计算机组织与体系结构

第二十讲

计算机科学与技术学院

张展

第八章　存储层次

本章内容

8.1 存储器的层次结构

8.2 Cache基本知识

8.3 降低Cache失效率的方法

8.4 减少Cache失效开销

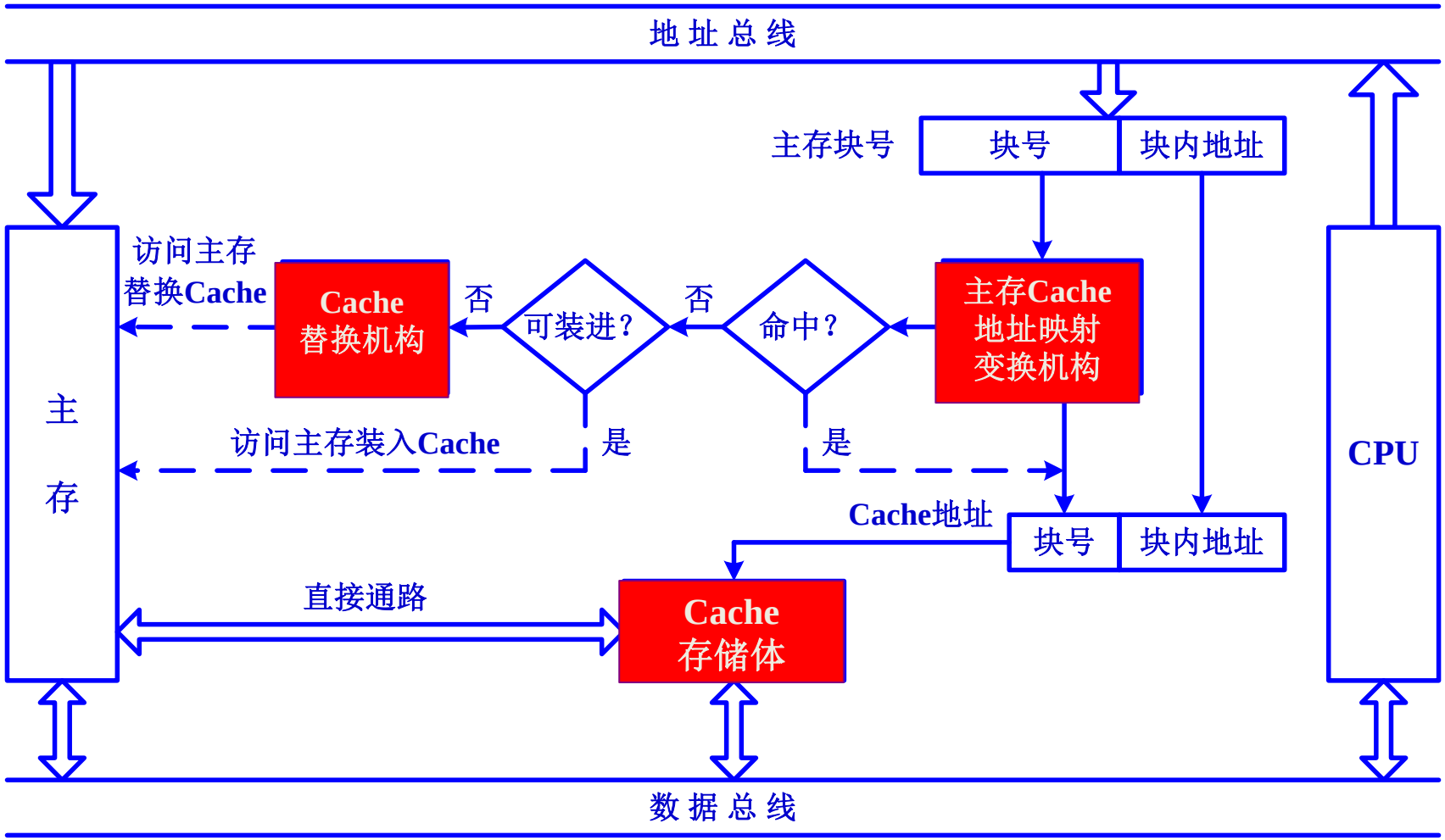
8.5 减少命中时间

8.6 主存

8.7 虚拟存储器

8.2 Cache基本知识

Cache 的基本结构



存储层次的四个问题

1. 当把一个块调入高一层(靠近CPU)存储器时，可以放在哪些位置上?
   (映象规则　调入块可以放在哪些位置)
2. 当所要访问的块在高一层存储器中时，如何找到该块?
   (查找算法　如何在映象规则 规定的候选位置查找)
3. 当发生失效时，应替换哪一块?
   (替换算法　规定的候选位置均被别的块占用)
4. 当进行写访问时，应进行哪些操作?
   (写策略　如何处理写操作)

