18}{8}\text{B}\times2^{10}+16\text{KB}$

$=18.375\text{KB}$

如果题干换为“一个16KB的cache”，默认是16KB数据的cache

一定要注意位和字节的转化！

直接映射：访存序列示例

对一个初始为空的8块cache，访存块号依次为22，26，22，26，16，3，16，18，16

分析每次访问的cache状态，注意块替换的情况

6 2 6 2 0 3 0 2 0

M M H H M M H M H

cache块号	内存块号
0	Mem[16]
1	
2	Mem[26] → Mem[18]
3	Mem[3]
4	
5	
6	Mem[22]
7	

直接映射：缺失分类3C模型 块大小与缺失率

根据产生原因，缺失分为以下三类：

①首次访问cache中没有的块必然产生的缺失

称为冷启动强制缺失（compulsory miss）

②由于cache容量不能容纳程序执行需要的所有块

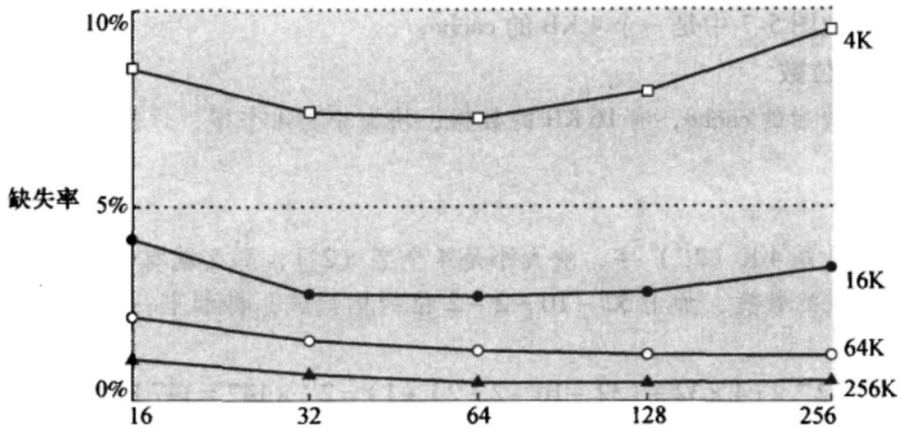
部分块被替换后再次调入cache

称为容量缺失（capacity miss）

③多个内存块竞争映射到同一个cache块

导致仍需使用的块被替换，称为冲突碰撞缺失（conflict miss）

三种原因导致的缺失统称为3C模型



适当增加块大小（同时也会减少块数）可以更好利用空间局部性，显著减少强制缺失

加大容量可以改善容量缺失

但是随着块数不断减少，则会因为竞争替换过于频繁，增加冲突碰撞缺失

并且，更大的块意味着传输一块数据的时间更长，缺失代价上升

直接映射：缺失处理和写策略

cache访问缺失处理的步骤为：

①将PC+4-4（即当前指令的地址）写回PC，并阻塞处理器

②访问内存，将内存块写入cache

③再次访问cache并命中

当CPU把新数据写入内存块时，又要把数据写到cache，保持内存和cache内容一致

方式一：CPU写cache时同时开始写内存，称为写直达（write through）

这种方式的处理器开销也接近访存时间，意义不大

因此，可改进为写缓冲，CPU较快写入缓冲后执行别的任务

由缓冲慢慢将数据写到内存 写缓冲可能面临什么问题？

方式二：CPU只写入cache，仅当这个cache块被索引位相同的其他块替换出去时

才将修改后的cache块写入内存，称为写回（write back）

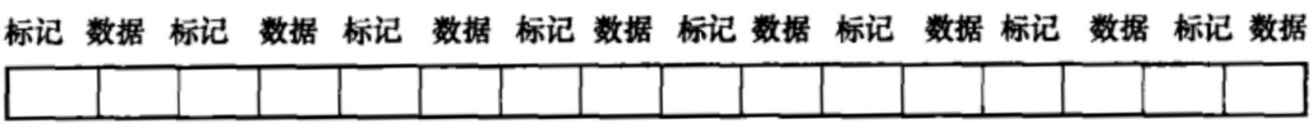
*写缓冲只是对写直达的必要改进，不是写直达和写回之外的方式三

写回通常能够减少写内存的次数、从而提高性能，但实现起来复杂得多

全相联映射

直接映射中，一个内存块只能映射到唯一的cache块，实现比较简单
另一种极端的情况是，内存块可能映射到任何一个cache块，称为全相联映射

第一个内存块可进入任何一个cache块，只要cache没满，其他块就可以见缝插针
如果cache已满，则替换掉最长时间没有使用过的内存块
这种替换算法称为最近最少使用（least recently used, LRU）
和直接映射相比，要使用的块更不容易被替换出去，缺失率更低
全相联有没有索引位？全相联中的碰撞缺失实际上转化成了哪种缺失？
但是查找时需要比较每一块的标记位，开销过大，只适用于块数较少的cache



