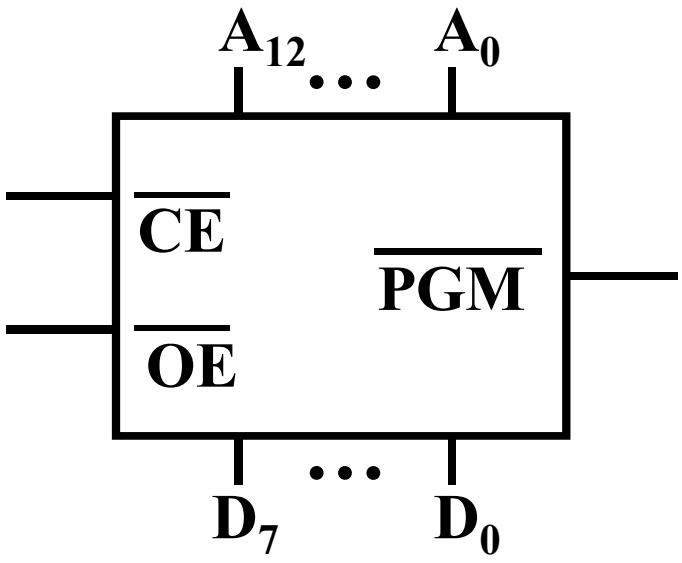
计算机组成原理

翁睿

哈尔滨工业大学

例 4.3 设 CPU 有 20 根地址线，8 根数据线。并用 IO/$\overline{M}$ 作访存控制信号。$\overline{RD}$ 为读命令，$\overline{WR}$ 为写命令。现有 2764 EPROM (8K × 8位), 外特性如下：



2. 存储器与 CPU 的连接: 解题流程

(1) 写出各区域对应的二进制地址码

(2) 确定芯片的数量及类型

(3) 分配（芯片的）地址线

(4) 确定片选信号

(5) 画出连接图（注意 特殊信号线 的处理）

用 138 译码器及其他门电路（门电路自定）画出 CPU 和 2764 的连接图。要求地址为 F0000H~FFFFFH , 并写出每片 2764 的地址范围。

六、存储器的校验

4.2

1. 奇偶校验码

2. 汉明校验码（汉明码 / 海明码）

1. 编码的最小距离 (汉明距离)