8.2 Cache基本知识

8.2.1 映象规则

1．全相联映象

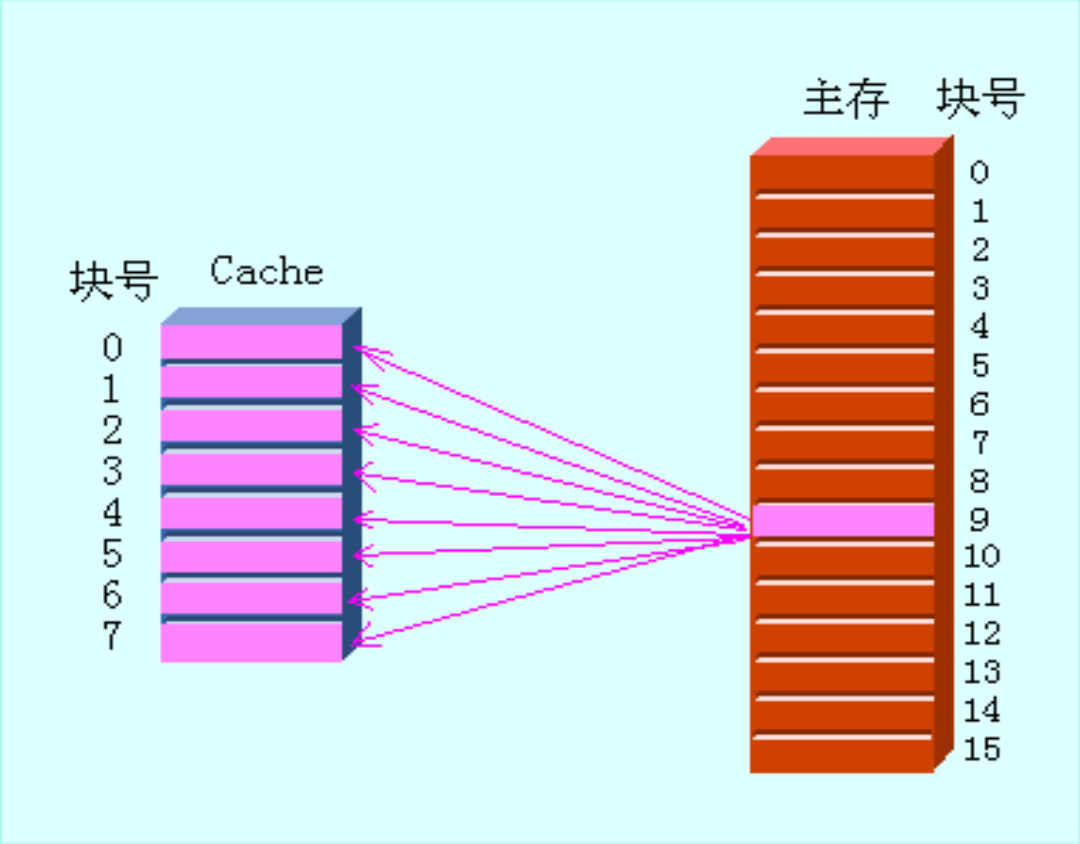
全相联：主存中的任一块可以被放置到Cache中的任意一个位置。

对比：阅览室位置 —— 随便坐

特点：空间利用率最高，冲突概率最低，实现最复杂。

全相联映射

（举例）



2. 直接映象

直接映象： 主存中的每一块只能被放置到Cache中唯一的一个位置。
(循环分配)

对比： 阅览室位置 —— 只有一个位置可以坐

特点： 空间利用率最低，冲突概率最高，实现最简单。

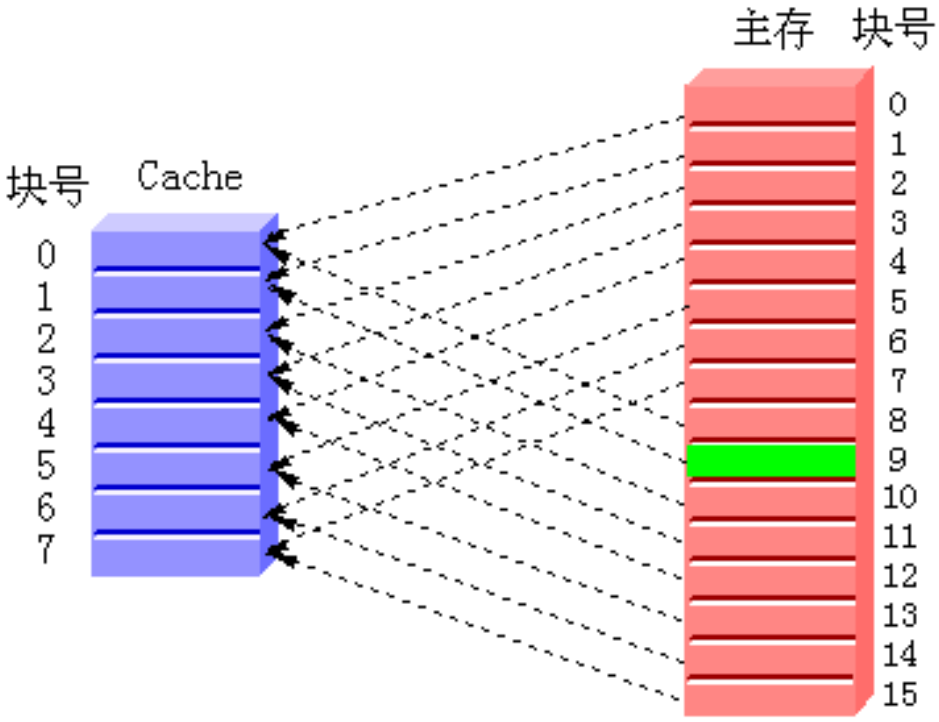
对于主存的第i块，若它映象到Cache的第j块，则:

$$j = i \bmod (M) \quad （M为Cache的块数）$$

设$M = 2^m$，则当表示为二进制数时，j实际上就是i的低m位:

直 接 映 射

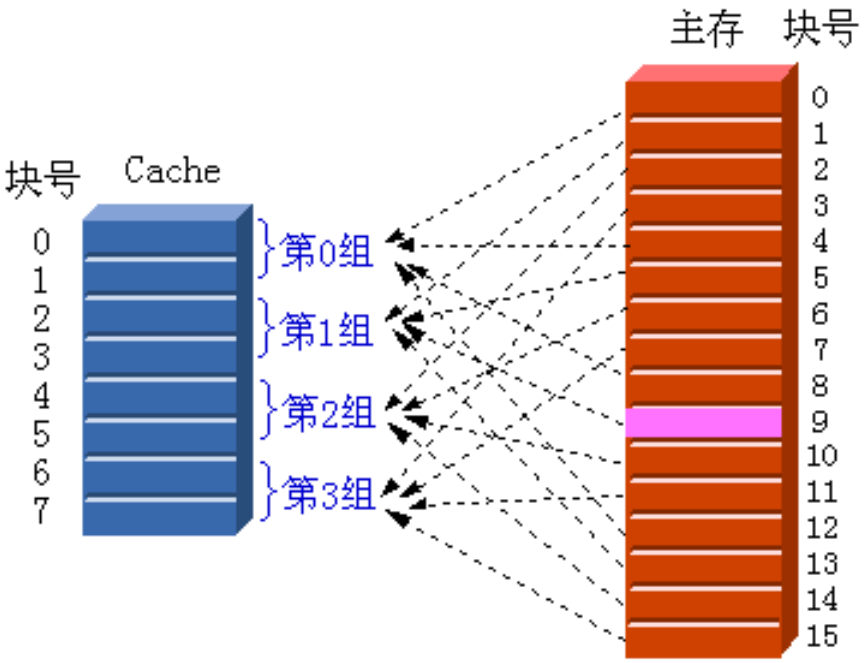
(举 例)