组相联映射：组号与地址字段

折中的方法是，对cache块进行分组，一个内存块直接映射到一个组

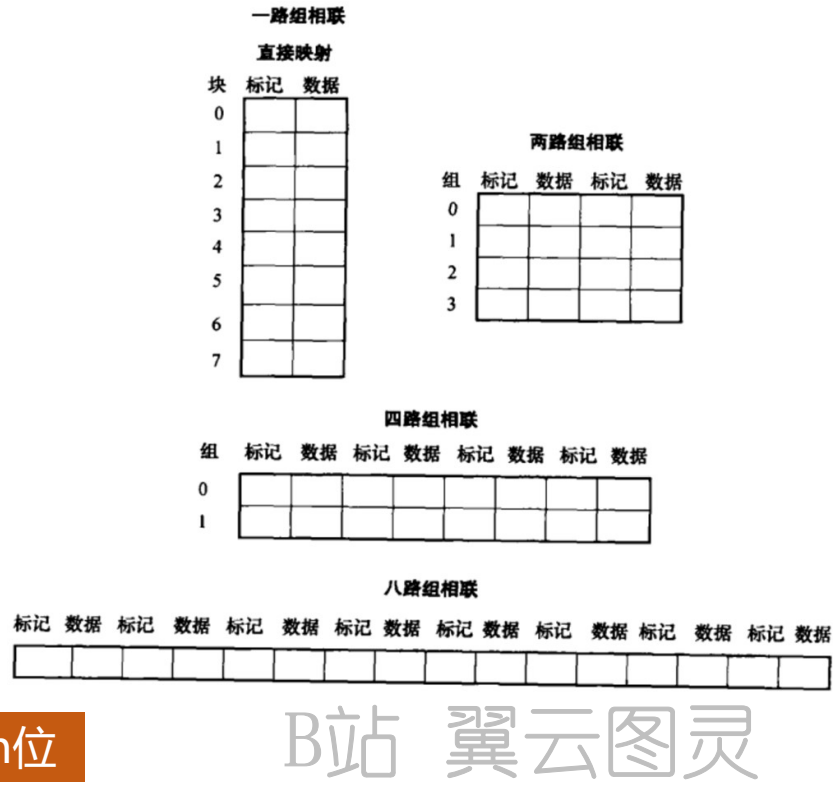
但分配到组内的哪一块较为自由，也就是在一组之内全相联

这样的映射策略称为组相联映射

一组包含n块则称为n路组相联，其相联度为n

直接映射可以理解为1路组相联

同样，全相联映射可视为相联度=块数的组相联



组相联中的内存块映射到那一组

计算方法和直接映射相似

cache组号 = 内存块号 % cache组数

组号同样由块号低位中的索引位确定

对2^n组的cache，cache组号 = 内存二进制块号的后n位

组相联映射：缺失处理、写策略、替换策略

全相联和组相联的缺失处理过程和直接映射相同
也都可以采用写直达和写回两种写策略

但是，全相联和组相联引入了一个新的问题：
内存块可以放入全相联cache唯一一组中的任何一块/组相联cache某组中的任何一块
如果这一组已经被其他内存块占满，那么新取入的块应该放入哪一块？
理想情况是，留下后续仍将使用的块，替换掉之后不再使用的块
但是这种预测需要的信息太多、处理过程太过复杂

我们可以记录历史使用状况，认为最久没被使用的块，最不可能被再次使用
选择它替换出cache，这种替换算法叫做最近最少使用（least recently used, LRU）
我们可以用1位来编号记录2路组相联一组中最久没有使用的块
用2位来编号记录4路组相联一组中最久没有使用的块，以此类推，这个编号称为LRU位
实际上LRU的开销仍然较大，很多4路组相联的系统也只是近似实现LRU

组相联映射硬件 相联度与缺失率

随着相联度的提高

cache块更不容易产生碰撞缺失

全相联更是将这种可能性降到最低

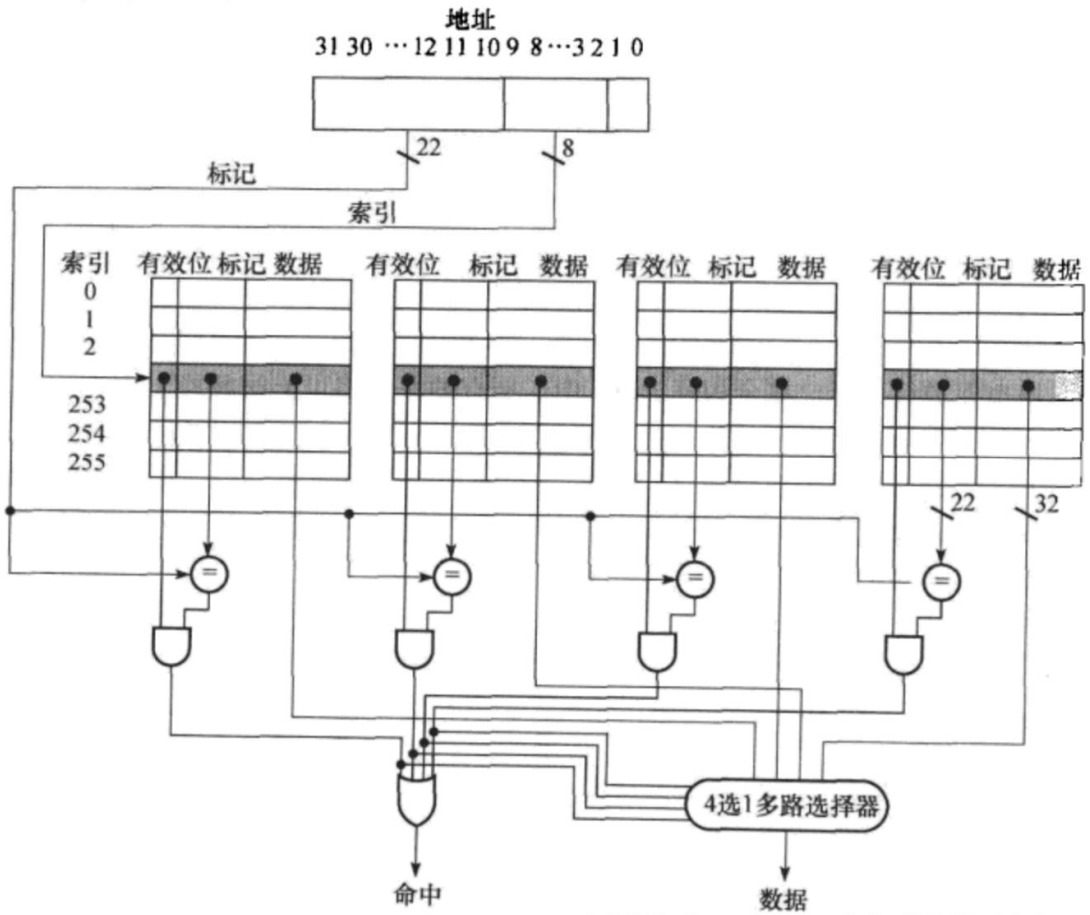
但是，提高相联度也意味着

每次访问cache时

比较标记位、选择数据的开销增大

实际cache的相联度通常为2或4

假设一个cache有4096个块，块大小为4个字，地址为32为，请计算两路组相联、全相联中cache的总位数。



cache小结

4个问题：

①块可能被放在何处？	直接映射/组相联/全相联
②如何查找一个块？	直接索引/比较/查表映射（虚拟存储器）
③如何选择替换块？	随机/LRU
④如何写入内存？	写直达（写缓冲）/写回