任意两组合法代码之间 二进制位数 的 最少差异

编码的纠错 、检错能力与编码的最小距离有关

$$L - 1 = D + C \ (D \geq C)$$

L —— 编码的最小距离　　$L = 3$

D —— 检测错误的位数　　具有 一位 纠错能力

C —— 纠正错误的位数

汉明码是具有一位纠错能力的编码

2 . 汉明码的组成

组成汉明码的三要素

汉明码的组成需增添？位检测位

$$2^k \geqslant n + k + 1$$

检测位的位置？

$$2^i \ (i = 0、1、2、3 \ldots\ldots)$$

检测位的取值？

检测位的取值与该位所在的检测“小组”中承担的奇偶校验任务有关

例4.4 求 0101 按“偶校验”配置的汉明码

解：∵ $n=4$

根据 $2^k \geq n+k+1$

得 $k=3$

汉明码排序如下：

二进制序号	1	2	3	4	5	6	7
名称	C_1	C_2	0	C_4	1	0	1
	0	1		0			

∴ 0101 的汉明码为 0100101

4. 汉明码的纠错过程

形成新的检测位 P_i　其位数与增添的检测位有关

如增添 3 位 （$k=3$）　新的检测位为 $P_4\,P_2\,P_1$

以 $k=3$ 为例，P_i 的取值为

$$P_1 = \overset{C_1}{b1} \oplus b3 \oplus b5 \oplus b7$$

$$P_2 = \overset{C_2}{b2} \oplus b3 \oplus b6 \oplus b7$$

$$P_4 = \overset{C_4}{b4} \oplus b5 \oplus b6 \oplus b7$$

对于按“偶校验”配置的汉明码

不出错时　$P_1=0$，$P_2=0$，$P_4=0$

例4.5　已知接收到的汉明码为 0100111（按配偶原则配置）试问要求传送的信息是什么？

解：纠错过程如下

P_1=b1⊕ b3⊕ b5 ⊕ b7 = 0 无错

P_2=b2⊕ b3⊕ b6⊕ b7 = 1 有错

P_4=b4⊕ b5⊕ b6⊕ b7 = 1 有错

∴　$P_4P_2P_1$ = 110

第 6 位出错，可纠正为 0100101，