3. 组相联映象

- **组相联**：主存中的每一块可以被放置到Cache中唯一的一个组中的任何一个位置。

组相联是直接映象和全相联的一种折中



- 组的选择常采用位选择算法

  若主存第 i 块映象到第 k 组，则:

  $k = i \bmod (G)$　（G为Cache的组数）

  设$G = 2^g$，则当表示为二进制数时，k 实际上就是 i 的低 g 位:

- *n* 路组相联：每组中有*n* 个块($n=M/G$)，*n* 称为相联度

  相联度越高，Cache空间的利用率就越高，块冲突概率就越低，失效率也就越低。

	n (路数)	*G* (组数)
全相联	*M*	1
直接映象	1	*M*
组相联	$1<n<M$	$1<G<M$

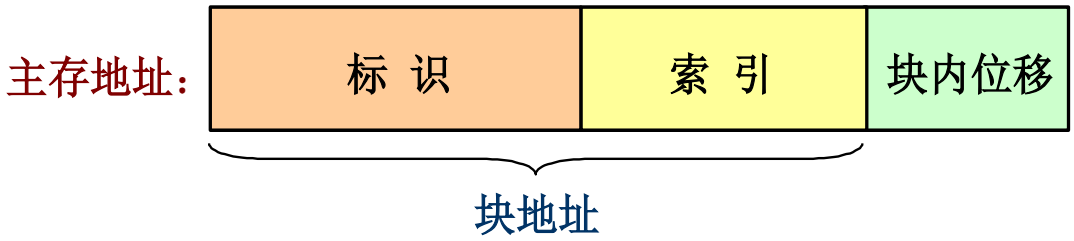
- 绝大多数计算机的Cache: $n \leq 4$

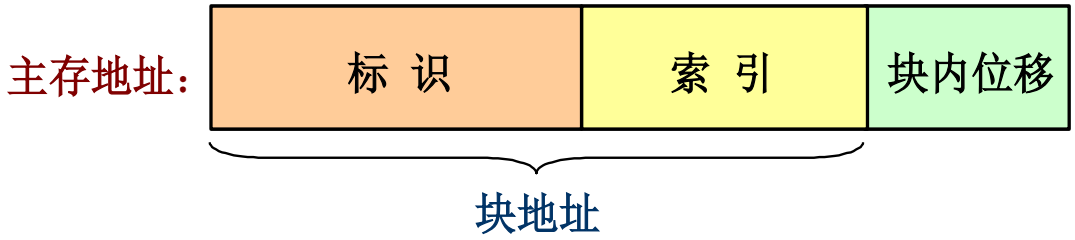
想一想：相联度一定是越大越好?

8.2.2 查找方法

- 当CPU访问Cache时，如何确定Cache中是否有所要访问的块?
- 若有的话，如何确定其位置?

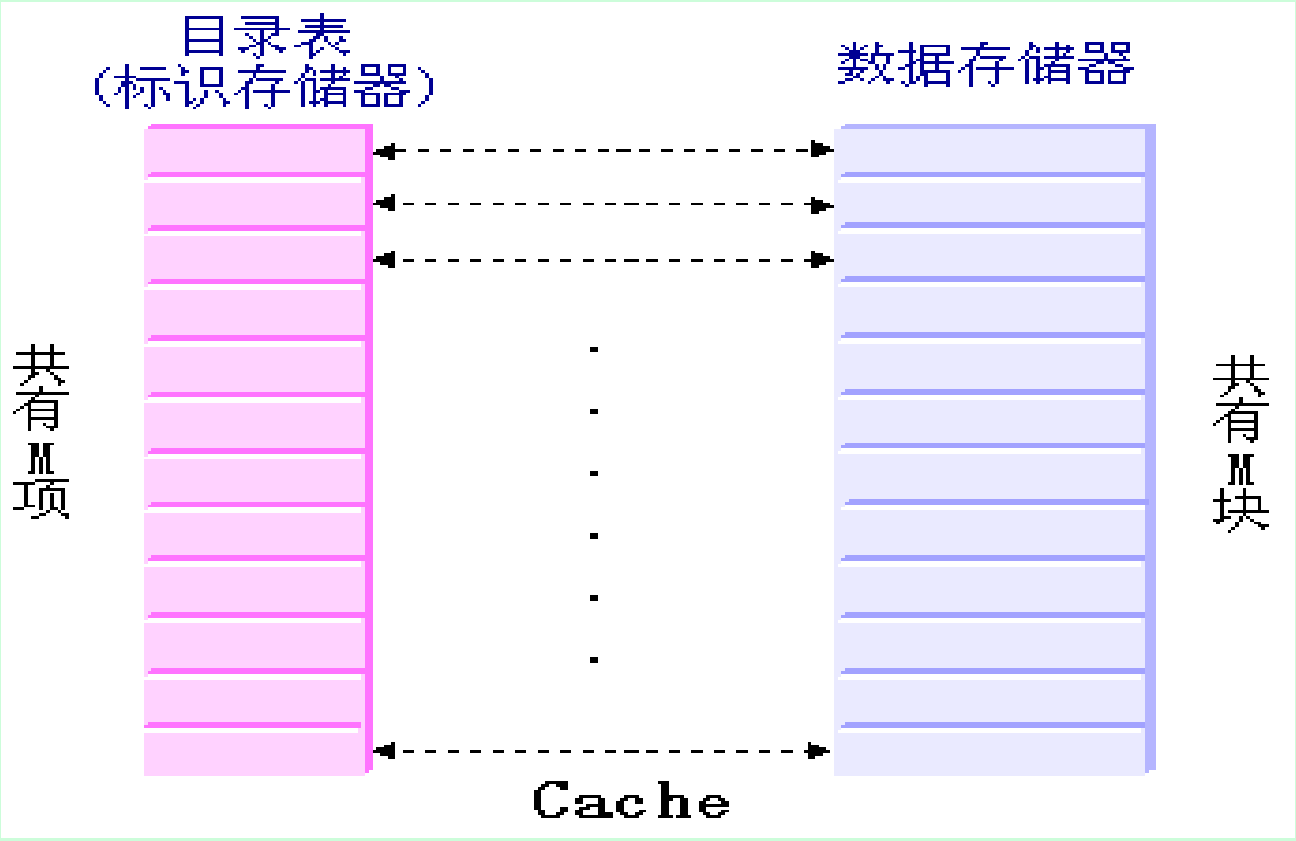
- 通过查找目录表来实现
  - 目录表的结构
    - 主存块的块地址的高位部分，称为标识 。
    - 每个主存块能唯一地由其标识来确定
    - 每一项有一个有效位，指出Cache中的块是否有效
  - 只需查找候选位置所对应的目录表项