3种缺失及其应对方案：

①冷启动强制缺失	加大块容量
②容量缺失	加大cache容量
③冲突碰撞缺失	提高相联度

任何一种改进策略都会对其他某些方面产生负面影响，抵消总体性能的提升

因此，设计存储器层次结构是一门折中的艺术

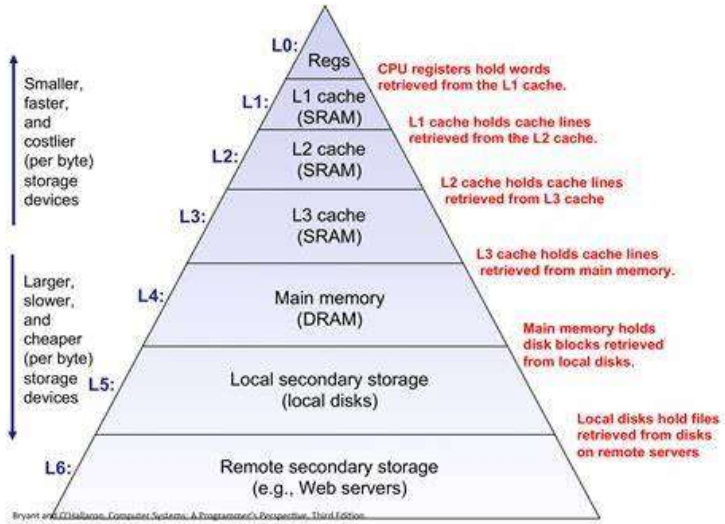
降低缺失代价：多级cache

cache产生的目的是缓解CPU和主存速度差距的矛盾，然而这种矛盾仍在加深
为此进一步细化了存储器层次结构，产生了多级cache

两级cache就是打鱼，更快贵小的L1 cache是粗眼网，更慢廉大的L2 cache是细眼网
鱼有大中小三种，分别有70条、20条、10条

先用粗眼网打70条大鱼，L1 cache命中率70%
30条中鱼小鱼成了漏网之鱼，L1局部缺失率30%

再用细眼网打20条中鱼，L2 cache命中
10条小鱼又跑了，L1 L2全局缺失率10%
粗眼网没打到、漏给细眼网的30条中
跑了10条，L2局部缺失率1/3



cache性能评价：AMAT MR和MP讨论范围

平均访存时间（average memory access time）有时用周期数表示即可，不必化成秒

平均访存时间 = 命中时间 + 缺失率 x 缺失代价
AMAT = HT + MR x MP

顾名思义，AMAT只考虑访存的时间，也就是只考虑访存指令花的时间
跟访存指令占总指令的条数MAR无关

我们所说的缺失率仅针对访存指令 回忆一下，哪个公式可以说明这个规定？
但是计算CPI时需要考虑所有指令
因此，需要区分（默认只针对访存指令的）缺失率MR和针对全指令的缺失率MR_{all}

一级缺失代价MP_{L1}只考虑一级缺失必然造成的、访问二级cache的额外开销时间
二级缺失代价MP_{L2}即为二级缺失必然造成的、访问主存的额外开销时间
后面会看到，高级存储器中的内容一定是低级存储器的子集，二级缺失意味着全局缺失
这里的“开销时间”在计算CPI时用时钟周期数表示即可

cache性能评价：CPI

处理器基本CPI为1，时钟频率4GHz。设访存指令占指令数的20%，一级cache缺失率为10%，速度可满足处理器全速运行；现增加一个二级cache，访问时间为5ns，全局缺失率为2.5%，访问主存的时间为100ns 。二级cache局部缺失率和该处理器的CPI为多少?

题干给出访存率MAR时

CPI = 基本CPI + 访存率 x (一级缺失率 x 一级缺失代价 + 全局缺失率 x 二级缺失代价)

$CPI = CPI_0 + MAR \times (MR_{L1} \times MP_{L1} + MR_{L2} \times MP_{L2})$

考虑题干划线部分改为：

一级cache中每条指令的缺失率为2%，二级cache使得访问主存的缺失率减少到0.5%

题干没有给出访存率MAR时

CPI = 基本CPI + 全指令一级缺失率 x 一级缺失代价 + 全指令全局缺失率 x 二级缺失代价

$CPI = CPI_0 \times MR_{all,\ L1} \times MP_{L1} + MR_{all,\ L2} \times MP_{L2}$

多核CPU中的cache一致性

假设一个双核CPU，每个核心拥有自己的cache，采用写直达策略
当A核向某一内存单元写入自己cache A的一个内存块时，B核对相应内存块进行读取
为了确保B核能读到A核写入这一块的内容
需要某种机制确保A核完成写入内存后，B核才能读取
从而确保两个处理器核心看到数据的一致性（coherence）

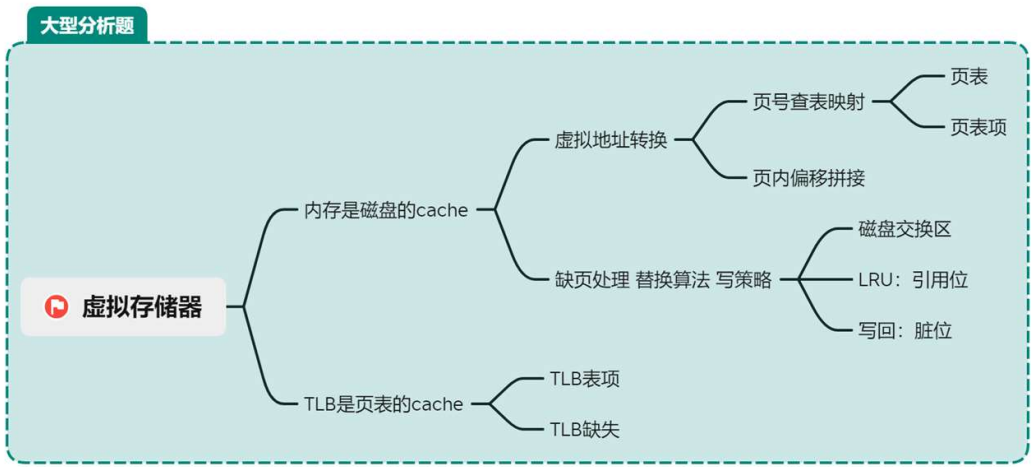
最常用的cache一致性协议是监听（snooping）协议
A核开始写入内存时，释放信息告诉其他核心：正在写入，请勿读取
B核监听到这种信息，从而暂停读取内存块以及cache B中的对应块
这称为写时无效协议（write invalidate protocol）
如果A核和B和需要同时写入一个内存块，则必须有某种机制
使得两个处理器的写操作一前一后
这种机制叫做写串行化（write serialization）