故要求传送的信息为 0101。

练习2 写出按偶校验配置的汉明码

0101101 的纠错过程

$P_4 = b4 \oplus b5 \oplus b6 \oplus b7 = 1$

$P_2 = b2 \oplus b3 \oplus b6 \oplus b7 = 0$

$P_1 = b1 \oplus b3 \oplus b5 \oplus b7 = 0$

$\therefore\ P_4 P_2 P_1 = 100$ 第 4 位错，可不纠

练习3 按配奇原则配置 0011 的汉明码

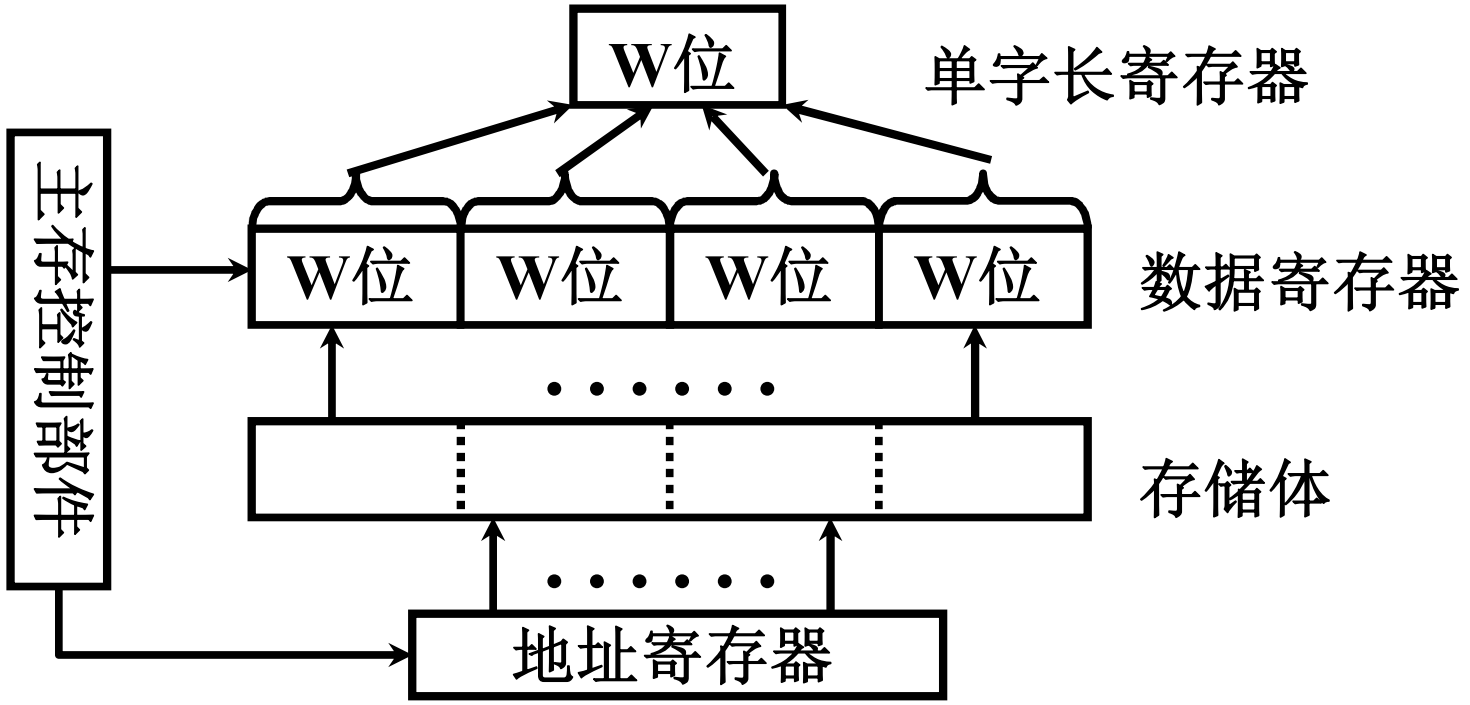
配奇的汉明码为 0101011

七、提高访存速度的措施

- 采用高速器件
- 采用层次结构 Cache — 主存
- 调整主存结构

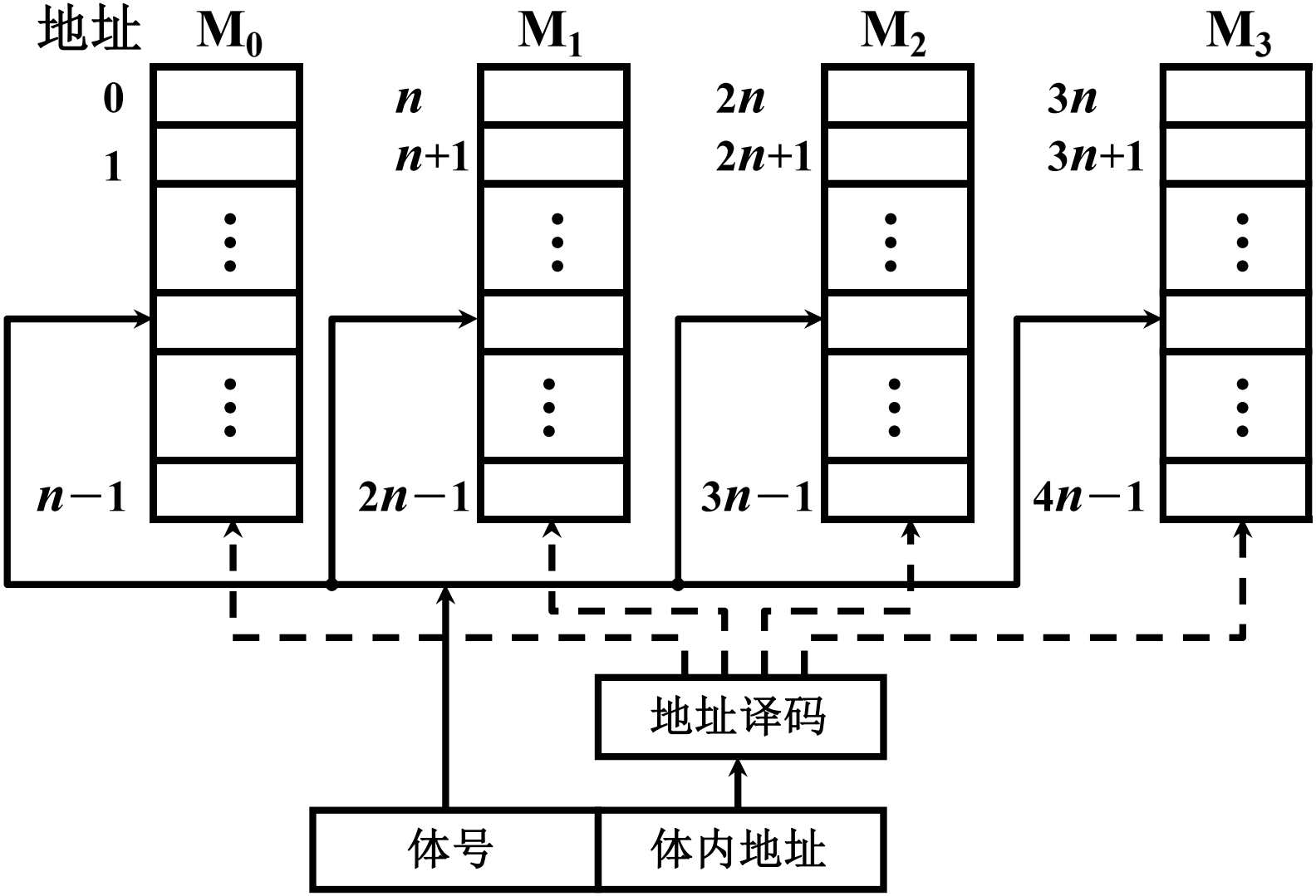
1. 单体多字系统

增加存储器的带宽



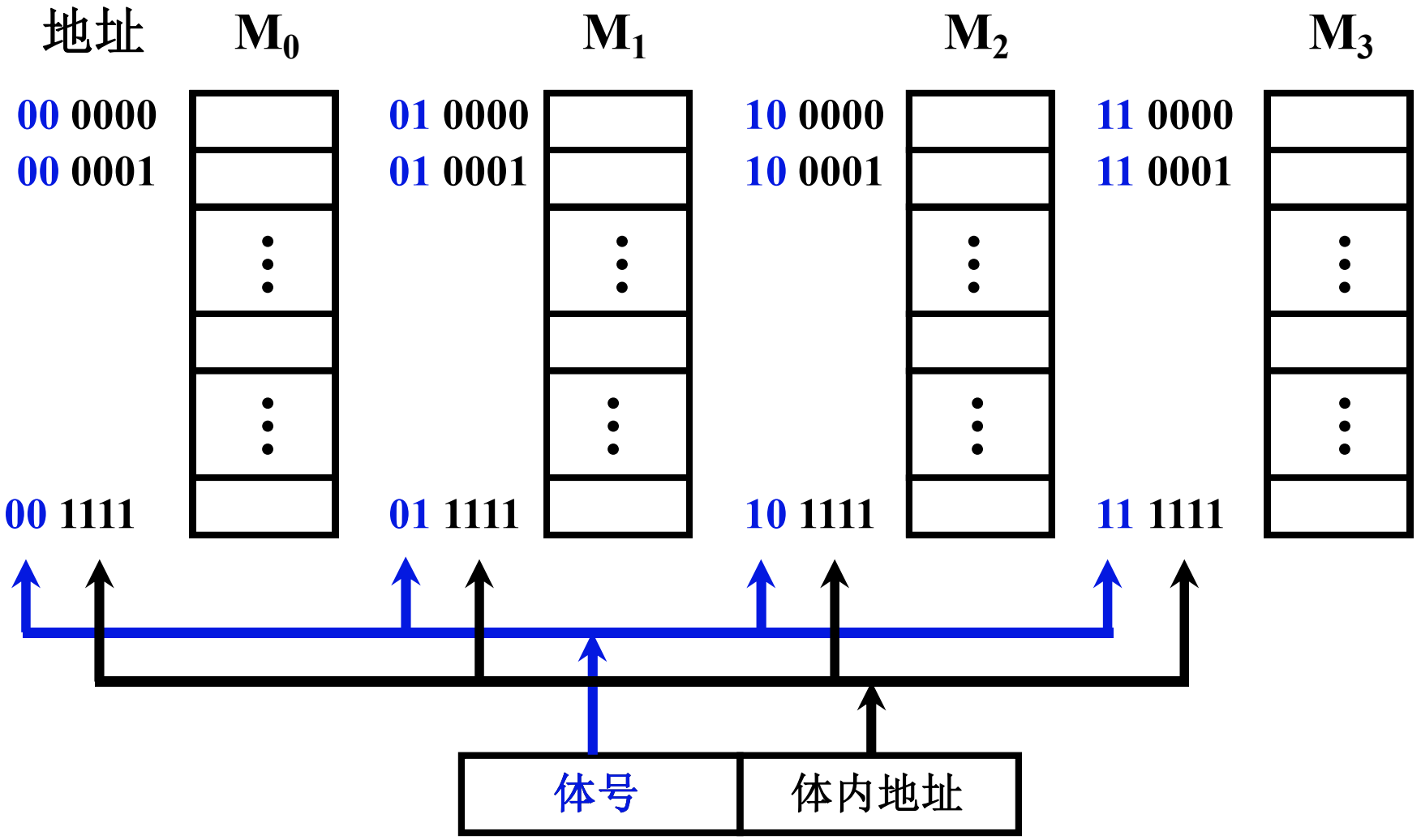
2. 多体并行系统

