计算机体系结构 期末考试说明

1、考试时间：11月10日晚

2、考试题型：单项选择题 、填空题、名词解释、简答题、分析计算题

3、考试范围：以课本为主，课本上没有讲的内容，不会考。

1、本周作业：

第8章

第2，3，5, 6 题。

第8章 多处理机

8.1 引 言

1. 单处理机系统结构正在走向尽头？

2. 多处理机正起着越来越重要的作用。近几年来，人们确实开始转向了多处理机。

- Intel于2004年宣布放弃了其高性能单处理器项目，转向多核（multi-core）的研究和开发。
- IBM、SUN、AMD等公司
- 并行计算机应用软件已有了稳定的发展。
- 充分利用商品化微处理器所具有的高性能价格比的优势。

3. 本章重点：中小规模的计算机(处理器的个数＜32)

（多处理机设计的主流）

8.1.1 并行计算机系统结构的分类

1. Flynn分类法

   SISD、SIMD、MISD、MIMD

2. MIMD已成为通用多处理机系统结构的选择，原因：

   - MIMD具有灵活性；□
   - MIMD可以充分利用商品化微处理器在性能价格比方面的优势。

   计算机机群系统（cluster）是一类广泛被采用的MIMD机器。

3. 根据存储器的组织结构，把现有的MIMD机器分为两类：

（每一类代表了一种存储器的结构和互连策略）

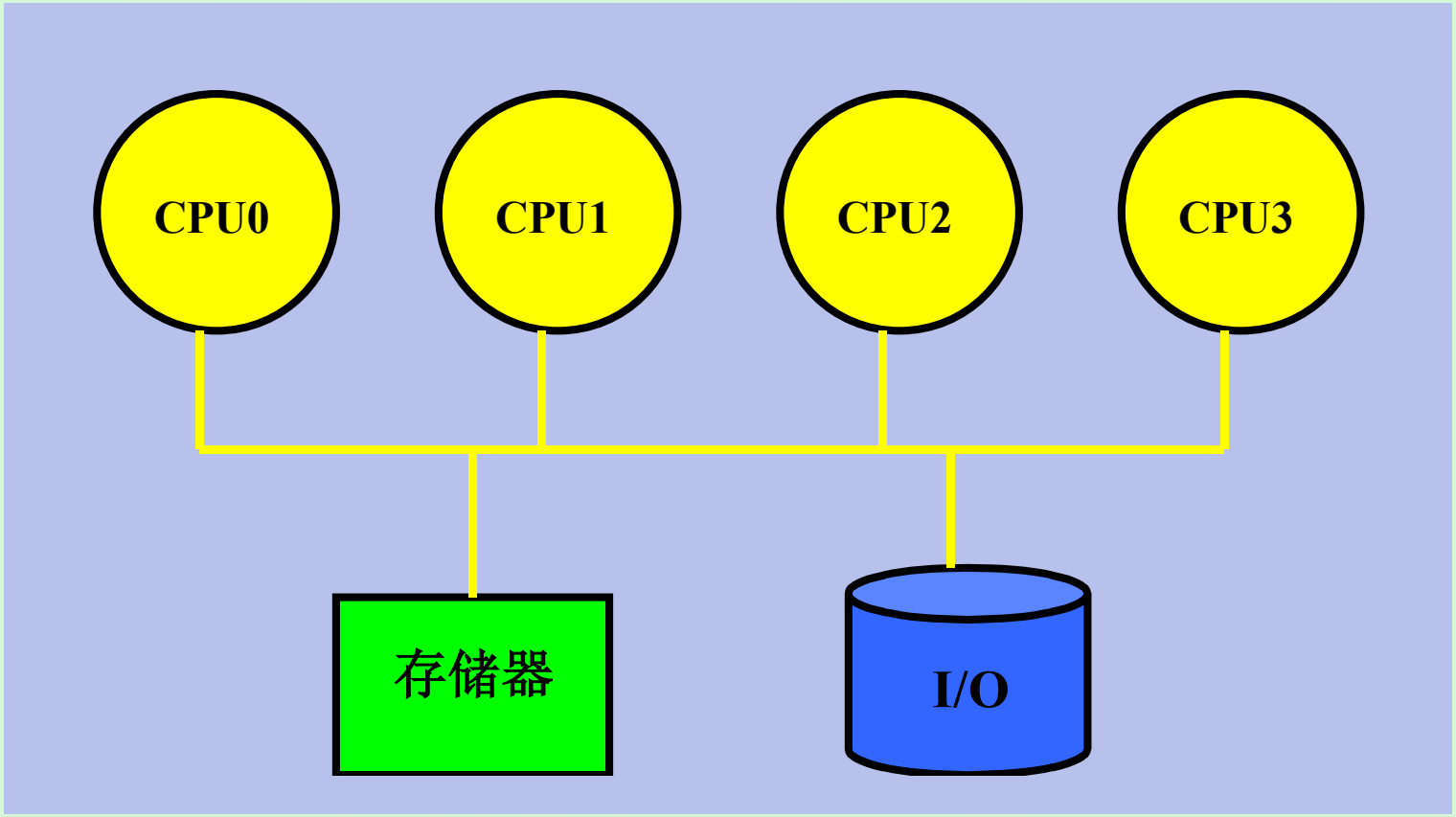
- （1）集中式共享存储器结构
- 最多由几十个处理器构成。
  - 各处理器共享一个集中式的物理存储器。

这类机器有时被称为

  - SMP机器

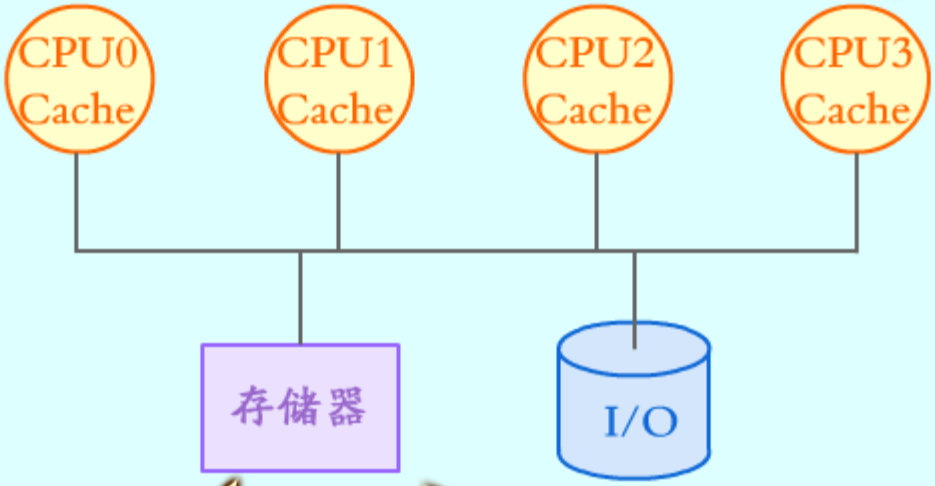
（Symmetric shared-memory MultiProcessor）

  - UMA机器（Uniform Memory Access）



对称式共享存储器多处理机的基本结构

这类多处理机在目前至多有几十个处理器



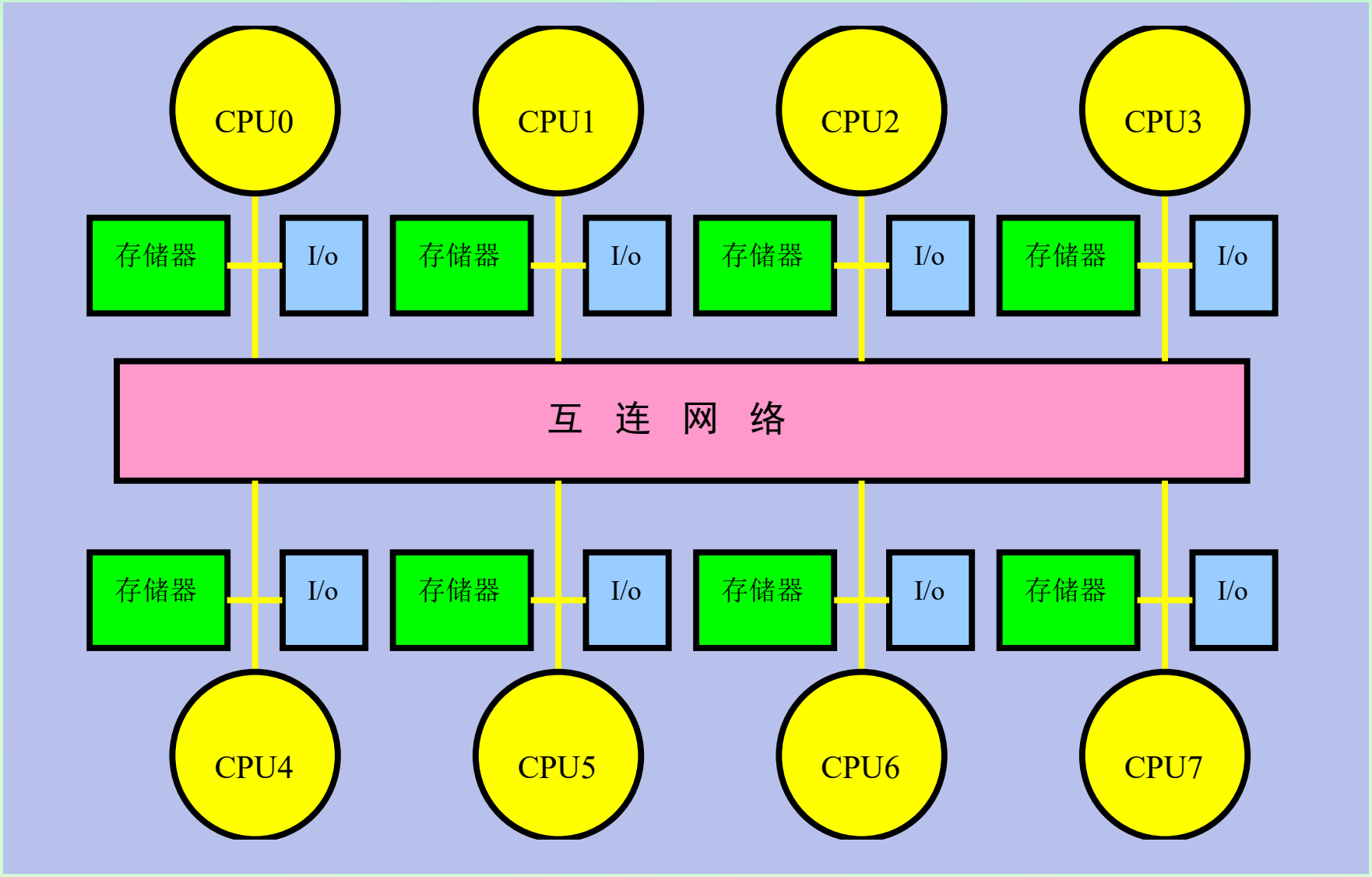
由于处理器数目较小，可通过大容量的Cache和总线互连使各个处理器共享一个单独的集中式存储器。