复习题

1、高速缓存、内存、辅存分别采用什么硬件技术?
2、时间局部性和空间局部性分别表示什么含义？为什么说局部性是存储器层次的基础?
3、访存指令有哪些别名？CPU是否直接访问内存?
4、直接映射的块号和组相联映射的组号如何确定?
5、内存地址可以分为哪三个字段？其中全相联映射没有哪个字段?
6、cache中每一项（每一行）的位数如何计算?
7、cache有哪4个基本问题？3种碰撞分别如何缓解?
8、提高相联度和采用多级cache分别改善了访存性能的哪方面?
9、访存阻塞周期数、AMAT分别怎么计算？为什么AMAT和MAR无关?

第三部分

虚拟存储器



广义cache 虚拟存储器的产生背景

狭义的cache指内存的上一级存储器即高速缓存

任意两级相邻的存储器中较高的一级都可以称为较低一级存储器的广义的cache

高速缓存是内存的cache

同样，内存是磁盘的cache，TLB快表是页表的cache

在虚拟机（virtual machine）中，多个操作系统共用一台硬件主机

需要某种机制，隔离不同系统的内存地址空间，避免交叉访问

20世纪70年代的内存容量通常只有几十KB

精细地划分、高效地装载和换出程序段，从而高效使用内存，是程序员的一大负担

虚拟存储器可以将各个操作系统相同的虚拟地址映射到不同的物理地址

同时借助硬盘的大容量，实现看起来无限大的内存

当今，内存容量很少成为编程的限制因素

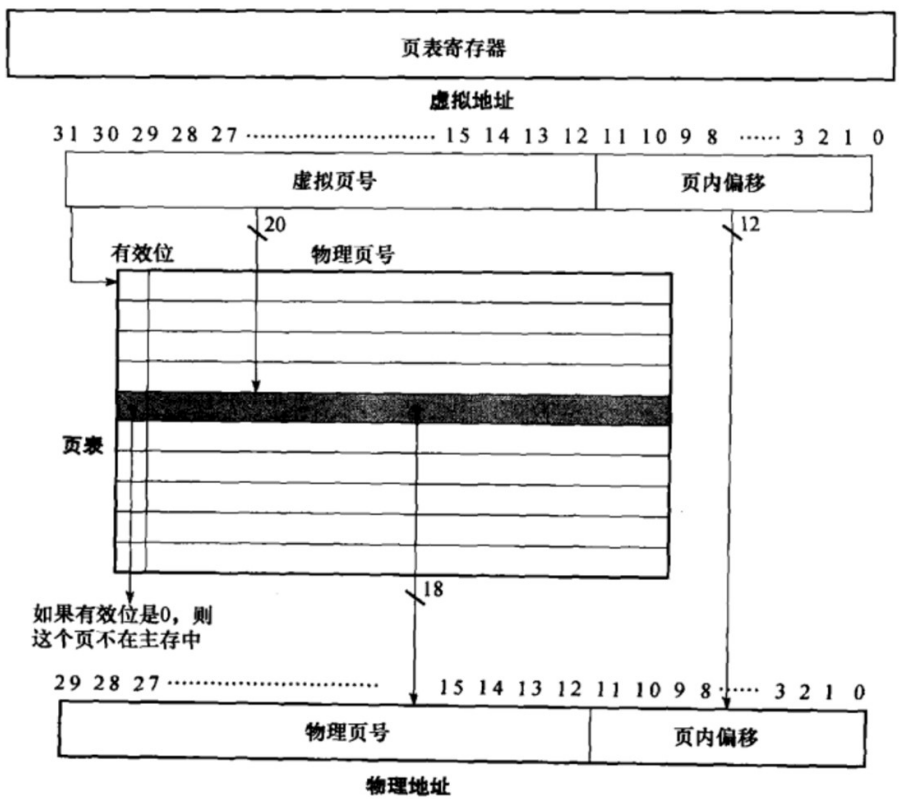
虚拟地址转换为物理地址

看起来“无限大”的内存意味着，虚拟内存的空间大于物理内存

因此虚拟地址往往使用更多的位数

例如，物理内存仅有1GB，用30位即可表示物理地址

但虚拟内存有4GB，需要用32位虚拟地址来表示



对应操作系统中的分页存储管理

磁盘和内存之间交换的块称为页（page）

对4KB的页，需要多少位表示页内偏移？

需要多少位表示虚拟页号和物理页号？