(1) 高位交叉　　各存储体连续编址



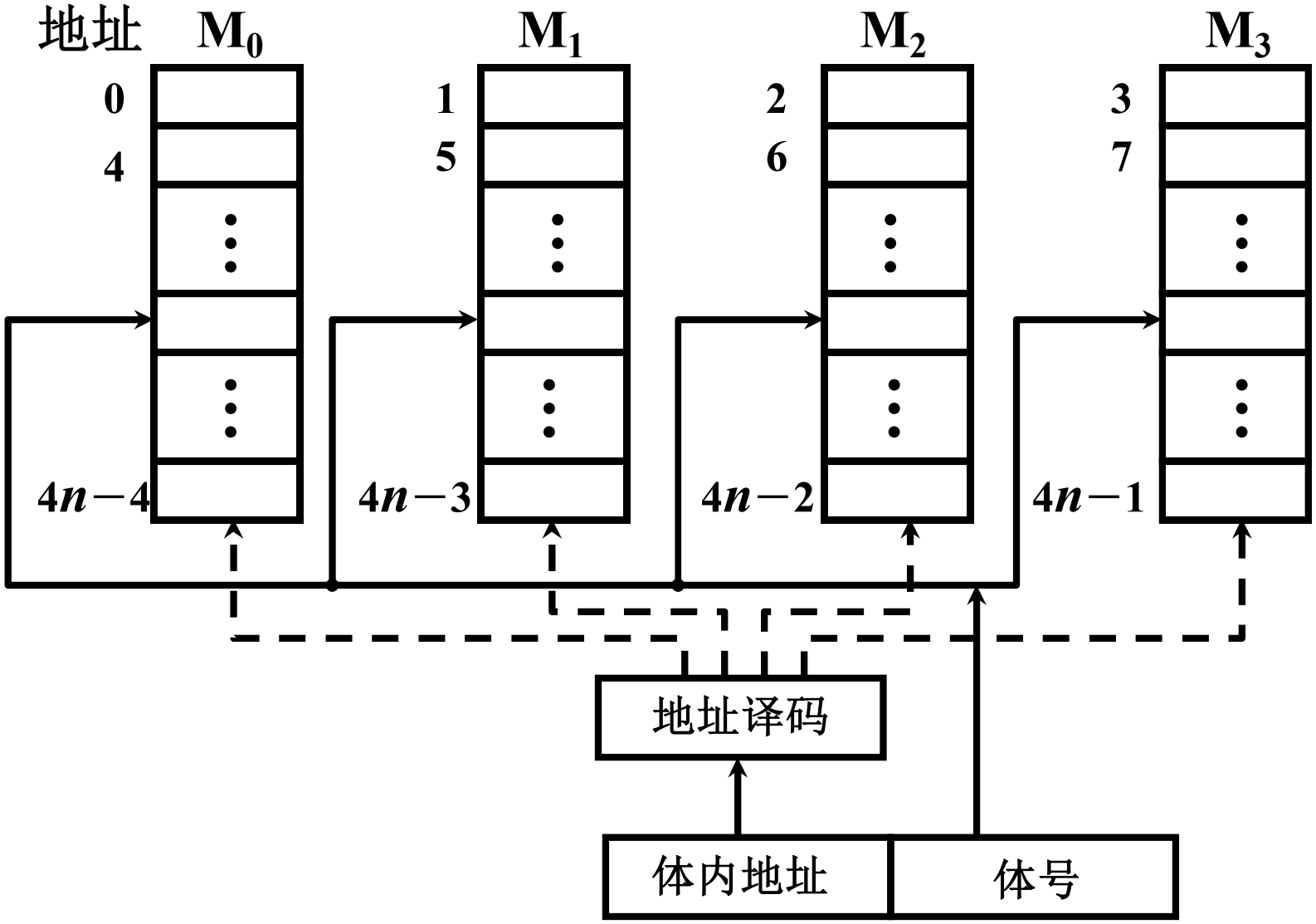
2. 多体并行系统

(1) 高位交叉　　各存储体连续编址



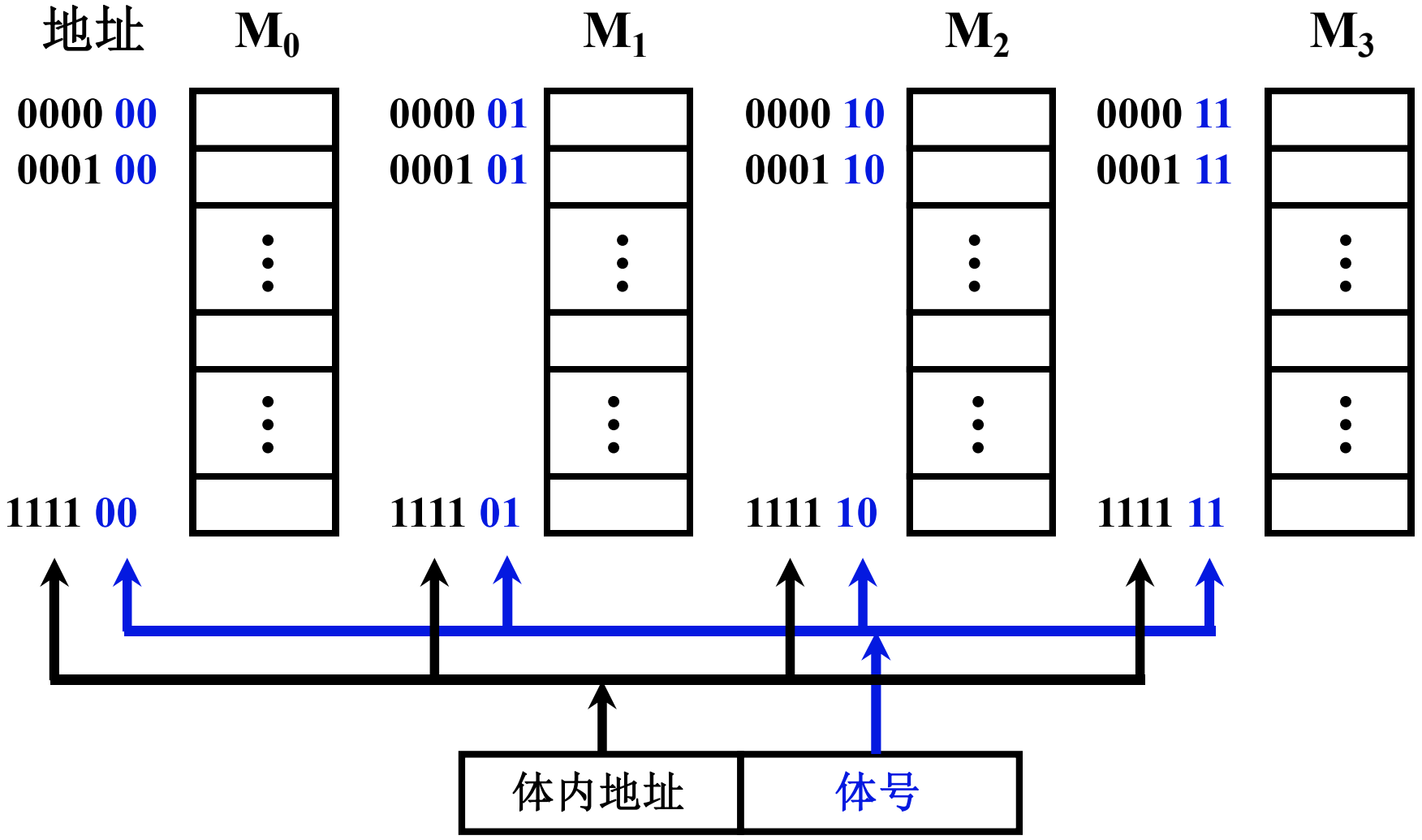
(2) 低位交叉

各存储体轮流编址



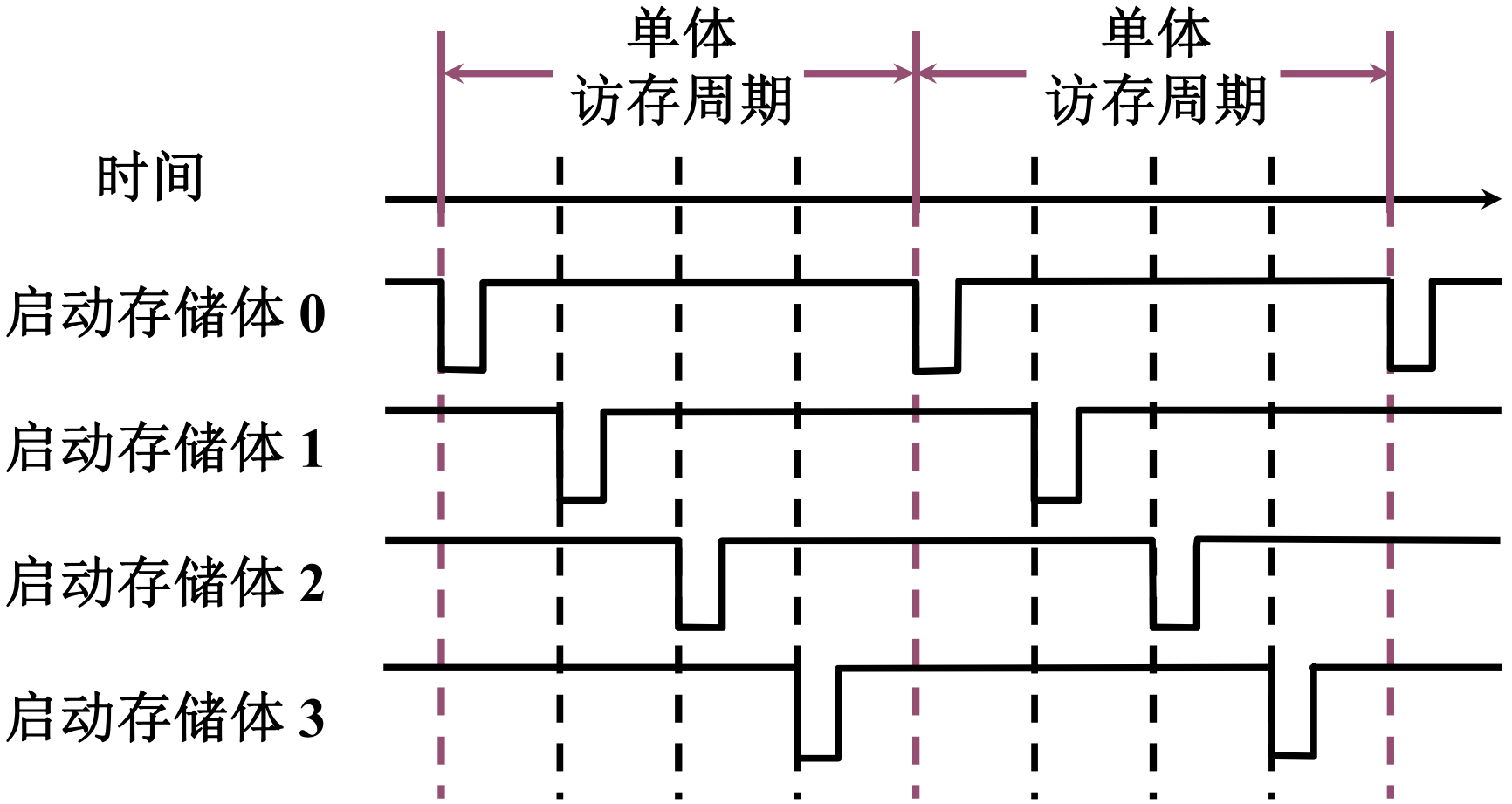
(2) 低位交叉

各存储体轮流编址



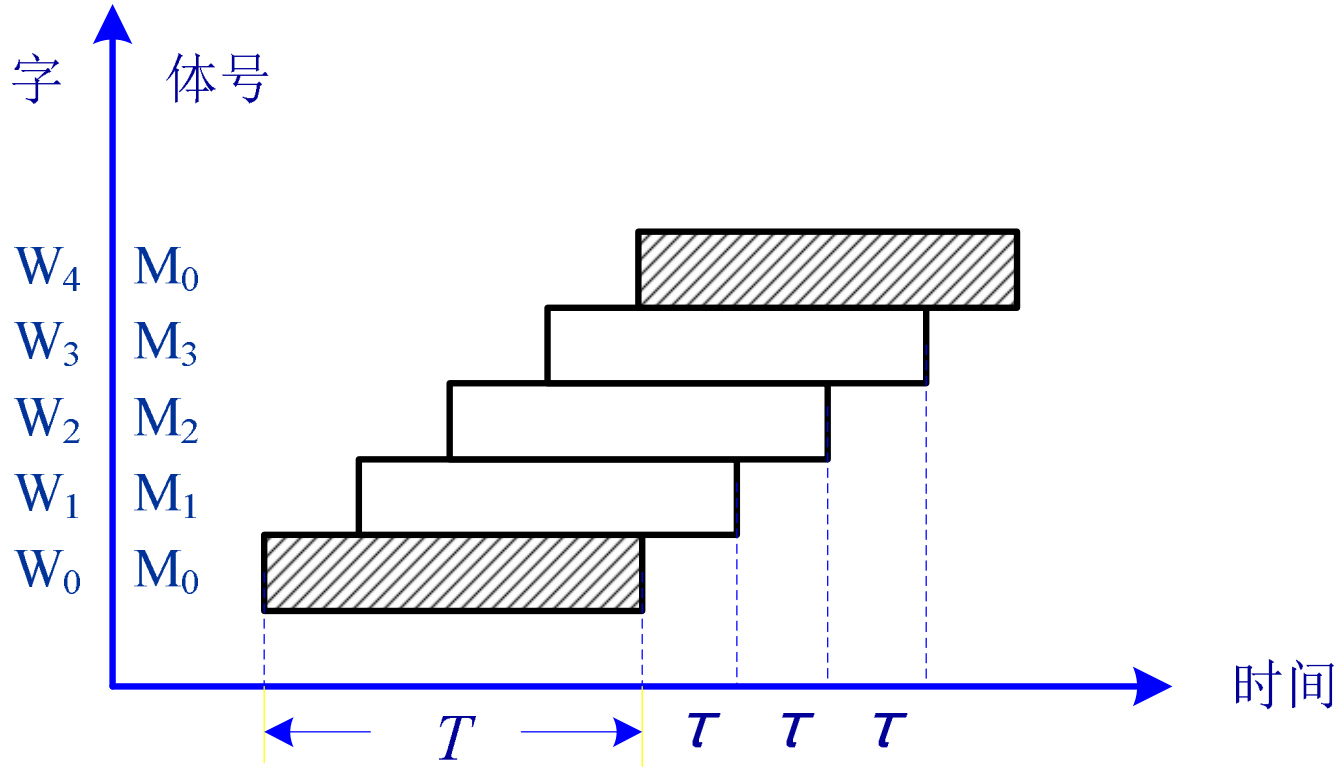
低位交叉的特点

在不改变存取周期的前提下，增加存储器的带宽



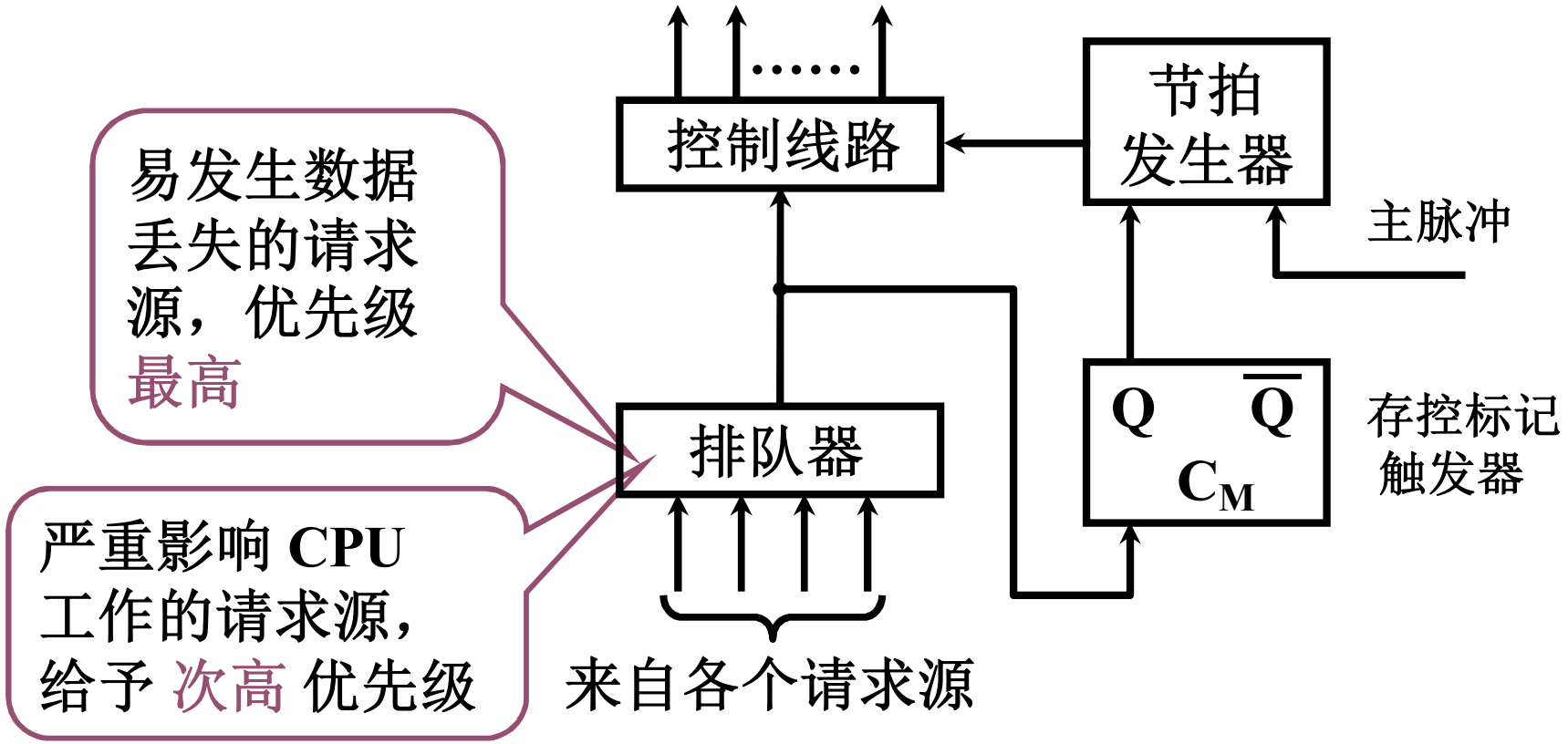
4.2

设四体低位交叉存储器，存取周期为T，总线传输周期为τ，为实现流水线方式存取，应满足 $T = 4\tau$。



连续读取 4 个字所需的时间为 $T+(4-1)\tau$

(3) 存储器控制部件（简称存控）



3.高性能存储芯片

(1) SDRAM (同步 DRAM)