因为只有一个单独的主存，而且从各个处理器访问该存储器的时间是相同的，所以这类机器有时被称为UMA(Uniform Memory Access)机器。

- （2）分布式存储器多处理机

  存储器在物理上是分布的。

  - 每个结点包含：
    - 处理器
    - 存储器
    - I／O
    - 互连网络接口
  - 在许多情况下，分布式存储器结构优于集中式共享存储器结构。


CPU0
CPU1
CPU2
CPU3
存储器
I/o
存储器
I/o
存储器
I/o
存储器
I/o
互 连 网 络
存储器
I/o
存储器
I/o
存储器
I/o
存储器
I/o
CPU4
CPU5
CPU6
CPU7

8.1.3 并行处理面临的挑战

并行处理面临着两个重要的挑战

- 程序中的并行性有限
- 相对较大的通信开销

$$\text{系统加速比} = \frac{1}{(1-\text{可加速部分比例}) + \dfrac{\text{可加速部分比例}}{\text{理论加速比}}}$$

1. 第一个挑战

有限的并行性使计算机要达到很高的加速比十分困难。

例8.1 假设想用100个处理器达到80的加速比，求原计算程序中串行部分最多可占多大的比例？

解 Amdahl定律为：

$$\text{加速比}=\frac{1}{\frac{\text{可加速部分比例}}{\text{理论加速比}}+(1-\text{可加速部分比例})}$$

$$80=\frac{1}{\frac{\text{并行比例}}{100}+(1-\text{并行比例})}$$

由上式可得：并行比例＝0.9975

2. 第二个挑战：多处理机中远程访问的延迟较大

- 在现有的机器中，处理器之间的数据通信大约需要50～1000个时钟周期。
- 主要取决于：

  通信机制、互连网络的种类和机器的规模

- 在几种不同的共享存储器并行计算机中远程访问一个字的典型延迟

机器	通信机制	互连网络	处理机最大数量	典型远程存储器访问时间（ns）
Sun Starfire servers	SMP	多总线	64	500
SGI Origin 3000	NUMA	胖超立方体	512	500
Cray T3E	NUMA	3维环网	2048	300
HP V series	SMP	8×8交叉开关	32	1000
HP AlphaServer GS	SMP	开关总线	32	400

例8.2 假设有一台32台处理器的多处理机，对远程存储器访问时间为200ns。除了通信以外，假设所有其他访问均命中局部存储器。当发出一个远程请求时，本处理器挂起。处理器的时钟频率为2GHz，如果指令基本的CPI为0.5（设所有访存均命中Cache），求在没有远程访问的情况下和有0.2%的指令需要远程访问的情况下，前者比后者快多少?

解 有0.2%远程访问的机器的实际CPI为：

CPI＝基本CPI＋远程访问率×远程访问开销

＝0.5＋0.2%×远程访问开销

远程访问开销为：

远程访问时间/时钟周期时间＝200ns/0.5ns＝400个时钟周期

∴ CPI＝0.5＋0.2%×400＝1.3

因此在没有远程访问的情况下的机器速度是有0.2%远程访问的机器速度的1.3/0.5=2.6倍。

- 问题的解决
  - 并行性不足： 采用并行性更好的算法
  - 远程访问延迟的降低：靠系统结构支持和编程技术

3. 在并行处理中，影响性能（负载平衡、同步和存储器访问延迟等）的关键因素常依赖于：

应用程序的高层特性

如数据的分配，并行算法的结构以及在空间和时间上对数据的访问模式等。

- 依据应用特点可把多机工作负载大致分成两类：
  - 单个程序在多处理机上的并行工作负载
  - 多个程序在多处理机上的并行工作负载

8.2 对称式共享存储器系统结构

- 多个处理器共享一个存储器。
- 当处理机规模较小时，这种计算机十分经济。
- 近些年，能在一个单独的芯片上实现2～8个处理器核。

  例如：Sun公司 2006年 T1 8核的多处理器
- 支持对共享数据和私有数据的Cache缓存

  私有数据供一个单独的处理器使用，而共享数据则是供多个处理器使用。
- 共享数据进入Cache产生了一个新的问题

  Cache的一致性问题

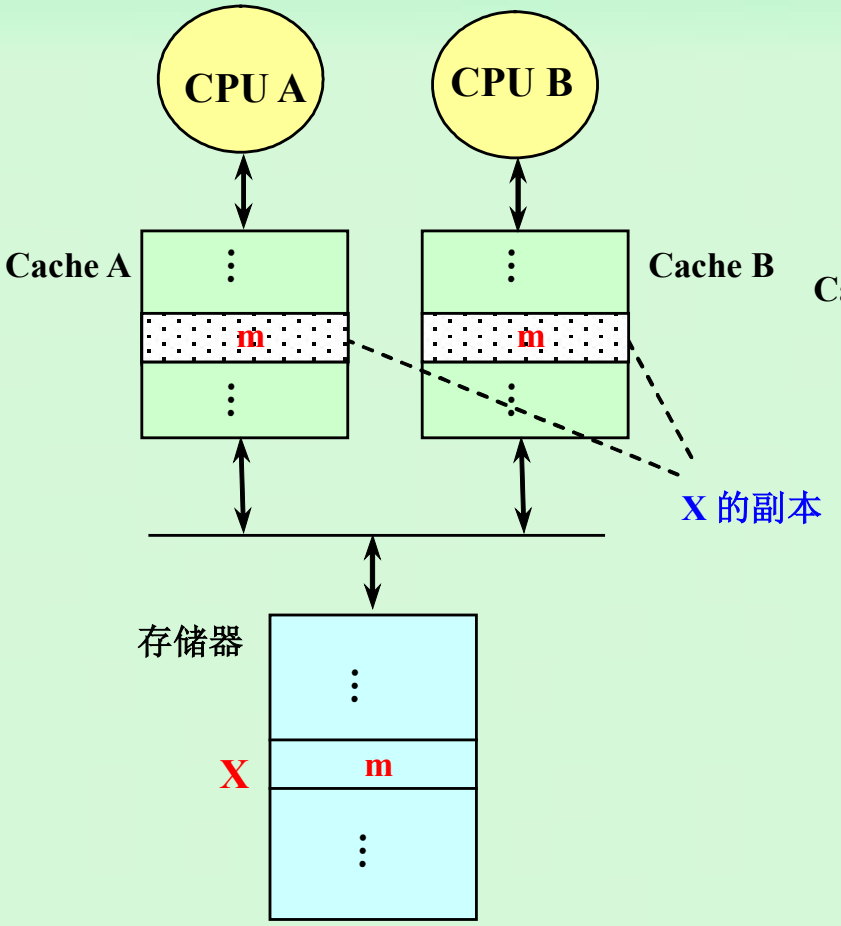
8.2.1 多处理机Cache一致性

1. 多处理机的Cache一致性问题

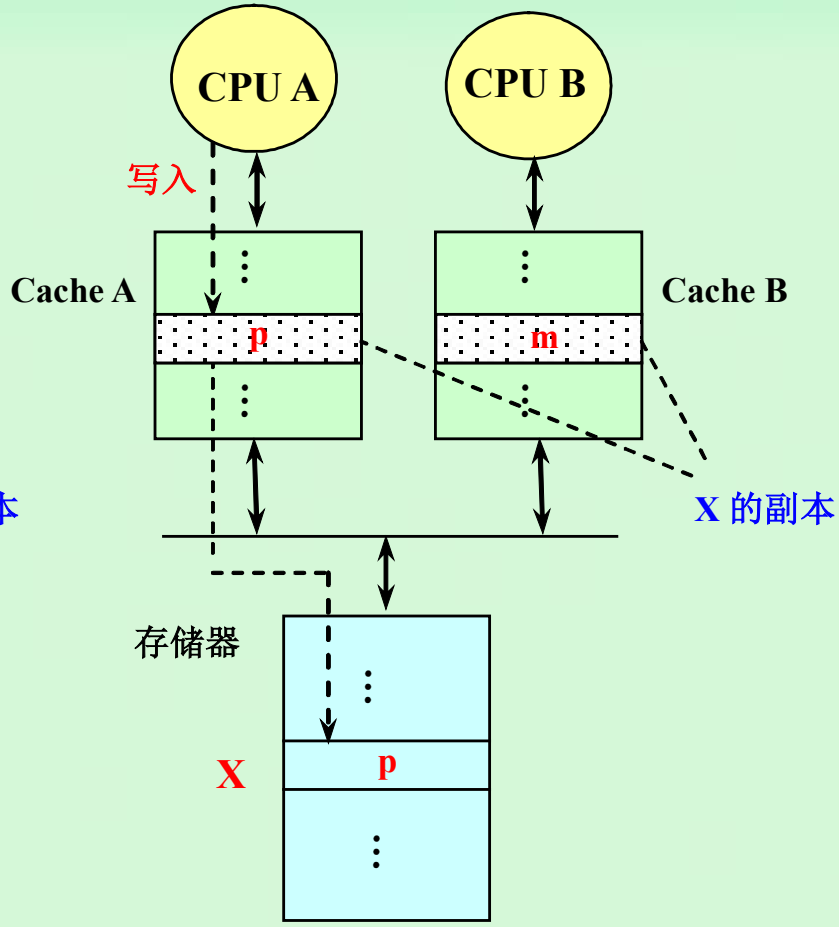
- 允许共享数据进入Cache，就可能出现多个处理器的Cache中都有同一存储块的副本，
- 当其中某个处理器对其Cache中的数据进行修改后，就会使得其Cache中的数据与其他Cache中的数据不一致。