虚拟页号通过查询页表（page table）

映射到物理页号

由于内存和磁盘中页大小相同

页内偏移和物理页号拼接形成物理地址

页表和页表项 页表寄存器 缺页处理

每个进程都拥有一张页表，存放在自己的内存空间

页表首地址存放在页表寄存器中

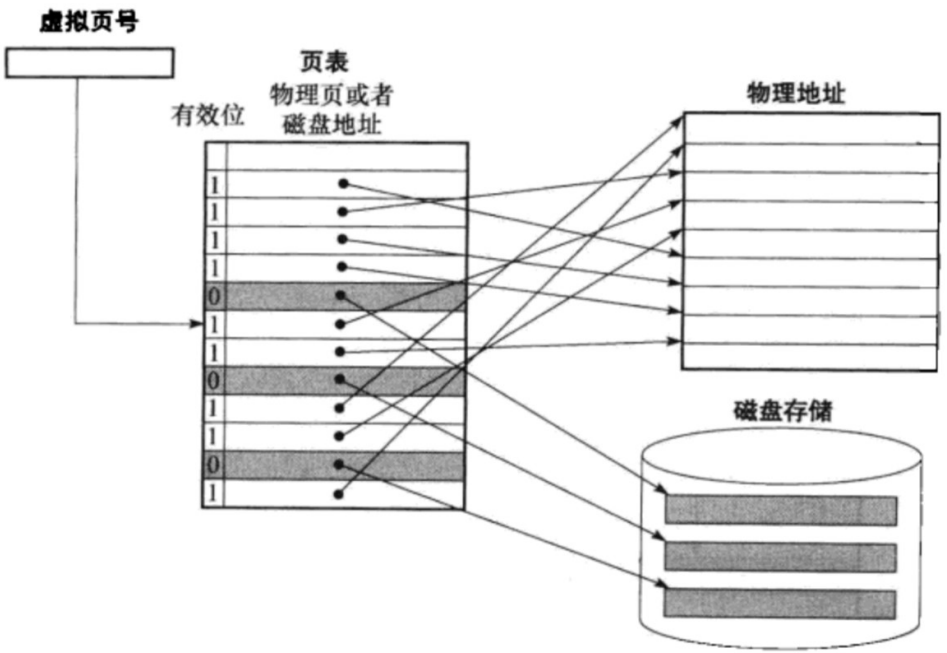
页表项记录了每个虚拟地址对应的物理地址

一张页表有多少项？

当进程数量较多时，页表规模会变得很大

可使用界限寄存器、哈希函数、多级页表等

策略减小页表占用空间



和高速缓存类似，页表中需要一个有效位

来标识访问的页是否在内存中

访问页不在内存时，产生一次缺页

需要读取磁盘用于扩充物理内存的部分

即交换区（swap space）

缺页由中断处理程序进行处理，属于软件方法

替换策略 写策略

当一个进程的物理内存空间占满时，从磁盘交换区中调入内存的块
需要替换掉内存中的某一块
我们使用引用位（reference bit）来近似实现LRU替换算法
操作系统定期将页表中的引用位清零，访问过的页引用位再设置为1
随后从引用位为0的页中选择一页进行替换

由于磁盘访问时间长达几千万个时钟周期，需要尽量减少访问磁盘特别是写入磁盘的操作
因此内存中的页只能使用写回机制写入磁盘
并且，还需要再页表中添加一个脏位（dirty bit）来标记写入过的内存页
当这个页被替换出内存时，只写回那些内存页被写入即脏位为1的页

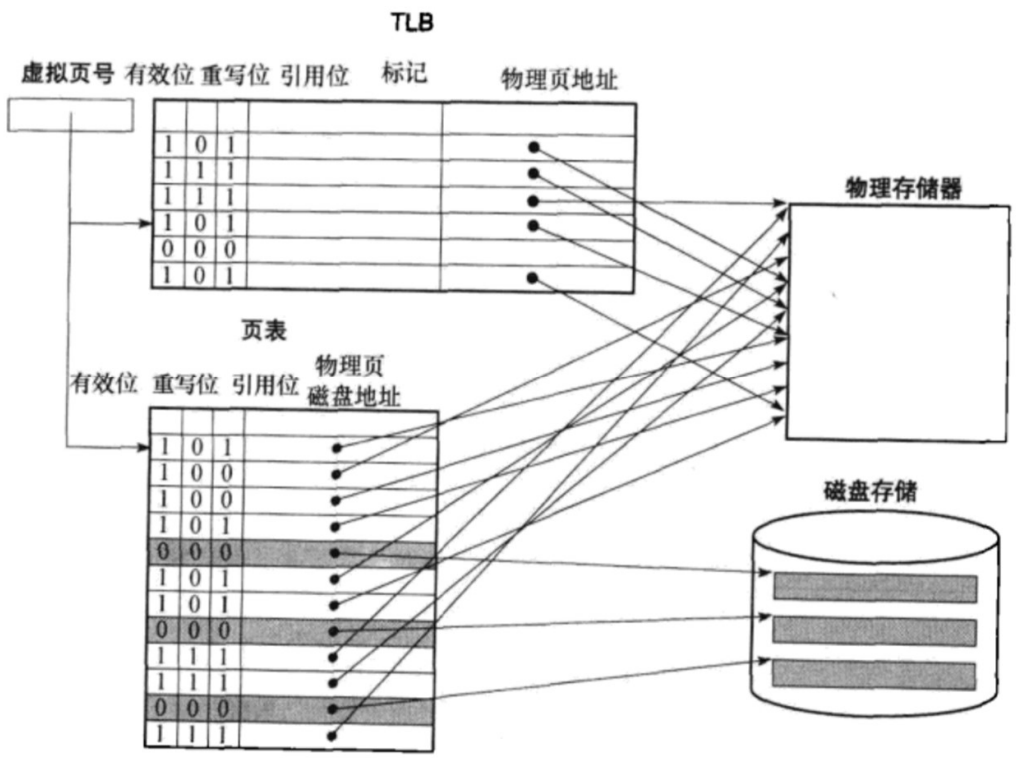
综上，页表项中除了有效位和物理页号
还需要额外的两个标志位——用于判断是否写回的脏位和用于近似实现LRU的引用位

TLB快表

页表位于内存中，CPU访存时，需要先拿着虚拟地址访存读取页表
得到物理地址，再真正进行访存（优先访问cache）
为了加快第一次获取物理地址的访存
引入了页表的cache
即TLB（translation-lookaside buffer）