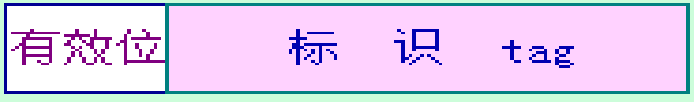


- **块内地址偏移用来从块中选出需要的数据**
- **索引域用来选择组**
- **通过比较标识域来判断是否发生命中**
  - **块内偏移是没有必要比较的，这是因为Cache命中与否的单位是整个块**
  - **由于索引域是用来选择被检查的组， 所以对索引域的比较是多余的**

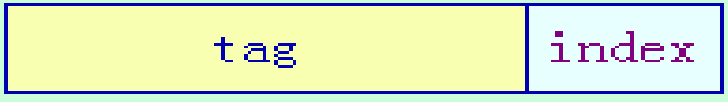
Cache目录表的结构



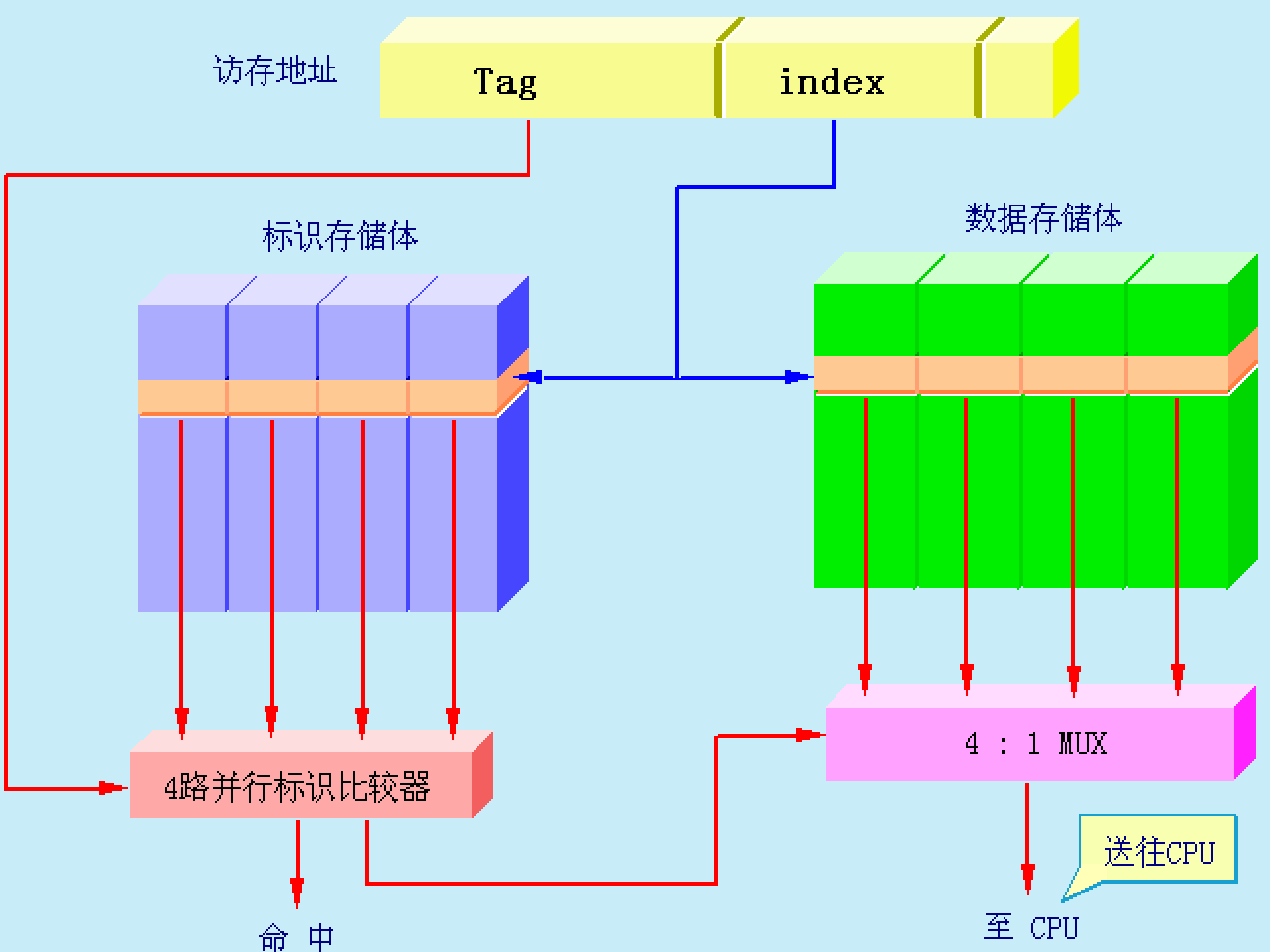
目录表项：

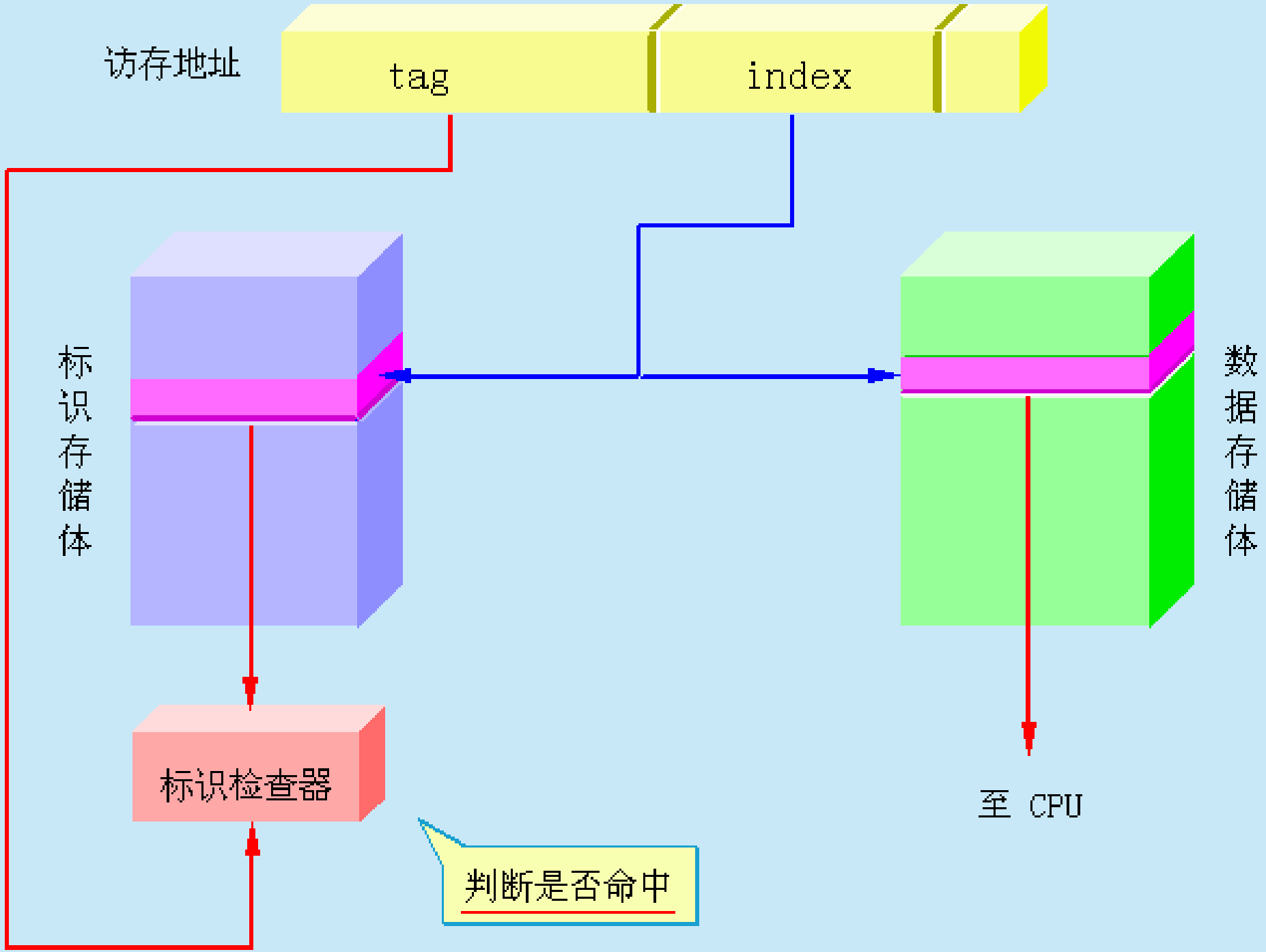


访存地址：



- 并行查找的实现方法：
  - ▲ 相联存储器
  - ▲ 单体多字存储器＋比较器
- 4 路组相联Cache的查找过程
  - 优缺点
    - 不必采用相联存储器，而是用按地址访问的存储器来实现。
    - 当相联度n增加时，不仅比较器的个数会增加，而且比较器的位数也会增加。
- 直接映象Cache的查找过程


访存地址
Tag
index
标识存储体
数据存储体
4路并行标识比较器
4 : 1 MUX
送往CPU
命 中
至 CPU


访存地址
tag
index
标识存储体
数据存储体
标识检查器
至 CPU
判断是否命中

8.2.3 替换算法

所要解决的问题：当新调入一块，而该块能够占用的Cache位置已被占满时，替换哪一块?