例 由两个处理器（A和B）读写引起的Cache一致性问题

动画



（a） CPU A 写入前



（b） CPU A 将 p 写入 X，p≠m

2. 存储器的一致性

如果对某个数据项的任何读操作均可得到其最新写入的值，则认为这个存储系统是一致的。

- 存储系统行为的两个不同方面
  - What：读操作得到的是什么值
  - When：什么时候才能将已写入的值返回给读操作
- 需要满足以下条件
  - 处理器P对单元X进行一次写之后又对单元X进行读，读和写之间没有其他处理器对单元X进行写，则P读到的值总是前面写进去的值。

- 处理器P对单元X进行写之后，另一处理器Q对单元X进行读，读和写之间无其他写，则Q读到的值应为P写进去的值。
- 对同一单元的写是串行化的，即任意两个处理器对同一单元的两次写，从各个处理器的角度看来顺序都是相同的。（写串行化 ）

➢ 在后面的讨论中，我们假设：

- 直到所有的处理器均看到了写的结果，这个写操作才算完成；
- 处理器的任何访存均不能改变写的顺序。就是说，允许处理器对读进行重排序，但必须以程序规定的顺序进行写。

8.2.2 实现一致性的基本方案

在一致的多处理机中，Cache提供两种功能：

- 共享数据的迁移

  减少了对远程共享数据的访问延迟，也减少了对共享存储器带宽的要求。

- 共享数据的复制

  不仅减少了访问共享数据的延迟，也减少了访问共享数据所产生的冲突。

一般情况下，小规模多处理机是采用硬件的方法来实现Cache的一致性。

1. Cache一致性协议

在多个处理器中用来维护一致性的协议。

- 关键：跟踪记录共享数据块的状态
- 两类协议（采用不同的技术跟踪共享数据的状态）
  - 目录式协议（directory）

    物理存储器中数据块的共享状态被保存在一个称为目录的地方。
  - 监听式协议（snooping）
    - 每个Cache除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享状态信息。

- Cache通常连在共享存储器的总线上，当某个Cache需要访问存储器时，它会把请求放到总线上广播出去，其他各个Cache控制器通过监听总线（它们一直在监听）来判断它们是否有总线上请求的数据块。如果有，就进行相应的操作。

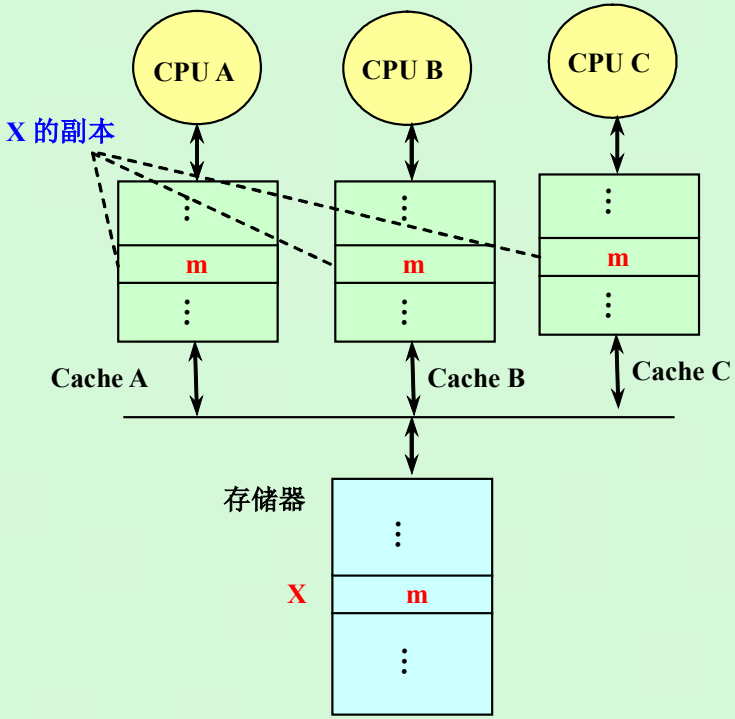
2. 有两种写协议来保证Cache一致性中的写操作。

➢ 写作废协议

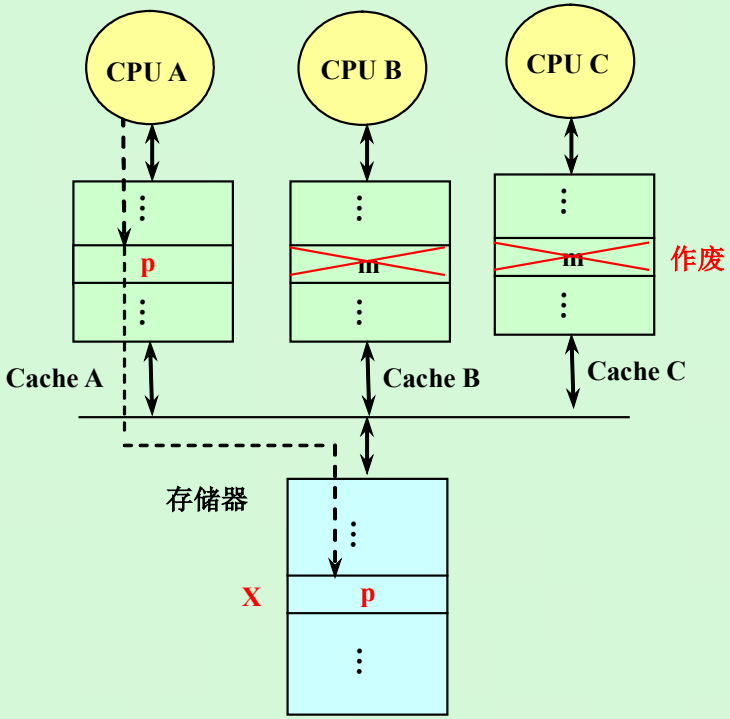
在处理器对某个数据项进行写入之前，保证它拥有对该数据项的唯一的访问权。(作废其他的副本)

例 监听总线、写作废协议举例（采用写直达法） 动画

初始状态：CPU A、CPU B、CPU C都有X的副本。在CPU A要对X进行写入时，需先作废CPU B和CPU C中的副本，然后再将p写入Cache A中的副本，同时用该数据更新主存单元X。



（a） CPU A 写入前



（b） CPU A 将 p 写入 X 后，作废其他 Cache 中的副本

➢ 写更新协议

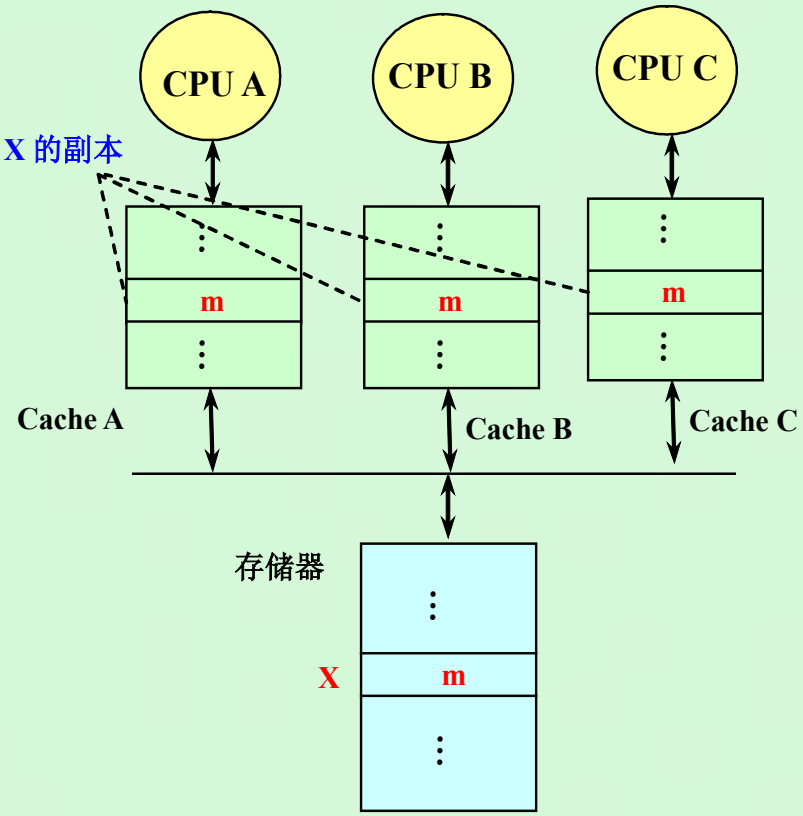
当一个处理器对某数据项进行写入时，通过广播使其他Cache中所有对应于该数据项的副本进行更新。