页表将每一个虚拟页号作为索引
因此不需要类似于标记位的字段
但是TLB只保存部分虚拟页号的映射信息

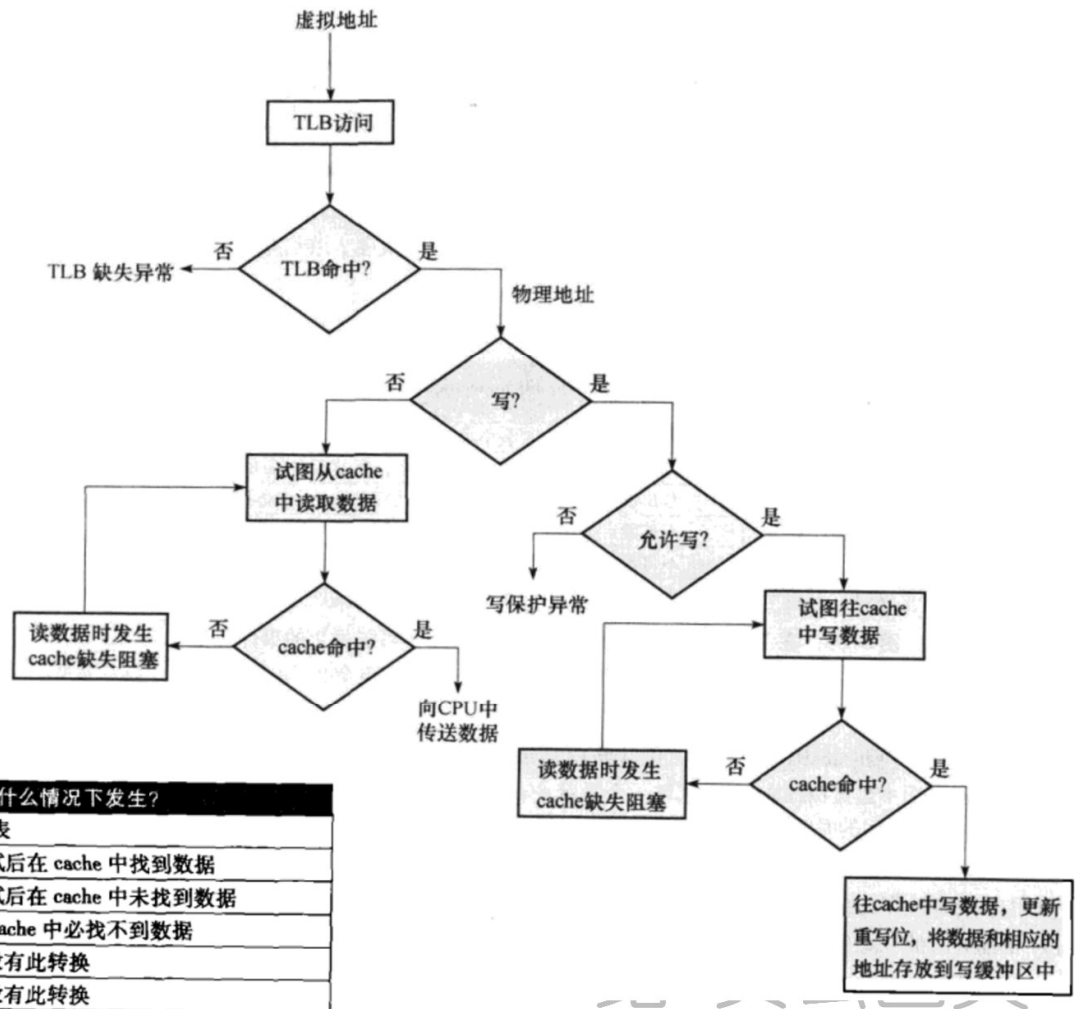
因此，TLB中需要将虚拟页号作为标记位
并保留页表中的全部信息
即3个标志位和物理页号



完整访存过程

在存储器层次结构中
高级存储器中的内容一定是
低级存储器的子集
尽管在写直达过程中和等待写回期间
同一时刻的具体数据有差异

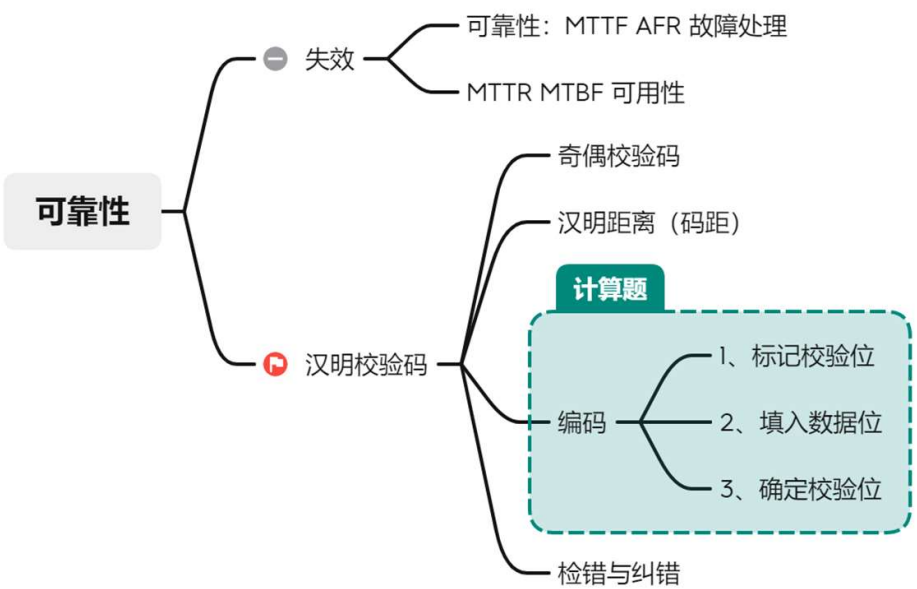
因此，高级存储器命中
低级存储器反而缺失的状况
不会发生



TLB	页表	cache	可能发生么？如果可能，在什么情况下发生？
命中	命中	缺失	可能，但若 TLB 命中就不可能检查页表
缺失	命中	命中	TLB 缺失，但在页表中找到表项；重试后在 cache 中找到数据
缺失	命中	缺失	TLB 缺失，但在页表中找到表项；重试后在 cache 中未找到数据
缺失	缺失	缺失	TLB 缺失，随后发生缺页；重试后在 cache 中必找不到数据
命中	缺失	缺失	不可能：如果页不在主存中，TLB 中没有此转换
命中	缺失	命中	不可能：如果页不在主存中，TLB 中没有此转换
缺失	缺失	命中	不可能：如果页不在主存中，数据不允许在 cache 中存在

第四部分

可靠性



可靠性与可用性评价

可靠性评价一个系统持续为用户提供服务的能力

从开始使用到失效的时间间隔称为平均无故障时间（mean time to failure, MTTF）

应对故障从而提高MTTF的策略有：①故障避免；②故障容忍；③故障预报

一年的时长比上MTTF得到年失效率（annual failure rate）

AFR = 一年 / MTTF

恢复一个系统功能的时间为平均维修时间（mean time to repair, MTTR）

从开始使用到失效维修结束为平均失效间隔时间（mean time between failure, MTBF）

可用性定义为系统正常工作的时间与两次服务中断间隔时间之比

从定义容易得出

MTBF = MTTF + MTTR
可用性 = MTTF / MTBF

用一年中可用性“9的个数”来表示：一个9表示可用性为90%，三个9表示99.9%

奇偶校验码 汉明距离（码距）

假设正确的8位数据为00010000
可以增加一个奇偶校验位，形成9位的奇偶校验码
在奇校验中，校验位确保整个校验码有奇数个1 偶校验如何定义？ 主要讨论偶校验

一个9位偶校验码为110001001（划线表示校验位）
如果传输中最高位数据出错，变为010001001，此时偶校验出错，请求重新传输
如果传输中最高两位同时出错，变为000001001，此时偶校验通过，不能检测出错
奇偶校验只能检测多少位错？