- 直接映象Cache中的替换很简单

  因为只有一个块，别无选择。
- 在组相联和全相联Cache中，则有多个块供选择。
  - 主要的替换算法有三种
    - 随机法

      优点：实现简单
    - 先进先出法FIFO
    - 最近最少使用法LRU

8.2.3 替换算法

- 最近最少使用法LRU
  - 选择近期最少被访问的块作为被替换的块。（实现比较困难）
  - 实际上：选择最久没有被访问过的块作为被替换的块。
  - 优点：命中率较高
- LRU和随机法分别因其不命中率低和实现简单而被广泛采用。
  - 模拟数据表明，对于容量很大的Cache，LRU和随机法的命中率差别不大。

8.2.4 写策略

1. “写”操作所占的比例

Load指令：26%　　Store指令：9%

“写”在所有访存操作中所占的比例：

9%/(100%＋26%＋9%)≈7%

“写”在访问数据Cache操作中所占的比例：

9%/(26%＋9%)≈25%

2. “写”操作必须在确认是否命中后才可进行

3. “写”访问有可能导致Cache和主存内容的不一致

4. 两种写策略

- **写直达法**：执行“写”操作时，不仅写入Cache，而且也写入下一级存储器。
- **写回法**：执行“写”操作时，只写入Cache。仅当Cache中相应的块被替换时，才写回主存。

(设置“脏位”)

5. 两种写策略的比较

- **写回法的优点**：速度快，占用存储器频带低
- **写直达法的优点**：易于实现，一致性好

6. 写缓冲器

7. “写”操作时的调块

- 按写分配(写时取)：写失效时，先把所写单元所在的块调入Cache，再行写入。
- 不按写分配(绕写法)：写失效时，直接写入下一级存储器而不调块。

8. 写策略与调块

写回法 —— 按写分配

写直达法 —— 不按写分配

8.2.5 Cache结构

例子：DEC的Alpha AXP21064中的内部数据Cache

1. 简介

容量：8KB

块大小：32B

块数：256

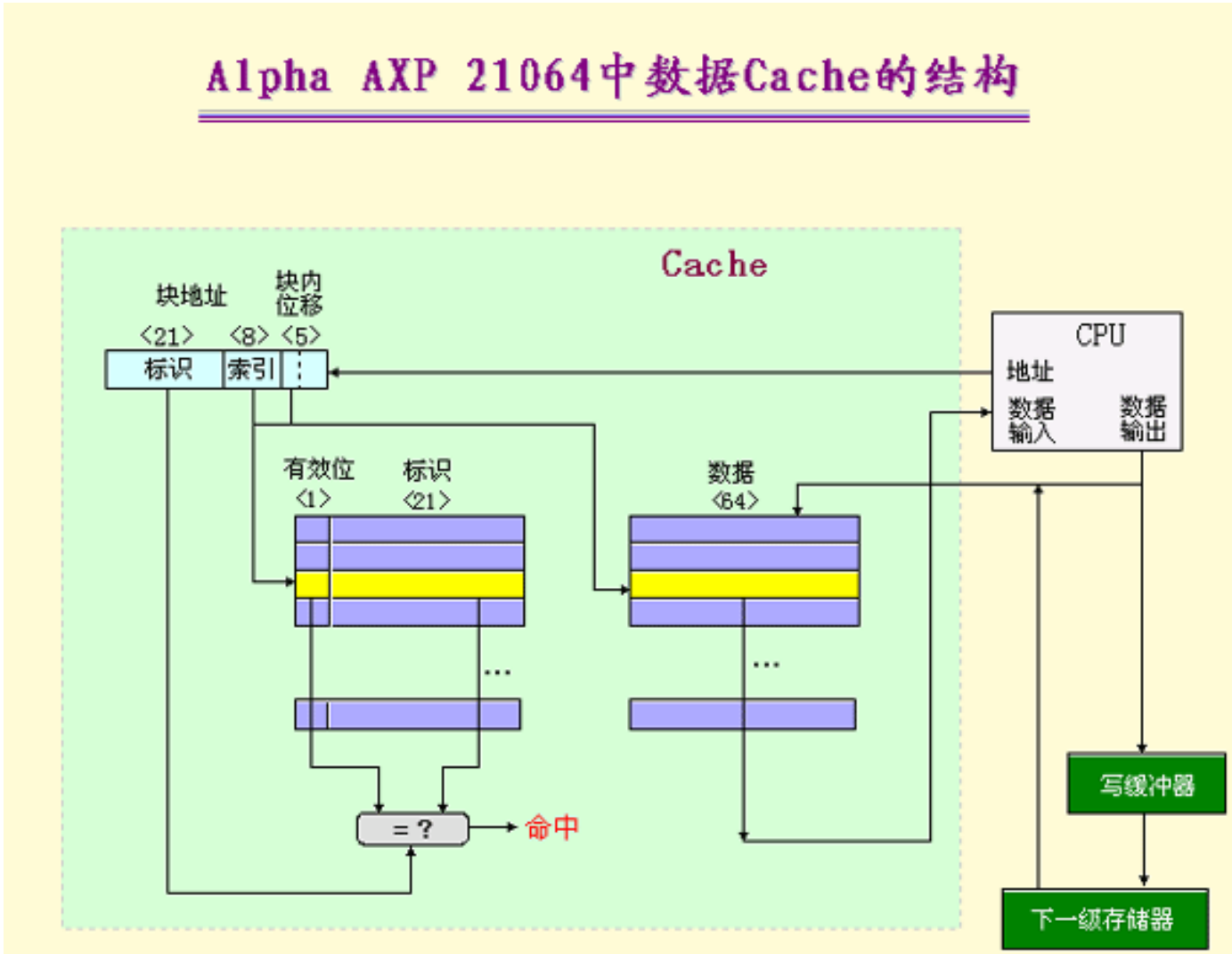
映象方法：直接映象

“写”策略：写直达—不按写分配

写缓冲器大小：4个块

内存地址：34位（块地址29位，块内地址5位）

结构图


Alpha AXP 21064中数据Cache的结构
Cache
块地址
块内位移
〈21〉
〈8〉
〈5〉
标识
索引
有效位
〈1〉
标识
〈21〉
数据
〈64〉
CPU
地址
数据输入
数据输出
写缓冲器
下一级存储器
= ?
命中