在系统时钟的控制下进行读出和写入

CPU 无须等待

(2) RDRAM

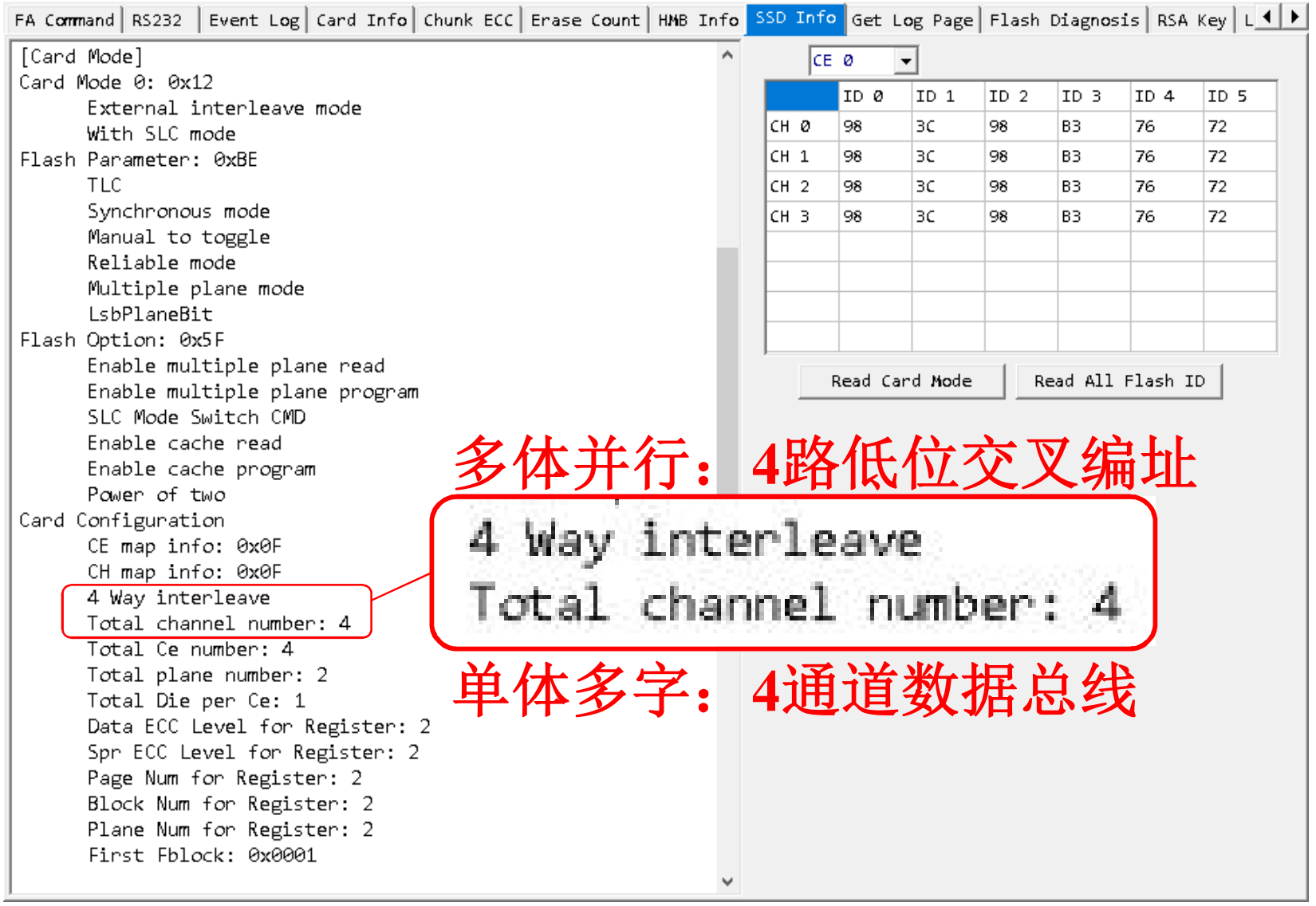
由 Rambus 公司 开发，提升了位宽和工作频率

主要解决 存储器带宽 问题 （现已淘汰）

(3) 带 Cache 的 DRAM

在 DRAM 的芯片内 集成 了一个由 SRAM 组成的 Cache，有利于 猝发式读取

存储器提速技术在辅存中的应用：NVME SSD 4.2



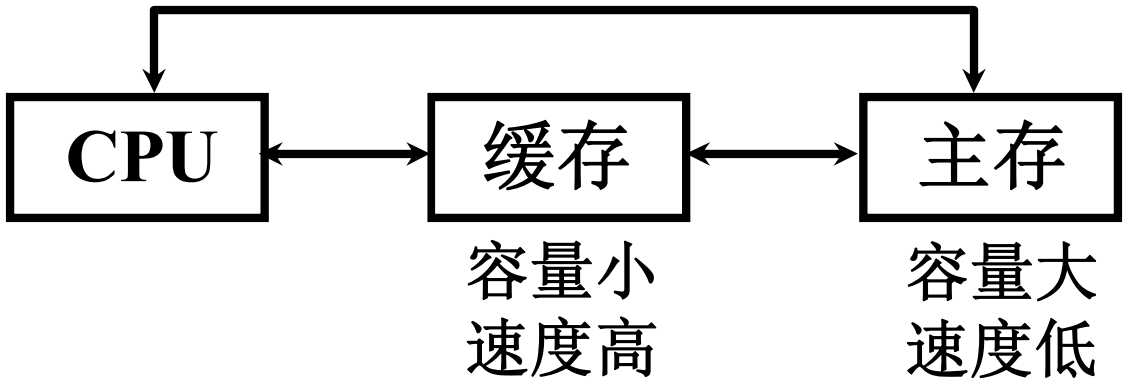
4.3 高速缓冲存储器

一、概述

1. 问题的提出

避免 CPU “空等” 现象

CPU 和主存（DRAM）的速度差异



程序访问的局部性原理

(1) 程序访问的局部性原理

对于绝大多数程序来说，程序所访问的指令和数据在地址上不是均匀分布的，而是相对簇聚的。

程序访问的局部性包含两个方面：

- 时间局部性：程序马上将要用到的信息很可能就是现在正在使用的信息。
- 空间局部性：程序马上将要用到的信息很可能与现在正在使用的信息在存储空间上是相邻的。

(2) 局部性举例

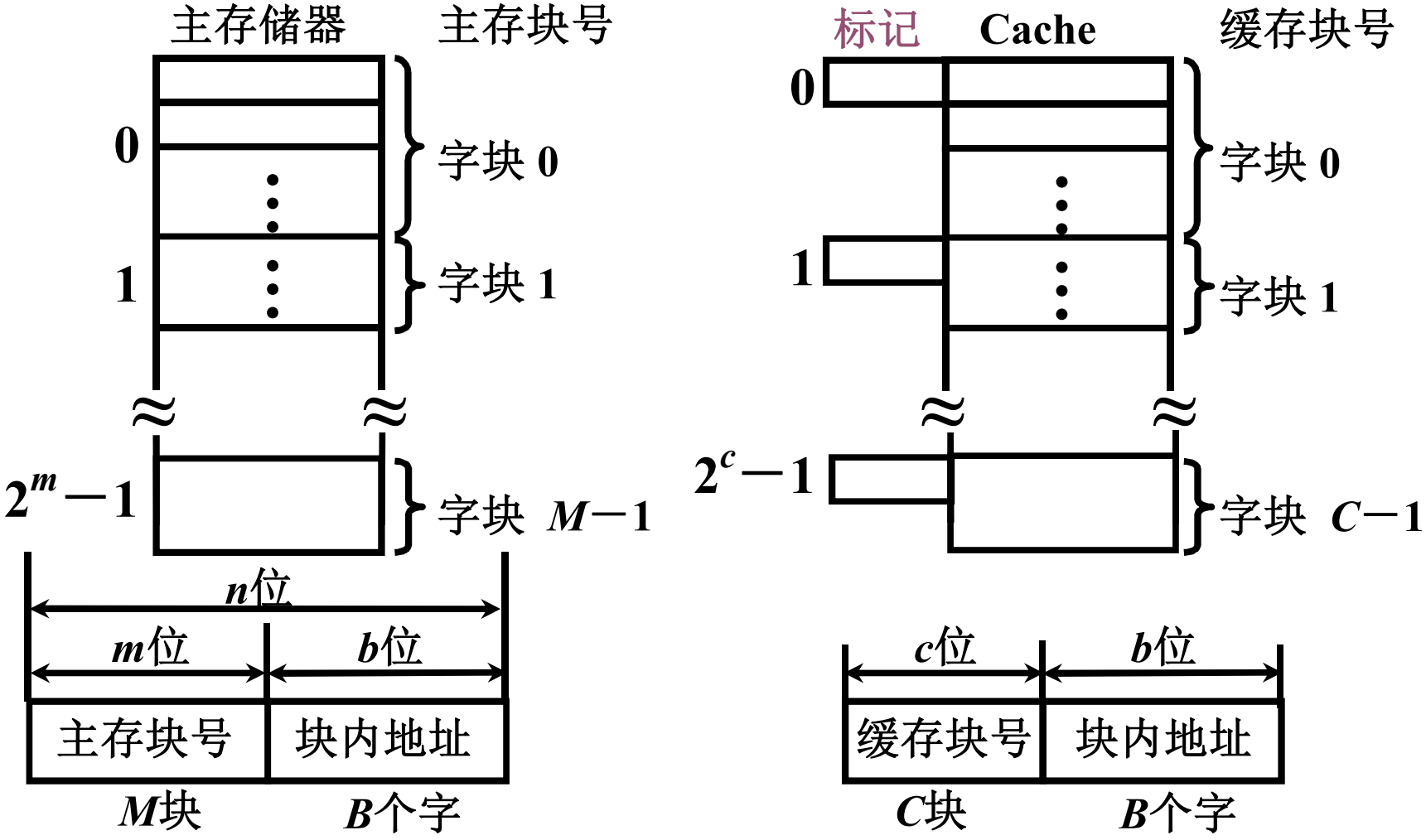

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- 对数据的引用
  - 顺序访问数组元素
    (步长为1的引用模式)　　空间局部性
  - 变量sum在每次循环迭代中被引用一次　　时间局部性

- 对指令的引用
  - 顺序读取指令　　空间局部性
  - 重复循环执行for循环体　　时间局部性

2. Cache 的工作原理

(1) 主存和缓存的编址



主存和缓存按块存储　　块的大小相同　　*B* 为块长

(2) 命中与未命中

缓存共有 C 块

主存共有 M 块　$M >> C$

命中　主存块 已调入 缓存

主存块与缓存块 建立 了对应关系

用 标记 记录与某缓存块建立了对应关系的 主存块块号

未命中　主存块 未调入 缓存

主存块与缓存块 未建立 对应关系

(3) Cache 的命中率

CPU 欲访问的信息在 Cache 中的 比率

命中率 与 Cache 的 容量 与 块长 有关

一般每块可取 4 至 8 个字

块长取一个存取周期内从主存调出的信息长度

CRAY_1　16体交叉　块长取 16 个存储字

IBM 370/168　4体交叉　块长取 4 个存储字

（64位×4＝256位）

(3) Cache 的命中率和平均访存时间 4.3

命中率： $H=\dfrac{N_1}{N_1+N_2}$

N_1 —— 访问Cache的次数

N_2 —— 访问主存的次数

不命中率： $F=1-H$

设 Cache 命中率 为 h，访问 Cache 的时间为 t_c，

访问 主存 的时间为 t_m

平均访存时间 $t_a=h\times t_c+(1-h)\times t_m$

(4) Cache –主存系统的效率

效率 e 与 命中率 有关

$$e = \frac{\text{访问 Cache 的时间}}{\text{平均访问时间}} \times 100\%$$

则 $$e = \frac{t_c}{h \times t_c + (1-h) \times t_m} \times 100\%$$

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?

   (映射规则　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　如何在映射规则 规定的候选位置查找)

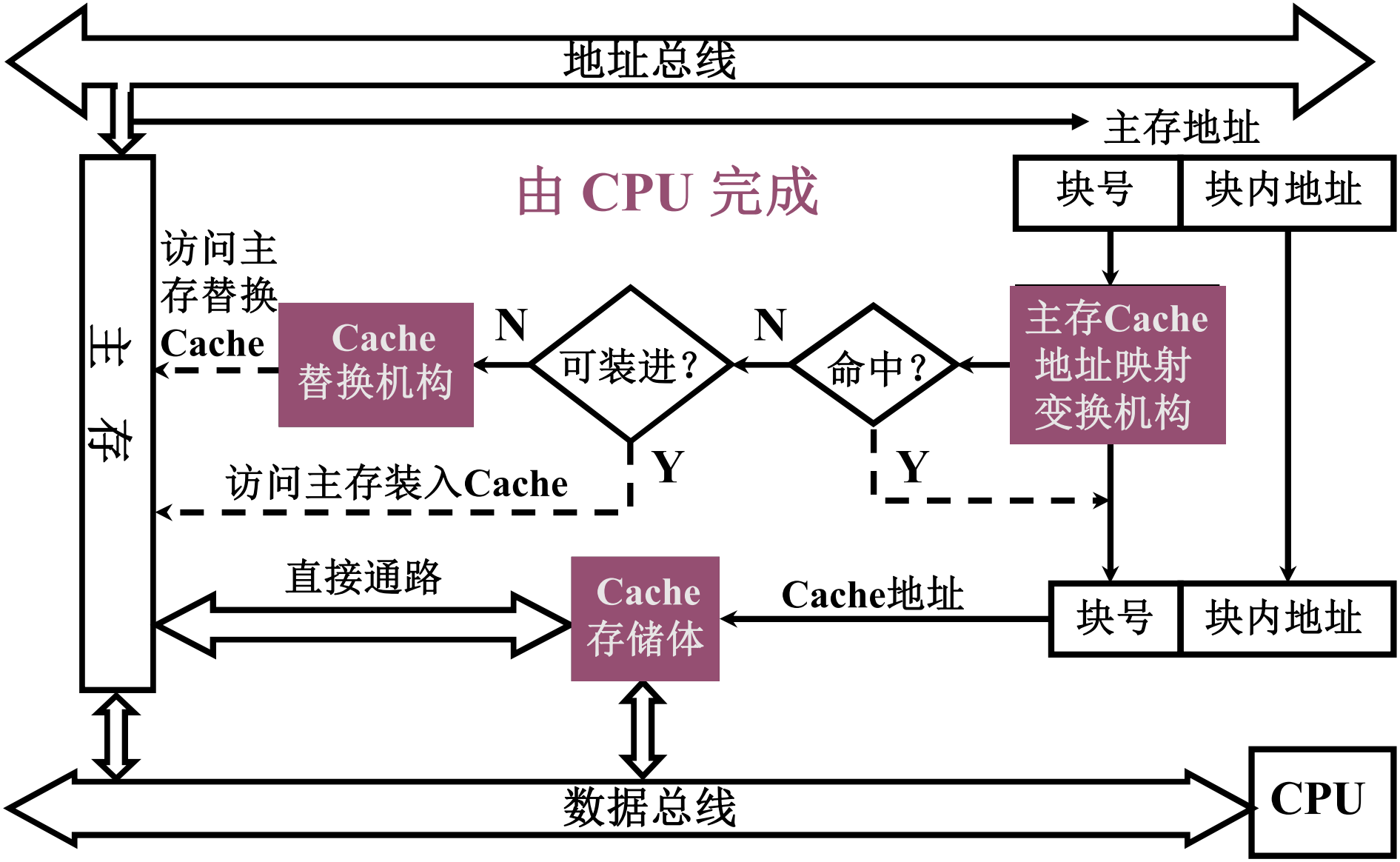
3. 当发生失效时，应替换哪一块?