对于一个合法的奇偶校验码，如果其中一位改变，必然导致出错
两位同时改变，则得到另一个合法的校验码
任意两个合法校验码之间至少相差的位数称为汉明距离（码距）
奇偶校验码距为2，能实现检测一位错（single error detect, SED）
但不能定位哪一位出错

汉明校验码：编码

对8位数据10011010编入偶校验的汉明校验码

①从左往右第1、2、4、8……位“挖坑”，稍后填入校验位

②其他位依次填入数据位

③对校验位p1，从0开始数，0不检测/1要检测，2不检测/3要检测……

对校验位p2，从0开始数，0、1不检测/2、3要检测，4、5不检测/6、7要检测……

对校验位p4，从0开始数，0、1、2、3不检测/4、5、6、7要检测……

依次进行奇偶校验（通常是偶校验）

第③步中，p1、p2、p4检测的位数有什么规律？每个数据位至少被几个校验位覆盖？

位号	1	2	3	4	5	6	7	8	9	10	11	12
分类位号	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
p1覆盖												
p2覆盖												
p4覆盖												
p8覆盖												

汉明校验码：检错与纠错

同样对于8位数据，汉明校验码的校验位有4位，远远多于奇偶校验的1位
现在，任意两个合法汉明校验码之间相差的位数至少为3，即码距为3
能够定位一位错，对该位取反即可实现纠正一位错（single error correction, SEC）

对汉明校验码011100101010，如果从左往右第十位取反011100101110
如何实现纠错？
①根据新的码字，重新进行汉明校验编码
②将新旧两组校验位从右往左写，按位异或（相异为1、相同为0）
③结果即为发生错误的位号

通过对整个汉明校验码再进行一次奇偶校验，码距扩大为4
当最高位的汉明校验码（p4）和新的奇偶校验位H均能通过偶校验时，表示没有出错
p4和H产生四种奇偶组合，能够实现纠正1位错/检测两位错（SEC/DED）

虚拟机 cache控制器 分块加速 ARM和x86存储器层次结构

虚拟机（virtual machine）能够实现多个客户端在一个硬件主机上运行
由虚拟机监视器（VMM）实现不同操作系统间的资源隔离与控制
随着处理器性能的不断增长，虚拟机在PC中得到了广泛使用

通过构建一个4态有限状态机，可以实现简单的cache控制器

通过使用软件方法对大型矩阵进行分块
让处理器每次运算一个小矩阵块，而不是二位数组中的一行或一列
可以显著降低cache缺失率，利用存储器层次结构，将矩阵乘法性能再次翻倍

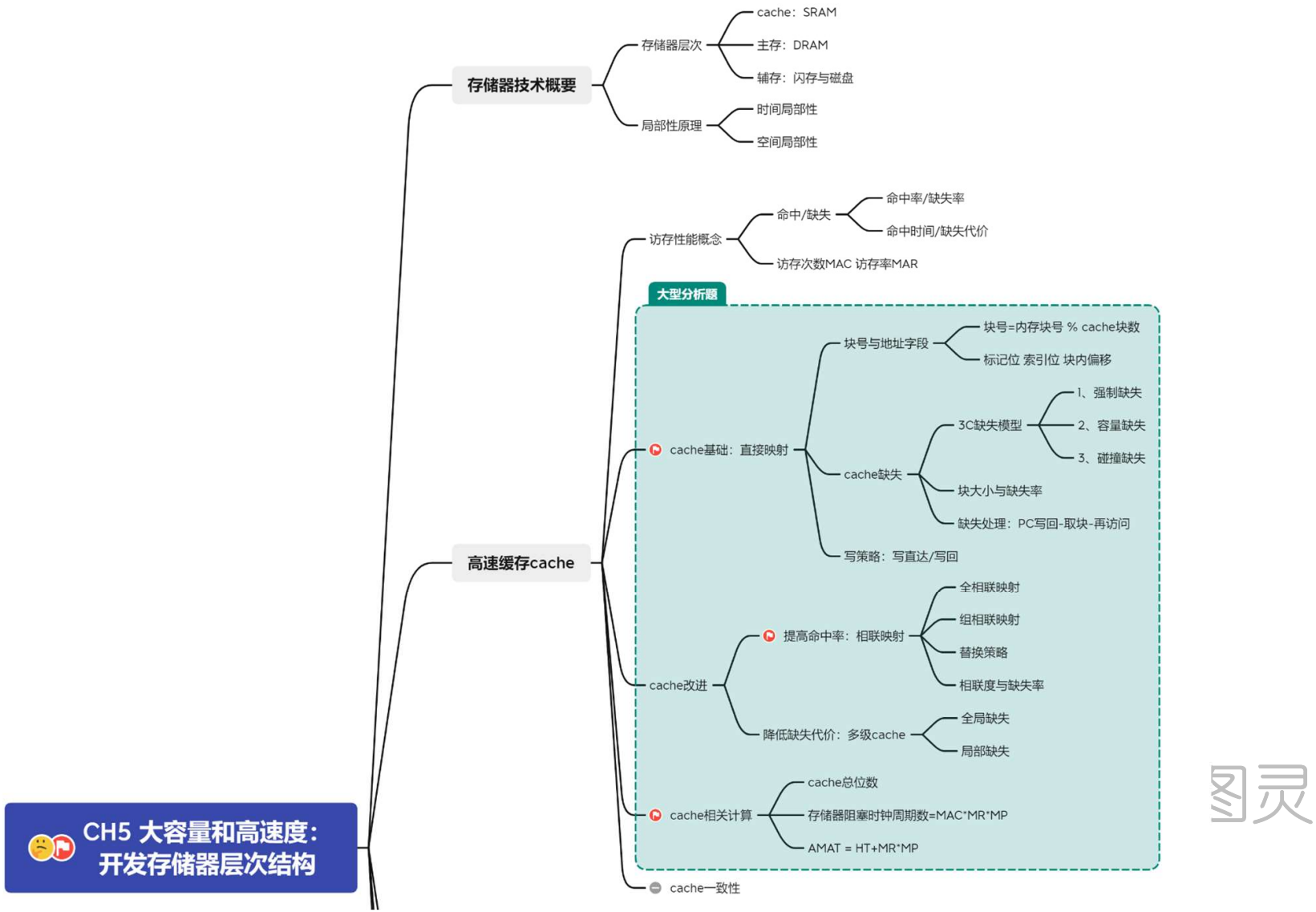
ARM Cortex-A8采用两级cache，Intel Core i7 Nehalem采用三级cache
其中L1 cache均为指令与数据分离的cache，A8的L2 cache使用统一cache
Core i7中的L2 cache每个核心统一，L3 cache由多核共享