3. 工作过程

① 处理器传送给Cache物理地址

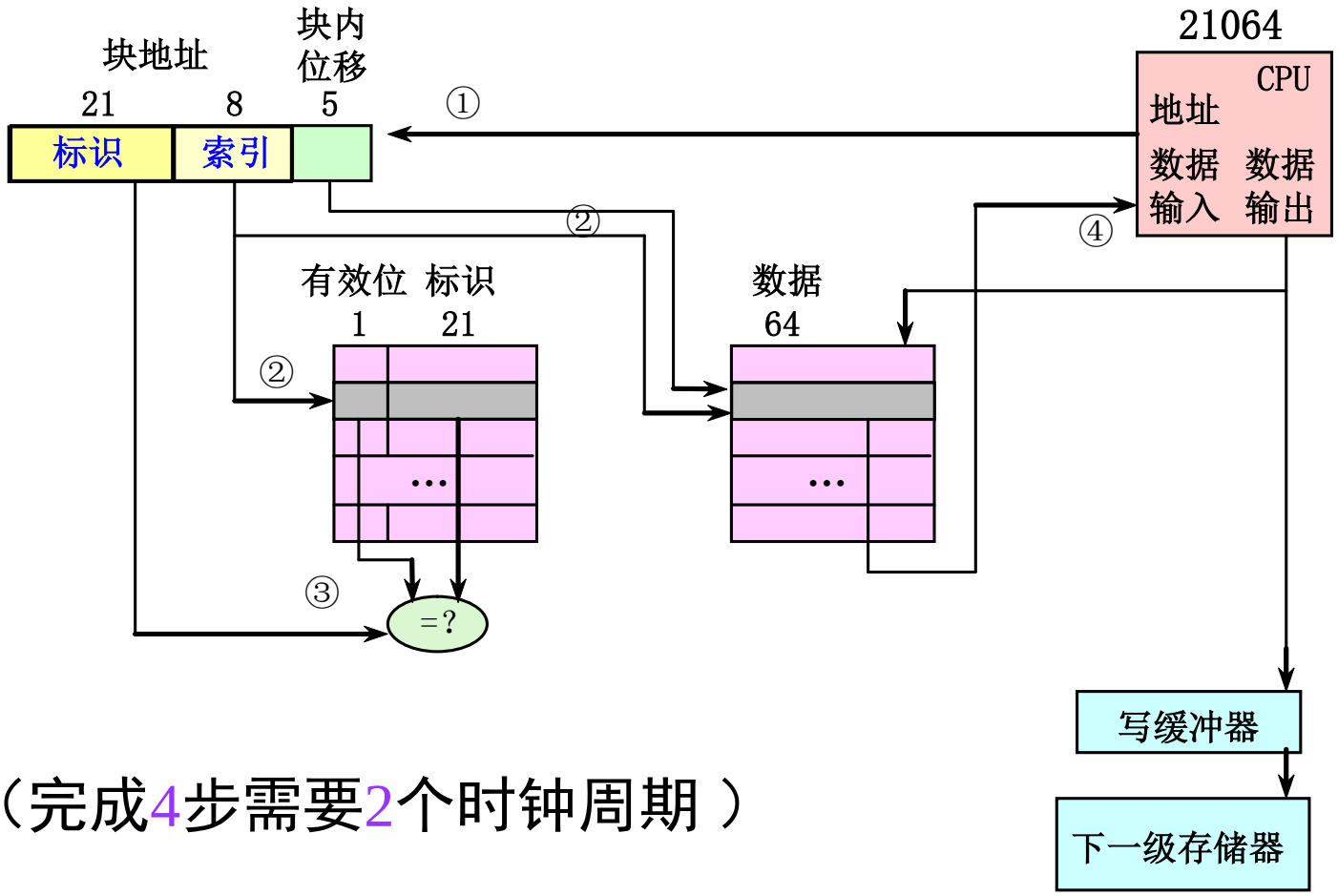
- Cache的容量与索引index、相联度、块大小之间的关系

Cache的容量=2^{index}×相联度×块大小

把容量为8192、相联度为1、块大小为32（字节）代入：

索引index：8位　　标识：29－8＝21位

3. 工作过程



（完成4步需要2个时钟周期）

3. 工作过程

① 处理器传送给Cache物理地址

② 由索引选择标识的过程

- 根据索引从目录项中读出相应的标识和有效位

③ 从Cache中读出标识之后，用来同从CPU发来的块地址中标志域部分进行比较

- 为了保证包含有效的信息，必须要设置有效位

④ 如果有一个标识匹配，且标志位有效，则此次命中

- 通知CPU取走数据

- “写”访问命中
  - 前三步一样，只有在确认标识匹配后才把数据写入
- 设置了一个写缓冲器

  （提高“写”访问的速度）
  - 按字寻址的，它含有4个块，每块大小为4个字。
  - 当要进行写入操作时，如果写缓冲器不满，那么就把数据和完整的地址写入缓冲器。对CPU而言，本次“写”访问已完成，CPU可以继续往下执行。由写缓冲器负责把该数据写入主存。
  - 在写入缓冲器时，要进行写合并检查。即检查本次写入数据的地址是否与缓冲器内某个有效块的地址匹配。如果匹配，就把新数据与该块合并 。

– 发生读不命中与写不命中时的操作

- 读不命中：向CPU发出一个暂停信号，通知它等待，并从下一级存储器中新调入一个数据块（32字节）
  - Cache与下一级存储的数据通路宽度为16B，传送一次需5个周期，因此，一次传送需要10个周期
- 写不命中：将使数据“绕过”Cache，直接写入主存。
  - 写直达 –不按写分配
- 因为是写直达，所以替换时不需要写回

8.2.6 性能分析

- **不命中率（缺失率）**
  - 与硬件速度无关
  - 容易产生一些误导
  - 平均访存时间比不命中率更好
- **平均访存时间**

平均访存时间 = 命中时间＋不命中率×不命中开销

- 可以采用绝对时间——如一次命中需0.25-1.0纳秒
- 或是用CPU等待存储器的时钟周期数——如缺失代价用75-100个时钟周期表示
- 仍旧不能代替执行时间