   (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?

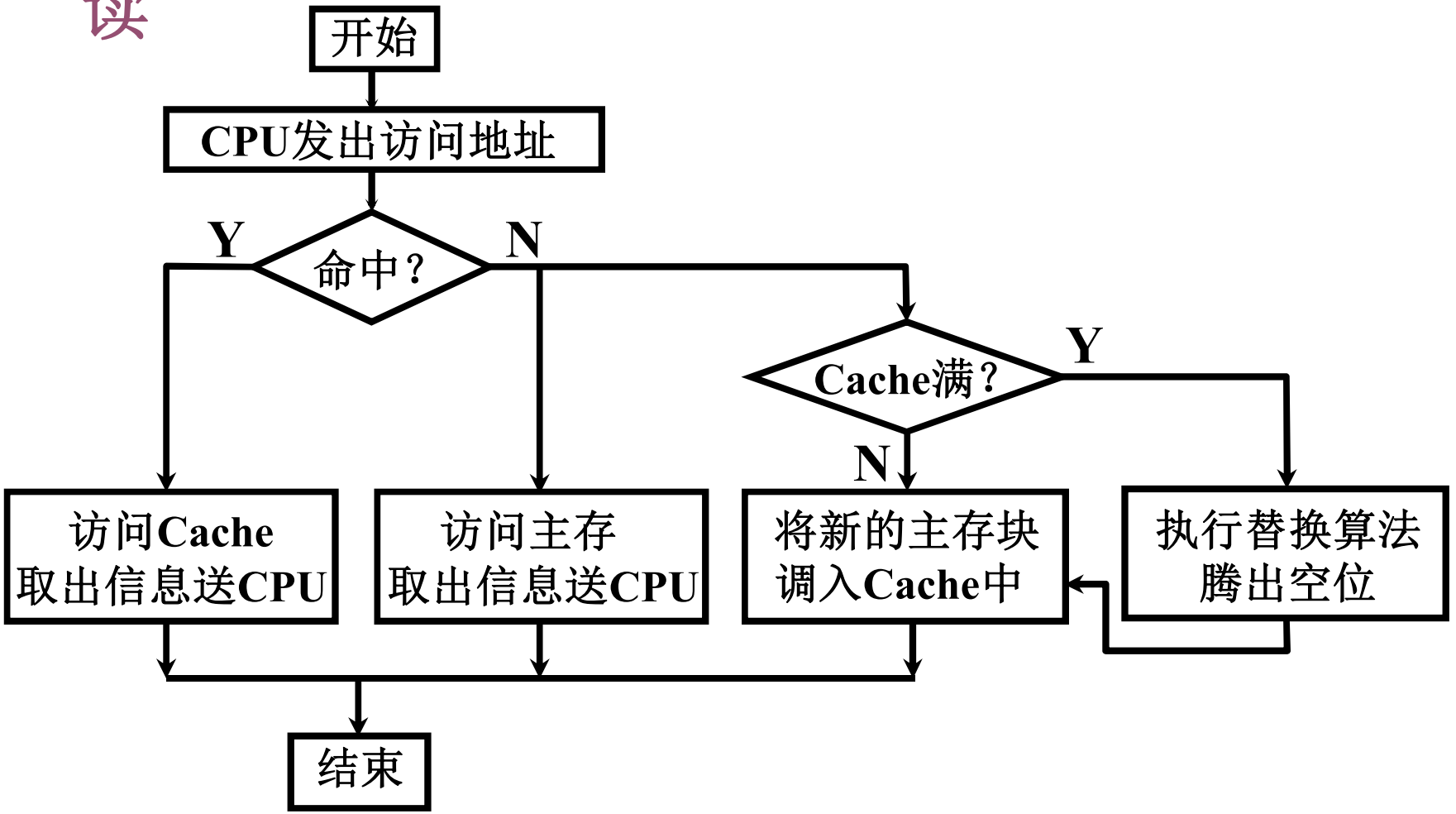
   (写策略　如何处理写操作)