例 监听总线、写更新协议举例（采用写直达法）

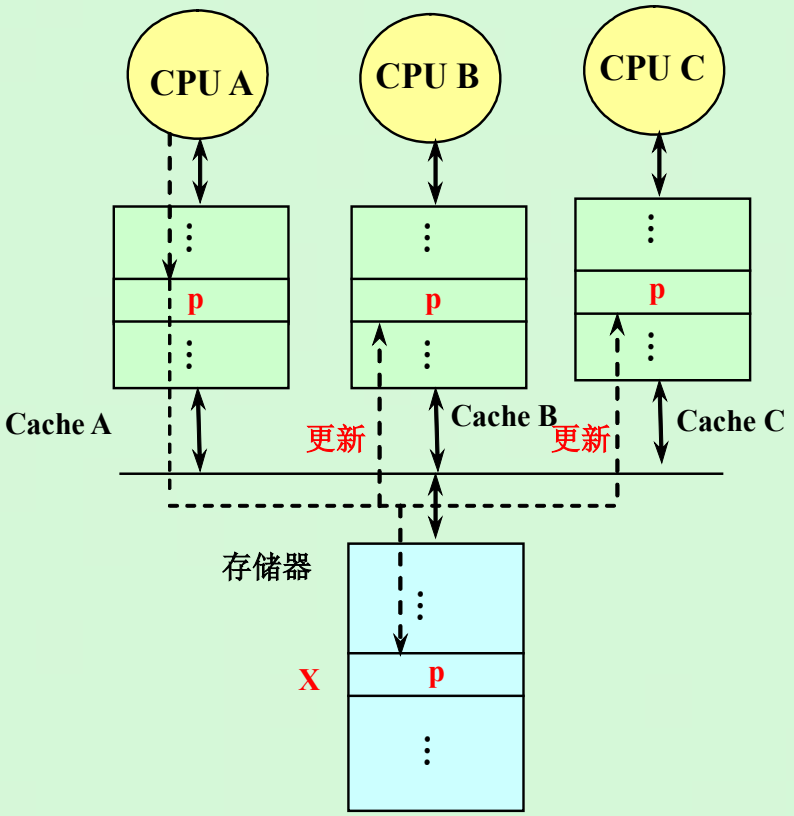
假设：3个Cache都有X的副本。

当CPU A将数据p写入Cache A中的副本时，将p广播给所有的Cache，这些Cache用p更新其中的副本。

由于这里是采用写直达法，所以CPU A还要将p写入存储器中的X。如果采用写回法，则不需要写入存储器。



（a） CPU A 写入前



（b） CPU A 将 p 写入 X 后，更新其他 Cache 中的副本

- 写更新和写作废协议性能上的差别主要来自：
  - 在对同一个数据进行多次写操作而中间无读操作的情况下，写更新协议需进行多次写广播操作，而写作废协议只需一次作废操作。
  - 在对同一Cache块的多个字进行写操作的情况下，写更新协议对于每一个写操作都要进行一次广播，而写作废协议仅在对该块的第一次写时进行作废操作即可。

    写作废是针对Cache块进行操作，而写更新则是针对字（或字节）进行。
  - 考虑从一个处理器A进行写操作后到另一个处理器B能读到该写入数据之间的延迟时间。

    写更新协议的延迟时间较小。

8.2.3 监听协议的实现

1. 监听协议的基本实现技术

- 实现监听协议的关键有3个方面
  - 处理器之间通过一个可以实现广播的互连机制相连。

    通常采用的是总线。
  - 当一个处理器的Cache响应本地CPU的访问时，如果它涉及全局操作，其Cache控制器就要在获得总线的控制权后，在总线上发出相应的消息。
  - 所有处理器都一直在监听总线，它们检测总线上的地址在它们的Cache中是否有副本。若有，则响应该消息，并进行相应的操作 。

- 写操作的串行化：由总线实现

（获取总线控制权的顺序性）

2. Cache发送到总线上的消息主要有以下两种：
   - **RdMiss**——读不命中
   - **WtMiss**——写不命中
- 需要通过总线找到相应数据块的最新副本，然后调入本地Cache中。
  - **写直达Cache：**因为所有写入的数据都同时被写回主存，所以从主存中总可以取到其最新值。
  - 对于**写回Cache**，得到数据的最新值会困难一些，因为最新值可能在某个Cache中，也可能在主存中。

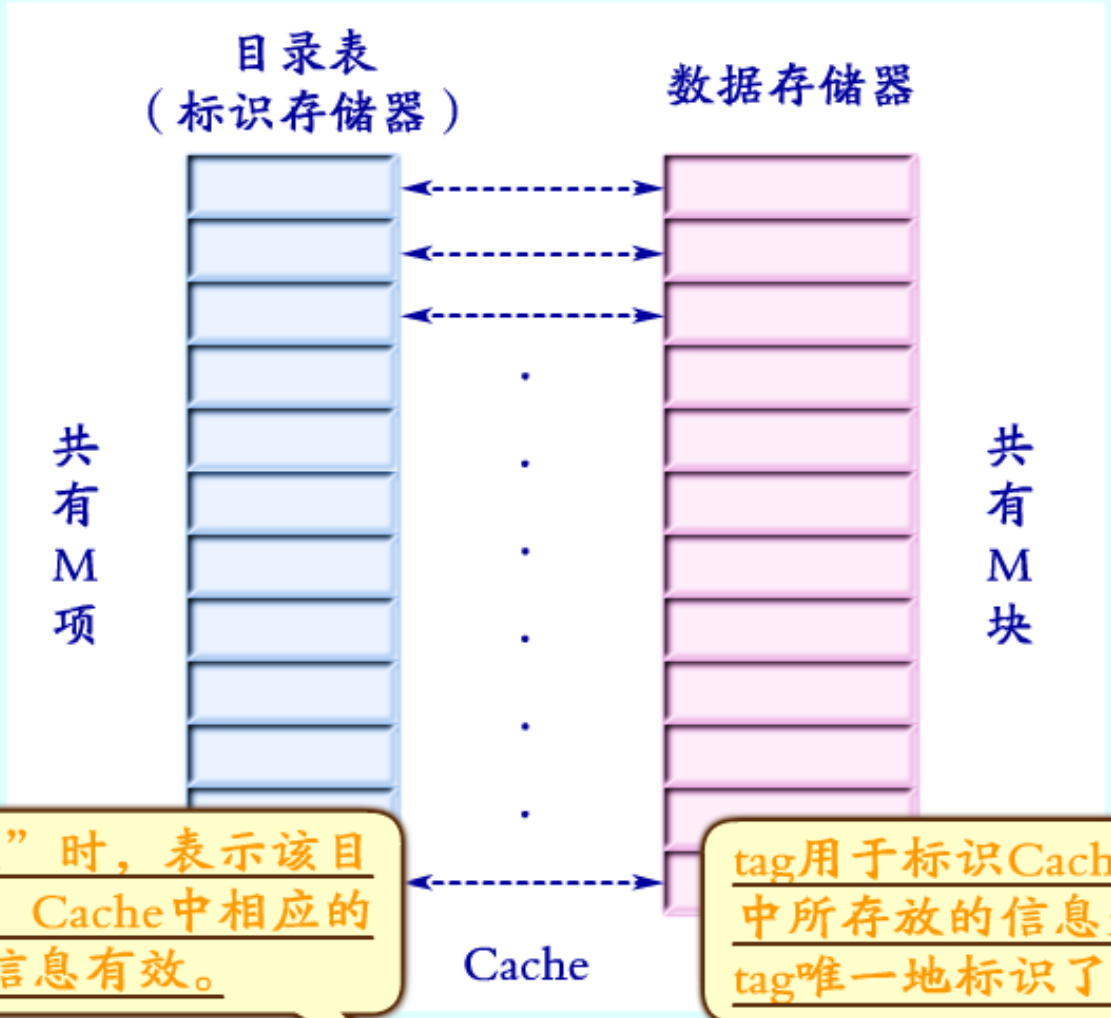
（后面的讨论中，只考虑写回法Cache）

- 有的监听协议还增设了一条Invalidate（作废）消息，用来通知其他各处理器作废其Cache中相应的副本。
  - 与WtMiss的区别：Invalidate不引起调块
- Cache的标识（tag）可直接用来实现监听。
- 作废一个块只需将其有效位置为0，即为无效。
- 给每个Cache块增设一个共享位
  - 为“1”：该块是被多个处理器所共享
  - 为“0”：仅被某个处理器所独占

块的拥有者：拥有该数据块的唯一副本的处理器。

有效位和共享位的组合可以是：00,01,10,11。其中：00和01 代表 该块无效。10代表该块为独占（已修改），11代表该块为共享。所以说，一个块有3中不同的状态：无效，已修改，共享

Cache目录表的结构



当该位为“1”时，表示该目录表项有效，Cache中相应的块所包含的信息有效。

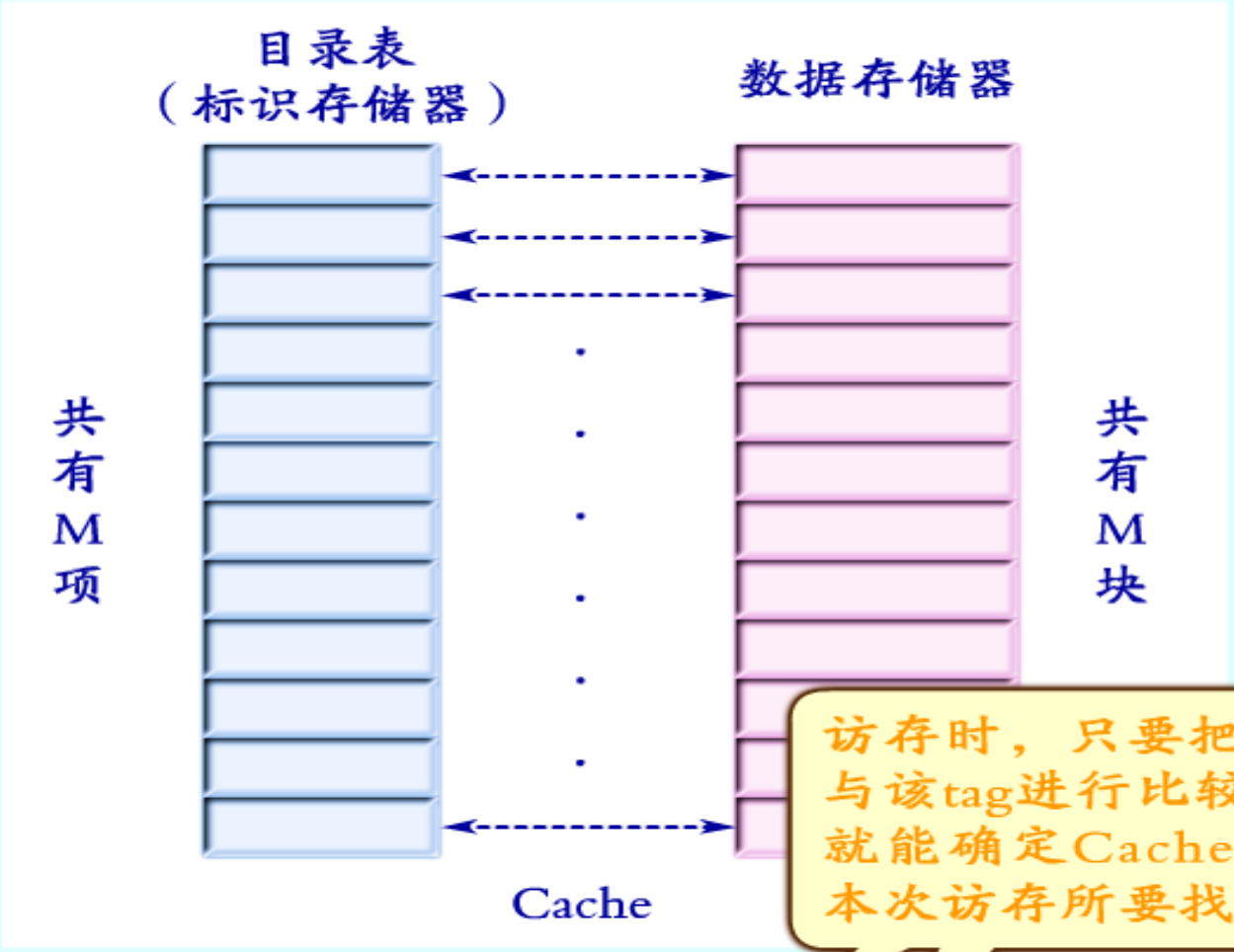
tag用于标识Cache中相应的块位置中所存放的信息是哪个主存块的。tag唯一地标识了一个主存块。