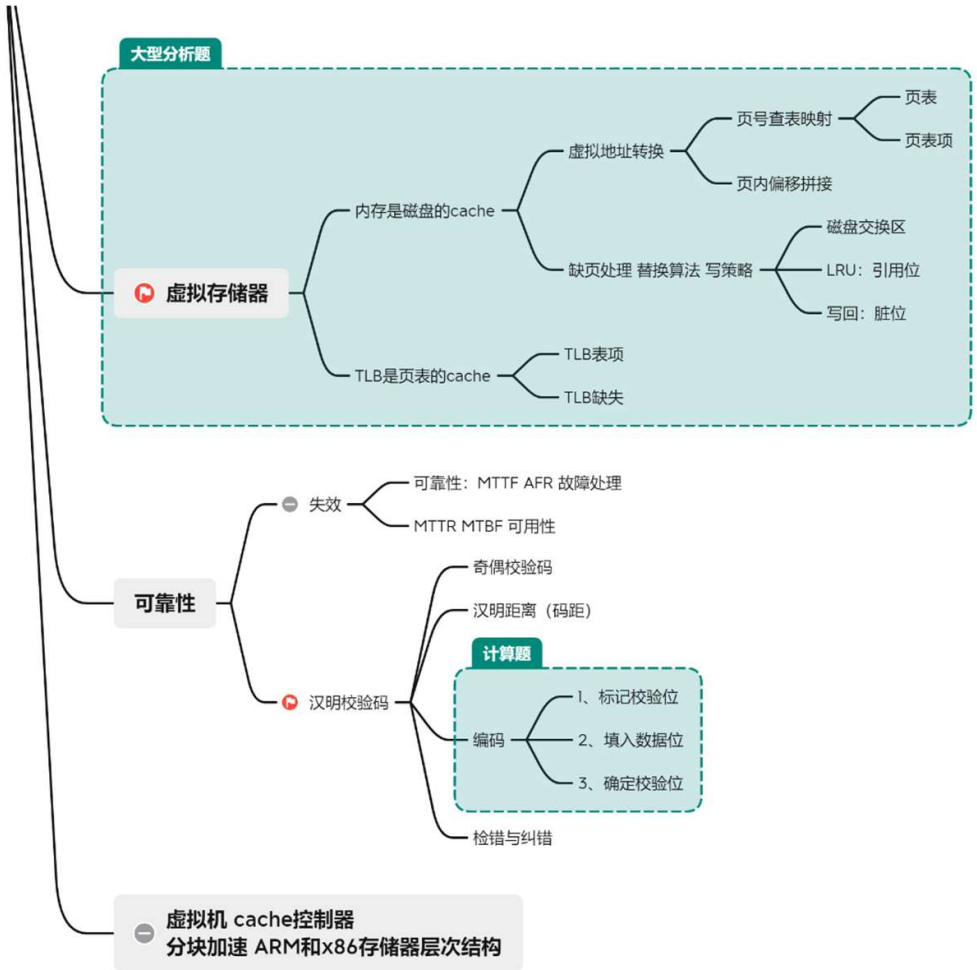
复习题

1、虚拟地址如何转化为物理地址?
2、页表项和TLB表项分别由哪些字段组成?为什么TLB多出了虚拟页号字段?
3、虚拟存储器缺页时，从哪里读取需要的页?
4、页表中的脏位和引用位分别有什么功能?为什么虚拟存储器只能采用写回机制?
5、TLB缺失、页表命中、cache缺失的情况是否有可能出现?
6、AFR和可用性如何定义?
7、数据0110如何进行偶校验汉明编码?
8、码距为2、3、4的校验码分别能实现什么功能?

全章复习

CH5 大容量与高速度：开发存储器层次结构


CH5 大容量和高速度：开发存储器层次结构
存储器技术概要
存储器层次
cache：SRAM
主存：DRAM
辅存：闪存与磁盘
局部性原理
时间局部性
空间局部性
高速缓存cache
访存性能概念
命中/缺失
命中率/缺失率
命中时间/缺失代价
访存次数MAC 访存率MAR
大型分析题
cache基础：直接映射
块号与地址字段
块号=内存块号 % cache块数
标记位 索引位 块内偏移
cache缺失
3C缺失模型
1、强制缺失
2、容量缺失
3、碰撞缺失
块大小与缺失率
缺失处理：PC写回-取块-再访问
写策略：写直达/写回
cache改进
提高命中率：相联映射
全相联映射
组相联映射
替换策略
相联度与缺失率
降低缺失代价：多级cache
全局缺失
局部缺失
cache相关计算
cache总位数
存储器阻塞时钟周期数=MAC*MR*MP
AMAT = HT+MR*MP
cache一致性


大型分析题
虚拟存储器
内存是磁盘的cache
虚拟地址转换
页号查表映射
页表
页表项
页内偏移拼接
缺页处理 替换算法 写策略
磁盘交换区
LRU：引用位
写回：脏位
TLB是页表的cache
TLB表项
TLB缺失
可靠性
失效
可靠性：MTTF AFR 故障处理
MTTR MTBF 可用性
汉明校验码
奇偶校验码
汉明距离（码距）
计算题
编码
1、标记校验位
2、填入数据位
3、确定校验位
检错与纠错
虚拟机 cache控制器
分块加速 ARM和x86存储器层次结构