混合Cache与分离Cache

(1) 优缺点

(2) 分离Cache平均失效率的计算：

访问指令Cache的百分比×指令Cache的失效率
＋访问数据Cache的百分比×数据Cache的失效率

例8.1 分体Cache与一体Cache

- 一个16K指令Cache和16K数据Cache与一个32K的一体Cache相比较，哪一个具有更低的缺失率?
- 假设36％的指令是数据传输指令，缺失率如图所示。假定Cache命中需要1个时钟周期，缺失代价是100个时钟周期，在一体Cache 中，Load和Store命中额外需要一个时钟周期，因为只有一个Cache端口来满足这两个同时发生的请求。使用上一章提到流水线的术语，一体Cache导致结构冲突。每种情况下的平均存储器存取时间是多少呢？假定是带有写缓存的写直达Cache，而且写缓存的停顿时间可以忽略不记

如上所述，大约 74%的存储器访问是访问指令的。因此，分立 Cache 的总的缺失率为：

$$(74\% \times 0.004) + (26\% \times 0.114) = 0.0324$$

因此，32K 的一体 Cache 有相对稍低的缺失率。

平均存储器存取时间公式可以分为指令和数据的访问：

$$\text{平均存储器存取时间} = \text{指令所占几率} \times (\text{命中时间} + \text{指令缺失率} \times \text{缺失代价}) + \text{数据所占几率} \times (\text{命中时间} + \text{数据缺失率} \times \text{缺失代价})$$

所以，两种 Cache 结构的平均存储器存取时间如下：

$$\text{平均存储器存取时间}_{\text{分立cache}} = 74\% \times (1 + 0.004 \times 100) + 26\% \times (1 + 0.114 \times 100)$$
$$= (74\% \times 1.38) + (26\% \times 12.36) = 1.023 + 3.214 = 4.24$$

$$\text{平均存储器存取时间}_{\text{一体cache}} = 74\% \times (1 + 0.0318 \times 100) + 26\% \times (1 + 1 + 0.0318 \times 100)$$
$$= (74\% \times 4.18) + (26\% \times 5.18) = 3.096 + 1.348 = 4.44$$

因此，本例中分立Cache——在每个时钟周期提供两个存储器端口，因此可以避免结构冲突——比只有一个端口的一体Cache有更好的平均存储器存取时间，尽管它的有效缺失率更高一些。

8.2.6 性能分析

CPU时间

CPU时间＝(CPU执行周期数＋存储器停顿周期数)
×时钟周期时间

存储器停顿周期数＝访存次数×失效率×失效开销

存储器停顿时钟周期数＝“读”的次数×读不命中率×读不命中开销＋“写”的次数×写不命中率×写不命中开销

CPU时间＝IC×[CPIexe＋访存次数/指令数×
失效率×失效开销]×时钟周期时间

CPU时间＝IC×[CPIexe＋每条指令的平均存储器停顿周期数]×时钟周期时间

例8.2

我们用一个和Alpha AXP类似的机器作为第一个例子。当不考虑存储器停顿时，所有指令的执行时间都是2.0个时钟周期。假设Cache失效开销为50个时钟周期，Cache的失效率为2%，平均每条指令访存1.33次。试分析Cache对性能的影响。

解：

CPU时间＝IC×[CPIexe＋访存次数/指令数×
失效率×失效开销]×时钟周期时间

考虑Cache的失效后，性能为：

$CPU\ 时间_{有cache} = IC \times (2.0 + (1.33 \times 2\% \times 50))$
×时钟周期时间
$= IC \times 3.33 \times$时钟周期时间

实际*CPI*： 3.33 —— 3.33/2.0 = 1.67(倍)

CPU时间也增加为原来的1.67倍。但若不采用Cache,
则：$CPI = 2.0 + 50 \times 1.33 = 68.5$

例8.3 考虑两种不同组织结构的Cache：直接映象Cache和两路组相联Cache，试问它们对CPU的性能有何影响？先求平均访存时间，然后再计算CPU性能。分析时请用以下假设：

(1)理想Cache（命中率为100%）情况下的CPI为2.0，时钟周期为2ns，平均每条指令访存1.3次。

(2)两种Cache容量均为64KB，块大小都是32字节。

(3)在组相联Cache中，由于多路选择器的存在而使CPU的时钟周期增加到原来的1.10倍。这是因为对Cache的访问总是处于关键路径上，对CPU的时钟周期有直接的影响。

(4) 这两种结构Cache的不命中开销都是70ns。（在实际应用中，应取整为整数个时钟周期）

(5) 命中时间为1个时钟周期，64KB直接映象Cache的不命中率为1.4%，相同容量的两路组相联Cache的不命中率为1.0%。

解 平均访存时间为：

平均访存时间＝命中时间＋不命中率×不命中开销

因此，两种结构的平均访存时间分别是：

平均访存时间1路＝2.0＋（0.014×70）＝2.98ns

平均访存时间2路＝2.0×1.10＋（0.010×70）＝2.90ns

两路组相联Cache的平均访存时间比较低。

CPU时间＝IC×（CPIexecution＋每条指令的平均访存次数×不命中率×不命中开销）× 时钟周期时间

＝IC×（CPIexecution× 时钟周期时间＋每条指令的平均访存次数×不命中率×不命中开销×时钟周期时间）

因此：

CPU时间1路 ＝ IC×(2.0×2＋(1.3×0.014×70))

＝ 5.27×IC

CPU时间2路 ＝ IC×(2.0×2×1.10＋(1.3×0.010×70))

＝ 5.31×IC

$$\frac{\text{CPU时间}_{2\text{路}}}{\text{CPU时间}_{1\text{路}}} = \frac{5.31 \times IC}{5.27 \times IC} = 1.01$$

与平均存储器存取时间的比较结果相反，直接映象Cache的平均性能好一些。

2-路组相联情况下，尽管它的缺失率低一些，但是所有指令时钟周期都延长了。

谢谢！