目录表项：

有效位	标识tag

共享位

Cache目录表的结构

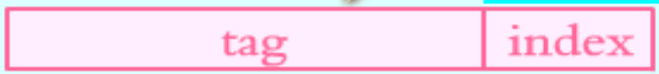


访存时，只要把访存地址中的高位与该tag进行比较，判断是否相等，就能确定Cache中相应的块是否是本次访存所要找的块。

目录表项：

共享位

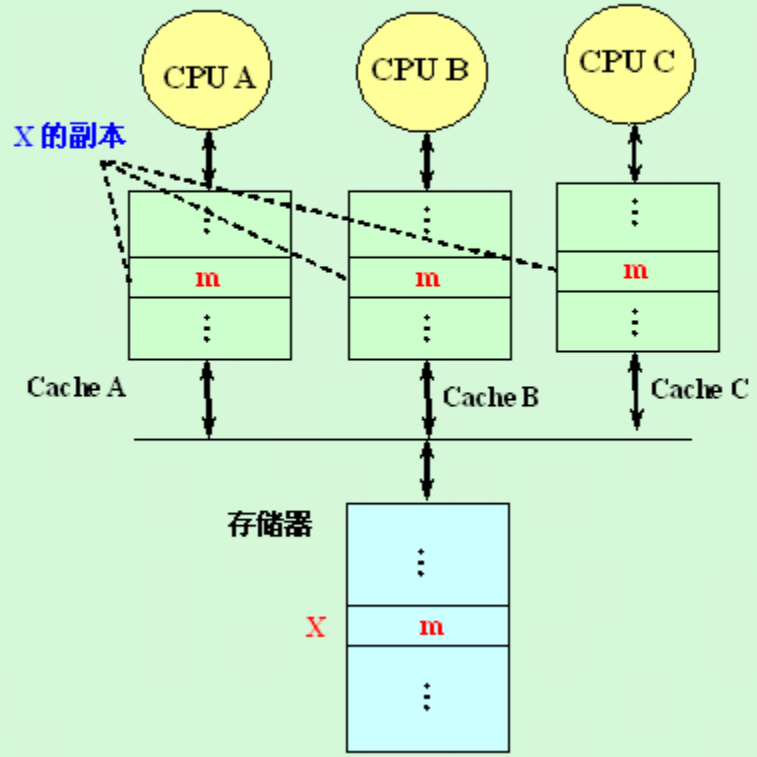
访存地址：



3. 监听协议举例

- 在每个结点内嵌入一个有限状态控制器。
  - 该控制器根据来自处理器或总线的请求以及Cache块的状态，做出相应的响应。
- 每个数据块的状态取以下3种状态中的一种：
  - 无效（简称I）：Cache中该块的内容为无效。
  - 共享（简称S）：该块可能处于共享状态。
    - 在多个处理器中都有副本。这些副本都相同，且与存储器中相应的块相同。
  - 已修改（简称M）：该块已经被修改过，并且还没写入存储器。（块中的内容是最新的，系统中唯一的最新副本）

有效位和共享位的组合可以是：00,01,10,11。其中：00和01 代表该块无效。10代表该块为独占（已修改），11代表该块为共享。所以说，一个块有3中不同的状态：无效，已修改，共享



有效位和共享位的组合可以是：00,01,10,11。其中：00和01 代表该块无效。10代表该块为独占（已修改），11代表该块为共享。所以说，一个块有3中不同的状态：无效，已修改，共享

状态

	有效位和共享位
无效，	0 0 或者 0 1
已修改，	1 0
共享	1 1

有限状态控制器

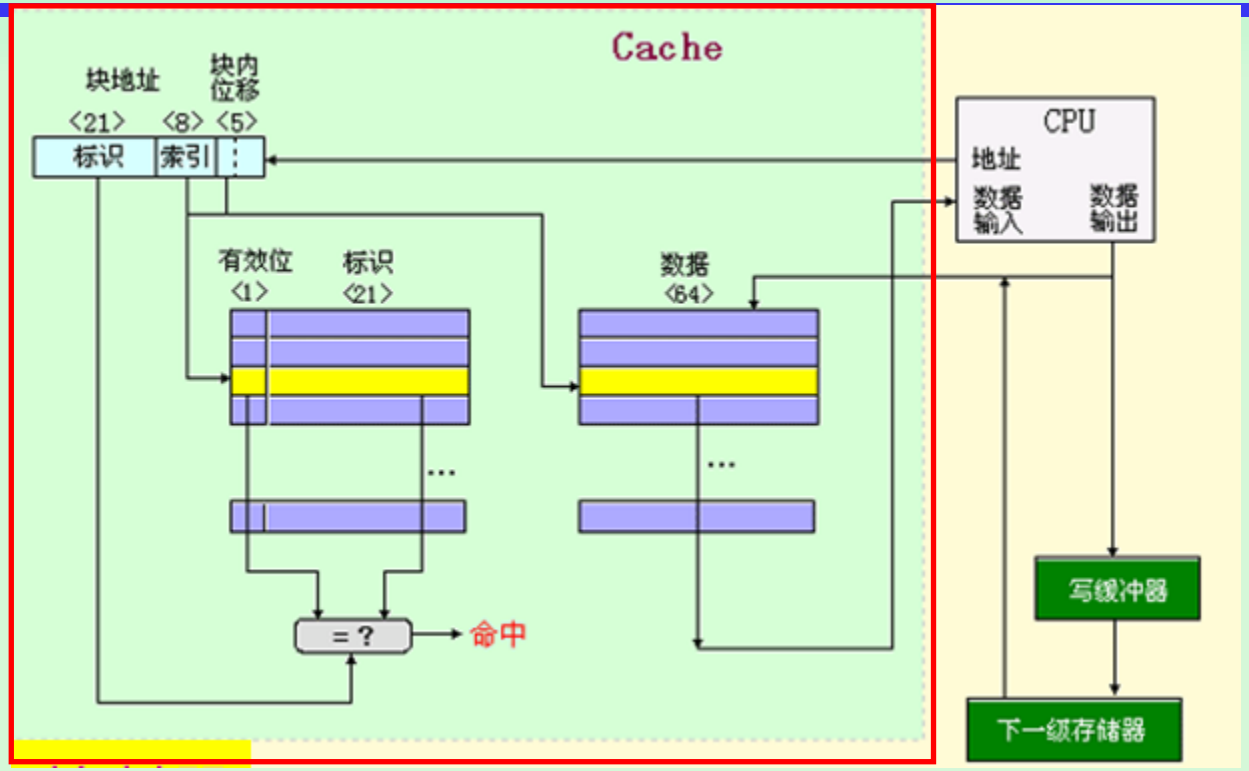
1位	1位	数据块B

有效位和共享位

目录表项：

有效位	标识tag	共享位

监听协议



有限状态控制器

目录表项：

有效位	标识tag	共享位

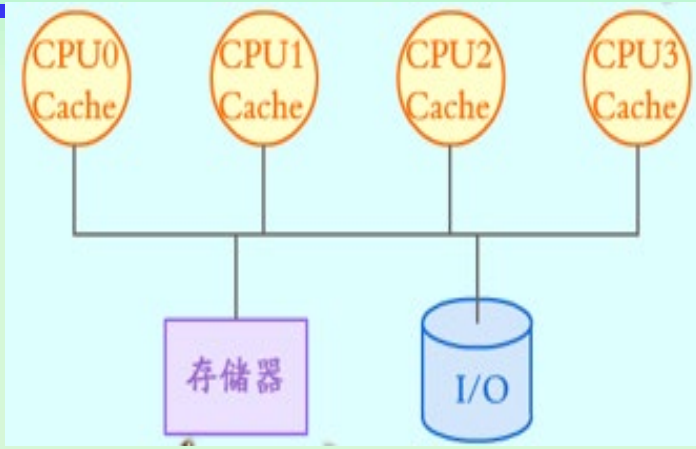
状态

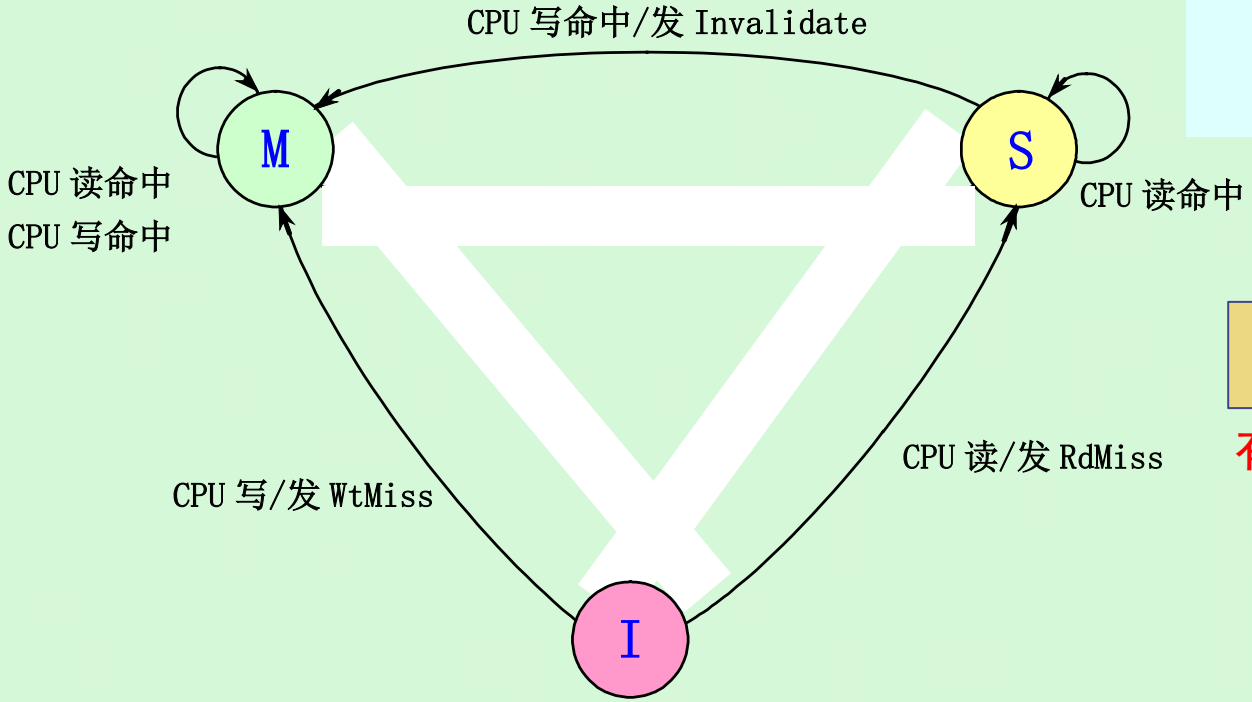
有效位和共享位

无效，	00 或者 01
已修改，	10
共享	11

下面来讨论在各种情况下监听协议所进行的操作。

- 响应来自处理器的请求
  - 不发生替换的情况





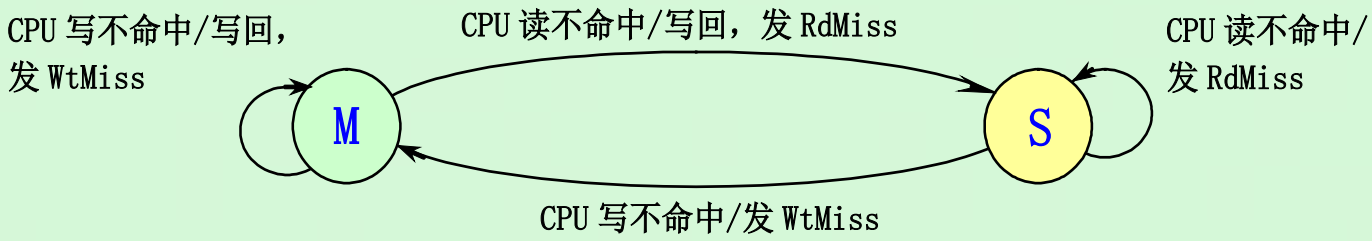
写作废协议中（采用写回法），Cache块的状态转换图

1位	1位	数据块B

有效位和共享位

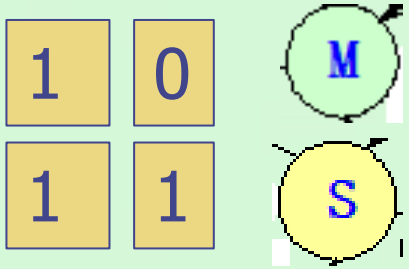
有效位	共享位	状态
0	X	I
1	0	M
1	1	S

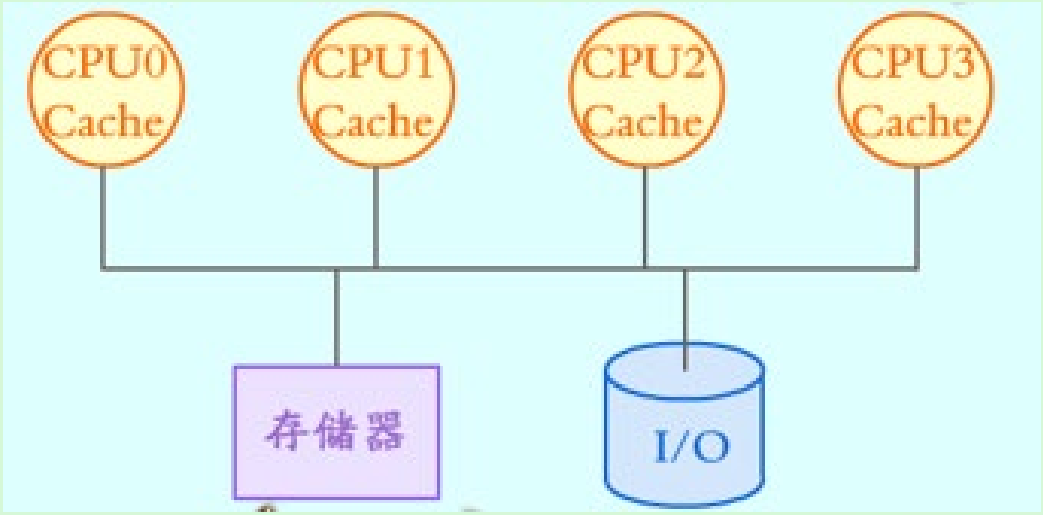
- 发生替换的情况



写作废协议中（采用写回法），Cache块的状态转换图

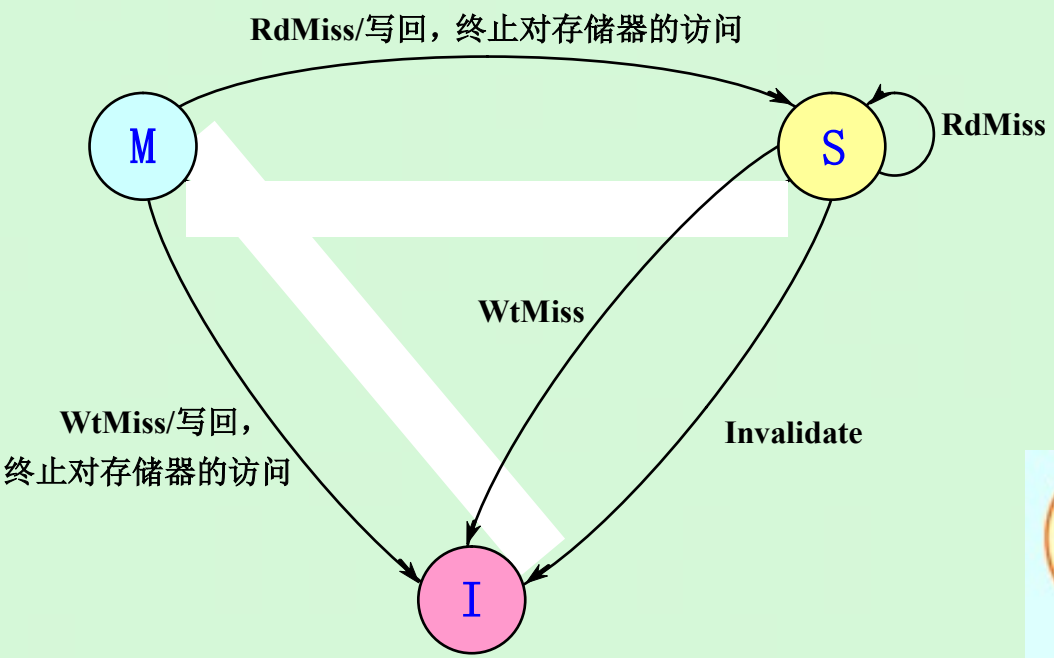
有效位和共享位 状态

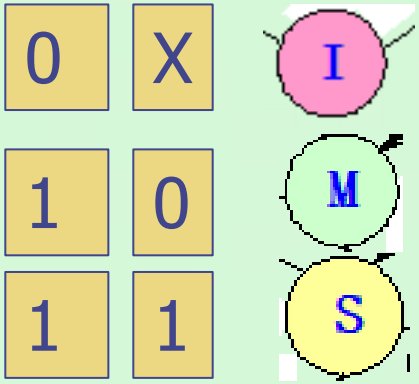


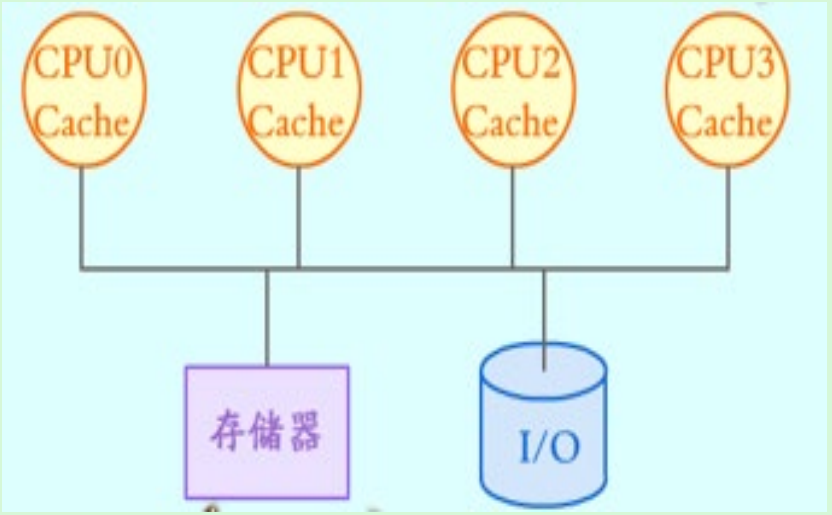


➢ 响应来自总线的请求

- 每个处理器都在监视总线上的消息和地址，当发现有与总线上的地址相匹配的Cache块时，就要根据该块的状态以及总线上的消息，进行相应的处理。有效位和共享位 状态







写作废协议中（采用写回法），Cache块的状态转换图

8.3 分布式共享存储器系统结构

8.3.1 目录协议的基本思想

- 广播和监听的机制使得监听一致性协议的可扩放性很差。
- 寻找替代监听协议的一致性协议。

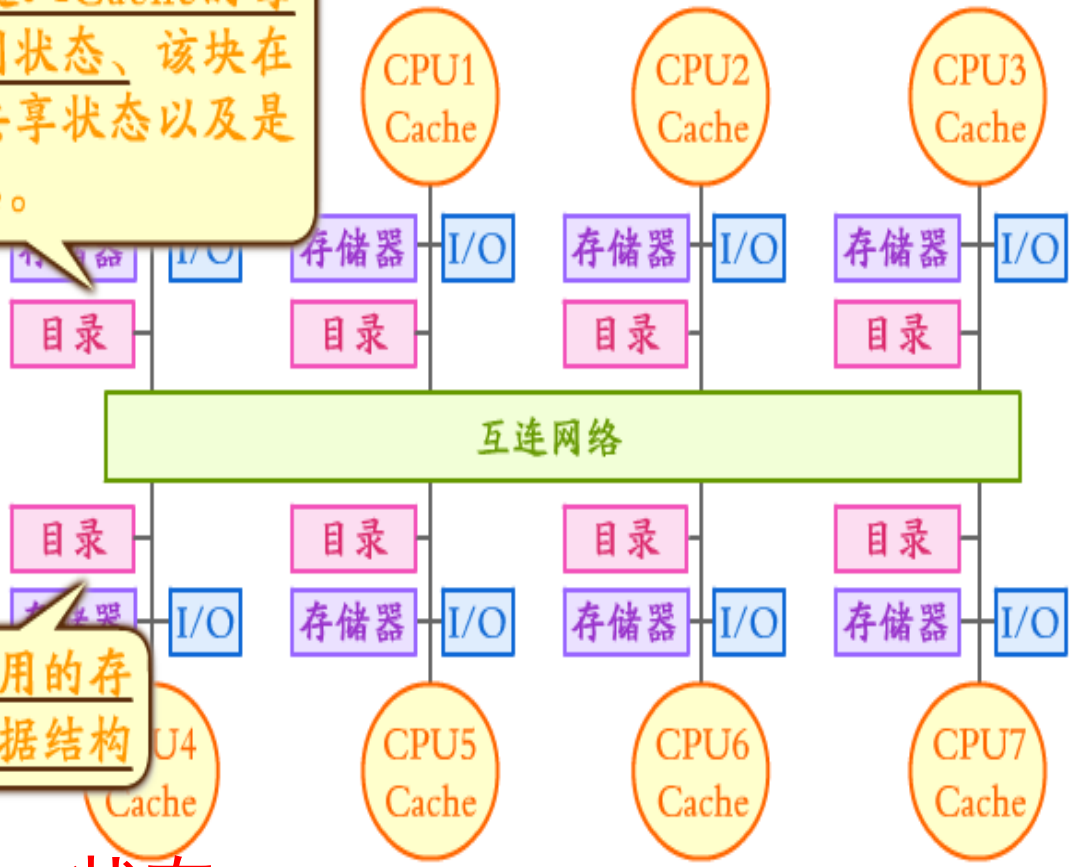
（采用目录协议）

1. 目录协议

- 目录：一种集中的数据结构。对于存储器中的每一个可以调入Cache的数据块，在目录中设置一条目录项，用于记录该块的状态以及哪些Cache中有副本等相关信息。

目录协议

它记录着可以进入Cache的每个数据块的访问状态、该块在各个处理器的共享状态以及是否修改过等信息。



目录是用一种专用的存储器所记录的数据结构

有效位和共享位 状态

0	X	I
1	0	M
1	1	S

1位	1位	数据块B	共享集S

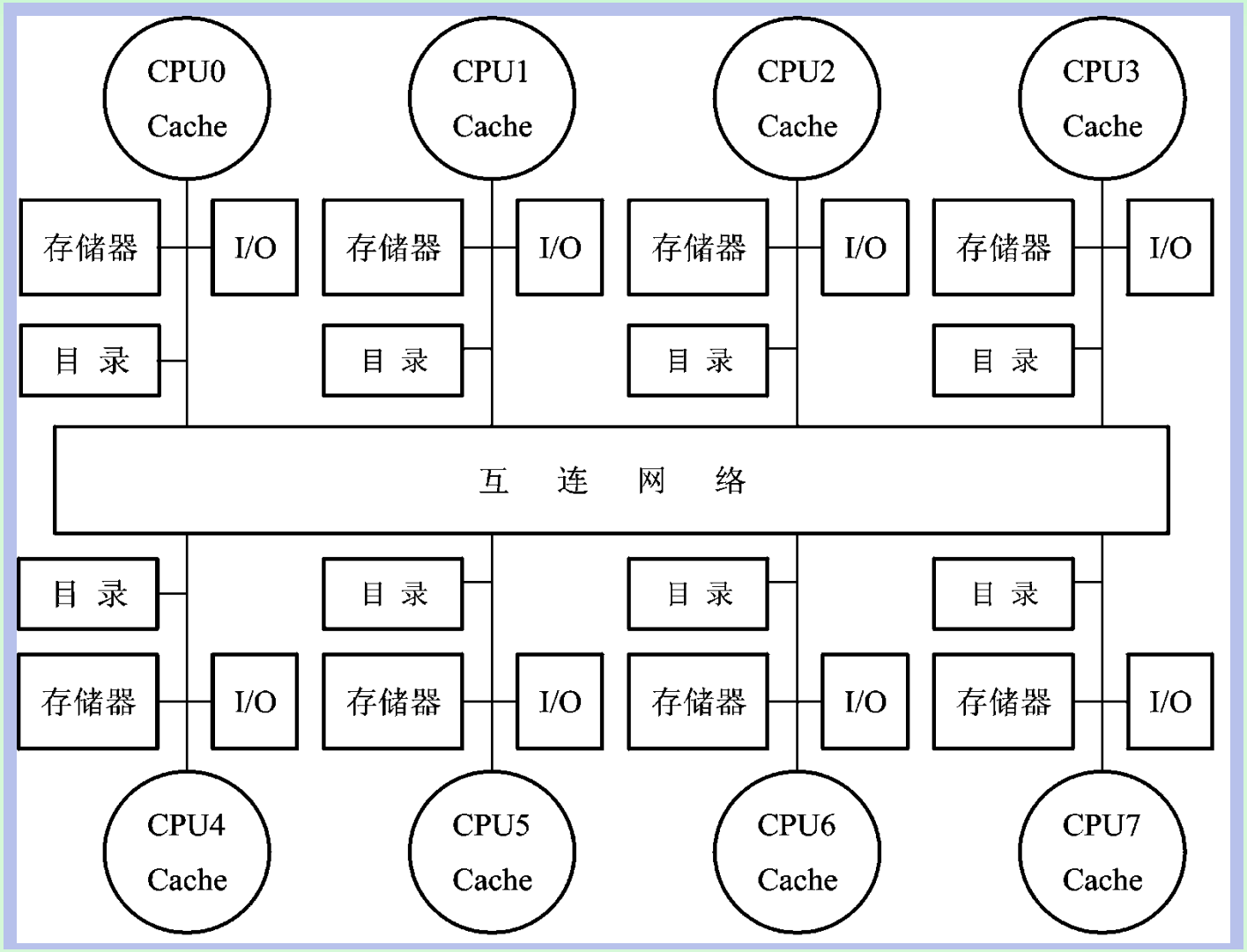
有效位和共享位

- 特点：

  对于任何一个数据块，都可以快速地在唯一的一个位置中找到相关的信息。这使一致性协议避免了广播操作。

- 位向量：记录哪些Cache中有副本。
  - 每一位对应于一个处理器。
  - 长度与处理器的个数成正比。
  - 由位向量指定的处理机的集合称为共享集S。
- 分布式目录
  - 目录与存储器一起分布到各结点中，从而对于不同目录内容的访问可以在不同的结点进行。

- 对每个结点增加目录后的分布式存储器多处理机



- 目录法最简单的实现方案：对于存储器中每一块都在目录中设置一项。目录中的信息量与M×N成正比。

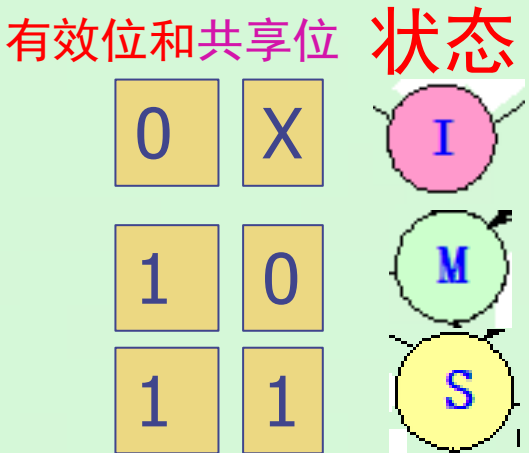
  其中：

  - M：存储器中存储块的总数量
  - N：处理器的个数
  - 由于M=K×N， K是每个处理机中存储块的数量，所以如果K保持不变，则目录中的信息量就与N^2成正比。

2. 在目录协议中，存储块的状态有3种：

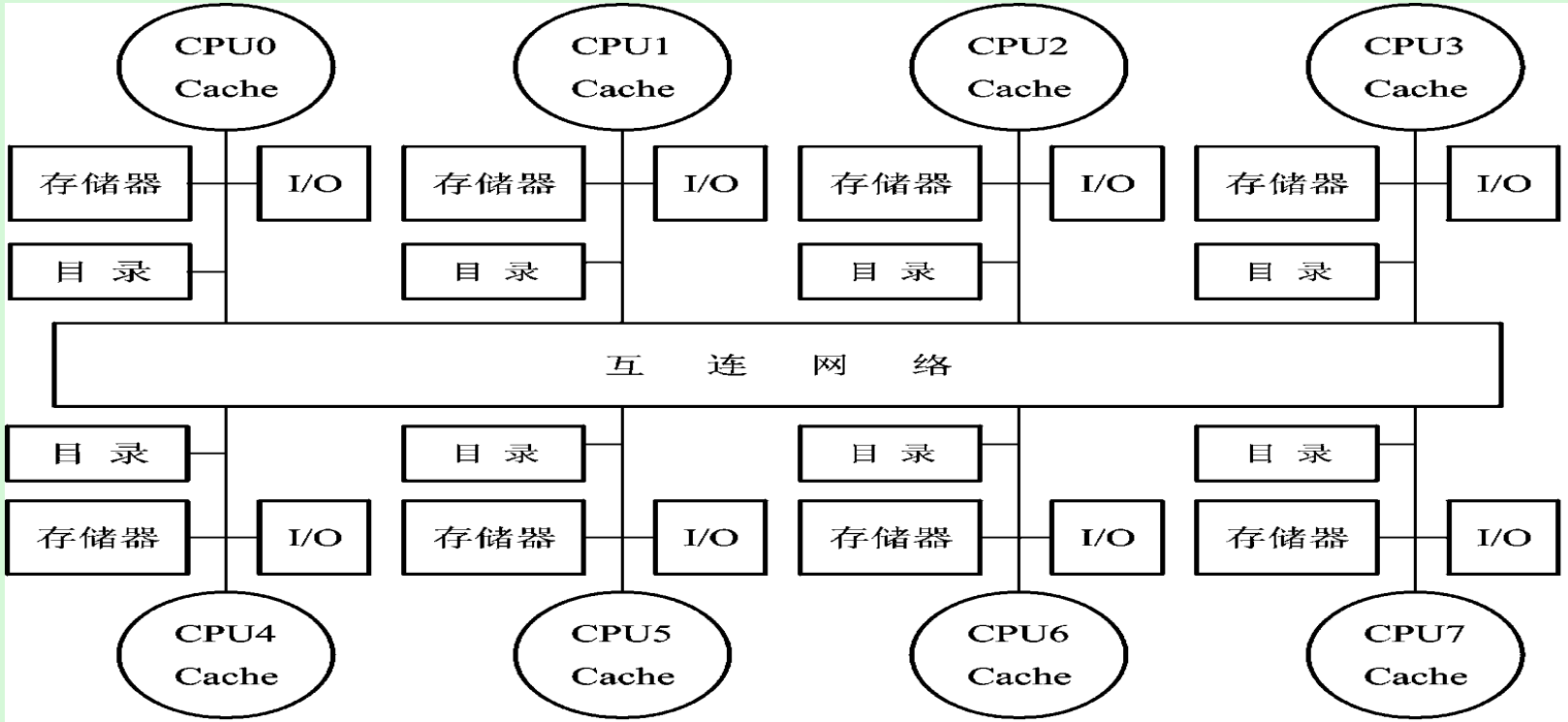
- 未缓冲：该块尚未被调入Cache。所有处理器的Cache中都没有这个块的副本。
- 共享：该块在一个或多个处理机上有这个块的副本，且这些副本与存储器中的该块相同。
- 独占：仅有一个处理机有这个块的副本，且该处理机已经对其进行了写操作，所以其内容是最新的，而存储器中该块的数据已过时。

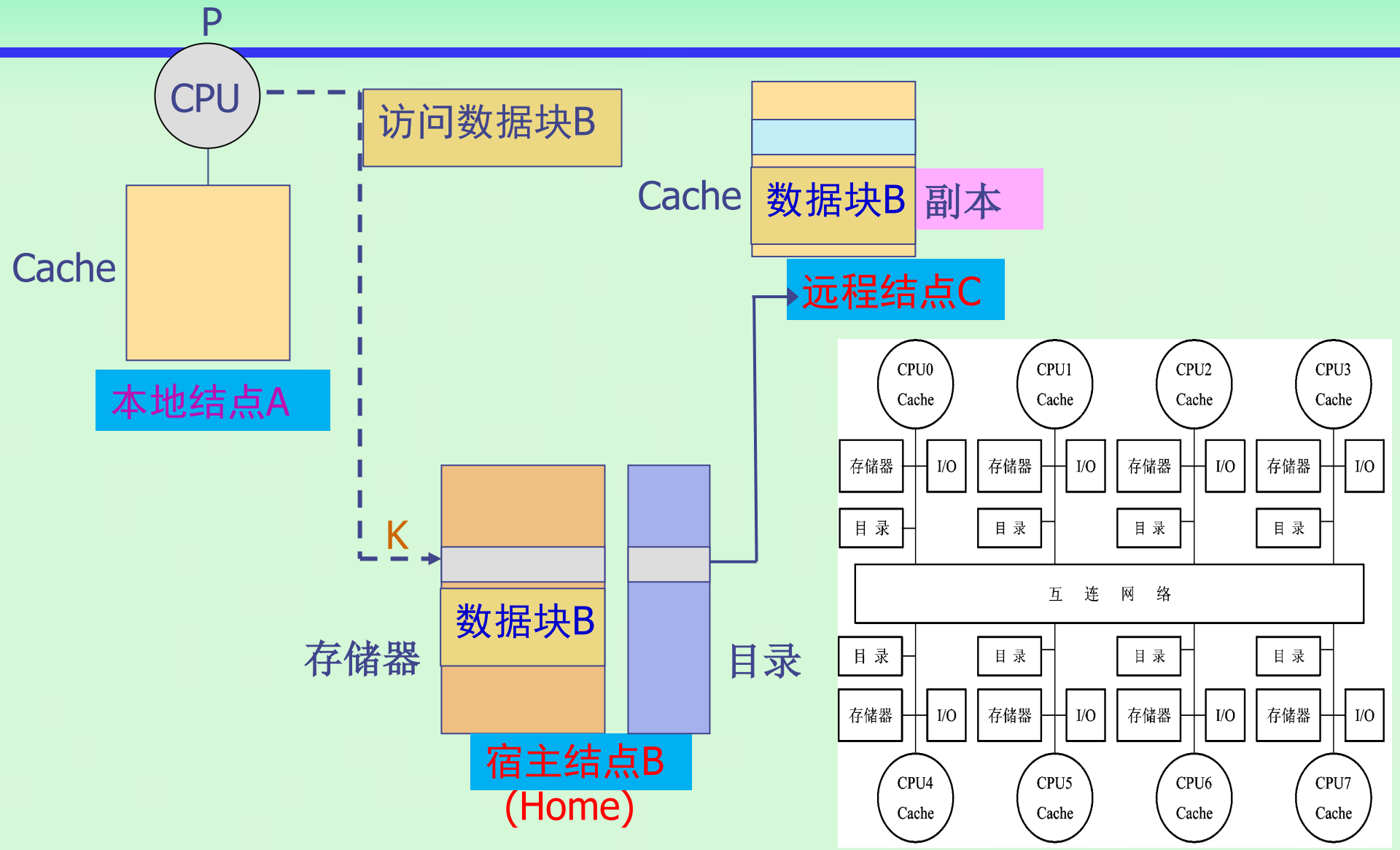
这个处理机称为该块的拥有者。



3. 本地结点、宿主结点以及远程结点的关系

- 本地结点：发出访问请求的结点
- 宿主结点：包含所访问的存储单元及其目录项的结点
- 远程结点可以和宿主结点是同一个结点，也可以不是同一个结点。





宿主结点：存放有对应地址的存储器块和目录项的结点

4. 在结点之间发送的消息

- 本地结点发给宿主结点（目录）的消息

说明：括号中的内容表示所带参数。

P：发出请求的处理机编号

K：所要访问的地址

- **RdMiss（P，K）**

处理机P读取地址为A的数据时不命中，请求宿主结点提供数据（块），并要求把P加入共享集。

- **WtMiss（P，K）**

处理机P对地址A进行写入时不命中，请求宿主结点提供数据，并使P成为所访问数据块的独占者。

- **Invalidate（K）**

  请求向所有拥有相应数据块副本（包含地址*K*）的远程Cache发Invalidate消息，作废这些副本。

➢ 宿主结点（目录）发送给远程结点的消息

- **Invalidate（K）**

  作废远程Cache中包含地址K的数据块。

- **Fetch（K）**

  从远程Cache中取出包含地址*K*的数据块，并将之送到宿主结点。把远程Cache中那个块的状态改为“共享”。

- **Fetch&Inv（K）**

- 从远程Cache中取出包含地址*K*的数据块，并将之送到宿主结点。然后作废远程Cache中的那个块。

➢ 宿主结点发送给本地结点的消息

DReply（D）

- D表示数据内容。
- 把从宿主存储器获得的数据返回给本地Cache。

➢ 远程结点发送给宿主结点的消息

WtBack（K，D）

- 把远程Cache中包含地址*K*的数据块写回到宿主结点中， 该消息是远程结点对宿主结点发来的“取数据”或“取/作废”消息的响应。

- 本地结点发送给被替换块的宿主结点的消息
  - **MdSharer（P，K）**

    用于当本地Cache中需要替换一个包含地址K的块、且该块未被修改过的情况。这个消息发给该块的宿主结点，请求它将P从共享集中删除。如果删除后共享集变为空集，则宿主结点还要将该块的状态改变为“未缓存”（U）。
  - **WtBack2（P，K，D）**

    用于当本地Cache中需要替换一个包含地址K的块、且该块已被修改过的情况。这个消息发给该块的宿主结点，完成两步操作：①把该块写回；②进行与MdSharer相同的操作。