4. Cache 的基本结构



4. Cache 的 读写 操作

读



写　　Cache 和主存的一致性

5. Cache 的改进

(1) 增加 Cache 的级数

片载（片内）Cache

片外 Cache

(2) 统一缓存和分开缓存

指令 Cache　　数据 Cache

与主存结构有关

与指令执行的控制方式有关　　是否流水

Pentium	8K 指令 Cache	8K 数据 Cache
PowerPC620	32K 指令 Cache	32K 数据 Cache

二、Cache－主存的地址映射 4.3

1. 直接映射

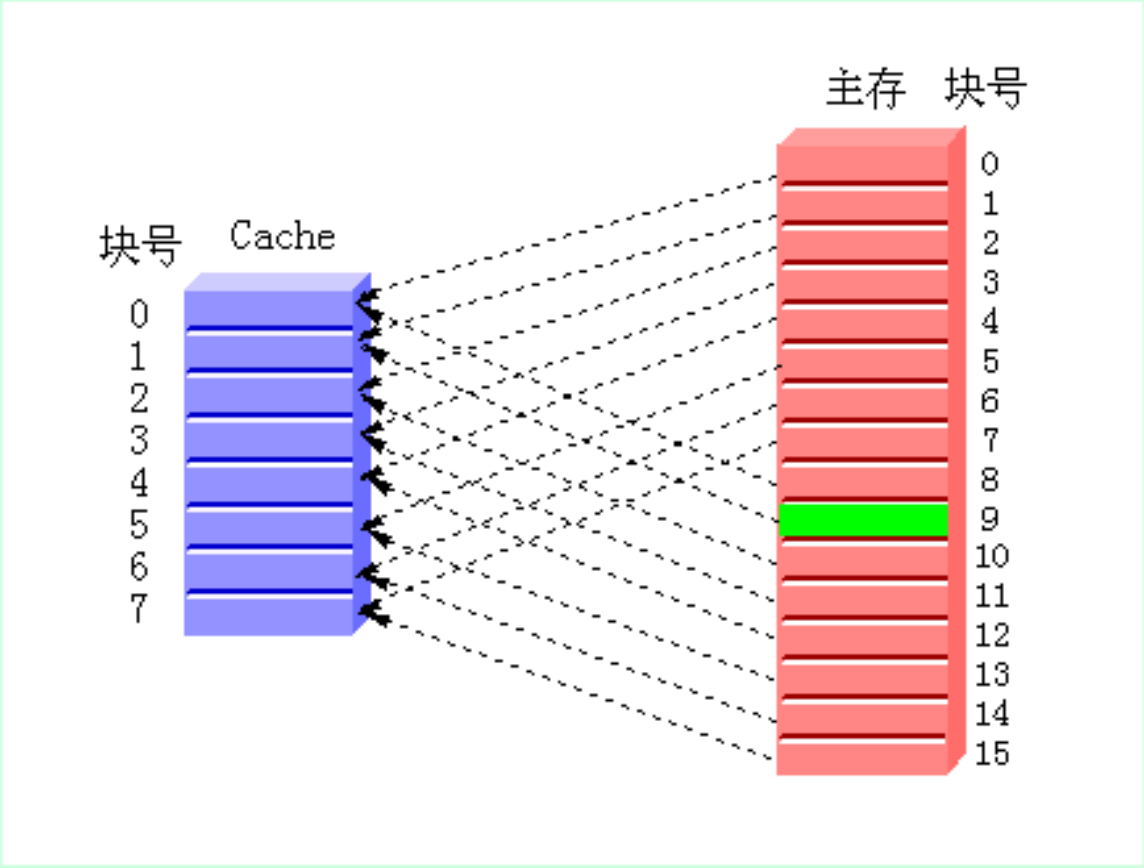
直接映射：主存中的每一块只能被放置到Cache中唯一的一个位置。（循环分配）

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。

直 接 映 射

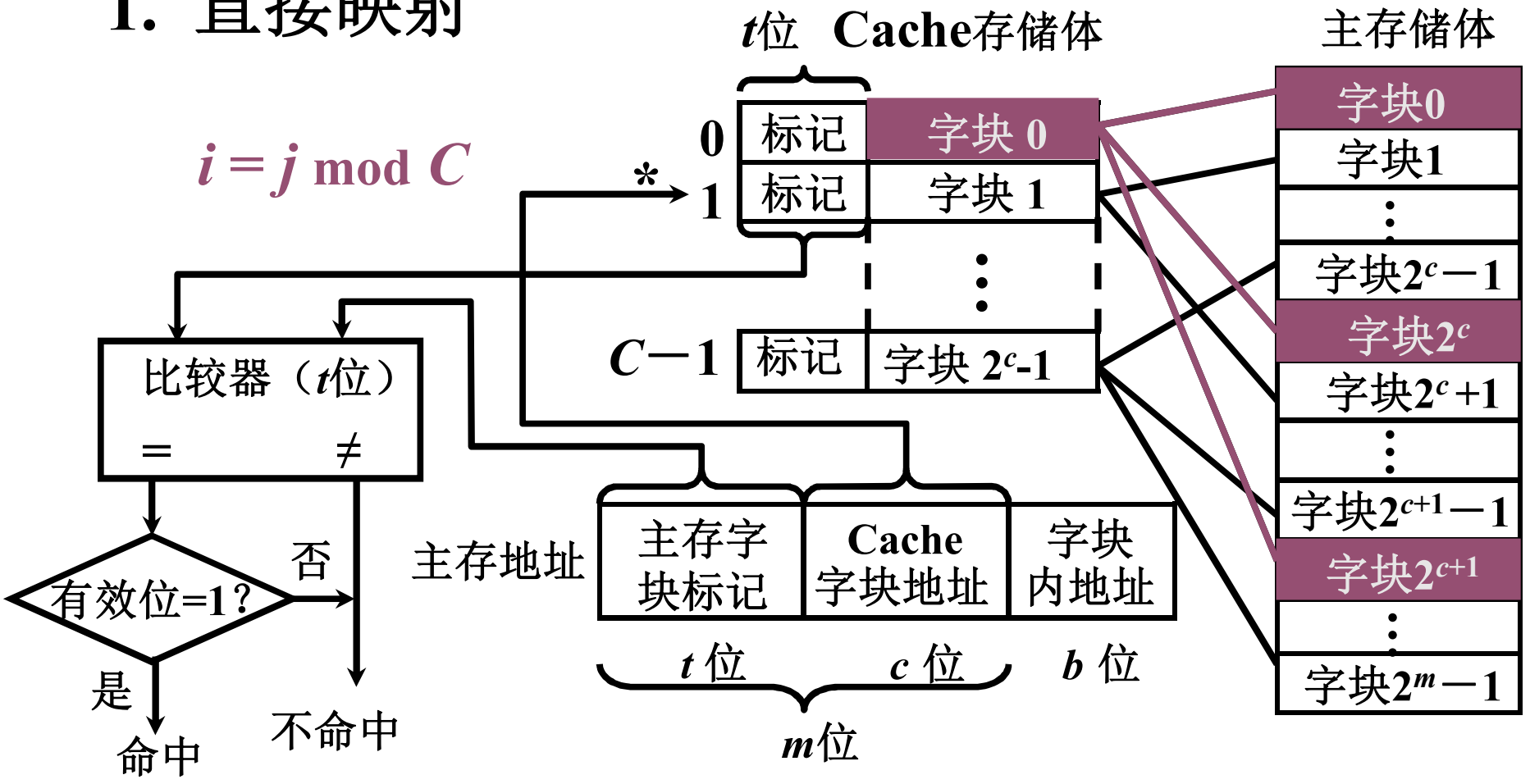
（举 例）



二、Cache—主存的地址映射

4.3

1. 直接映射



每个缓存块 i 可以和 若干 个 主存块 对应

每个主存块 j 只能和 一 个 缓存块 对应

2. 全相联映射

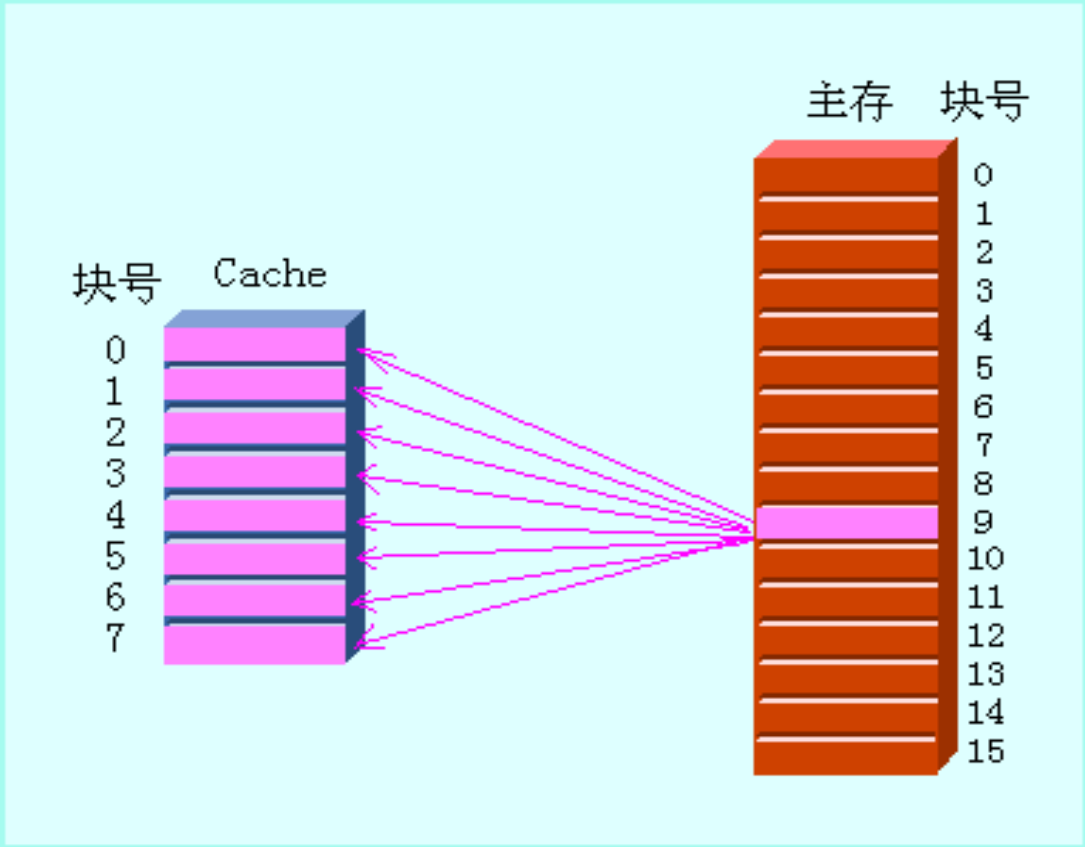
全相联：主存中的任一块可以被放置到Cache中的任意一个位置。

对比：　阅览室位置--随便坐

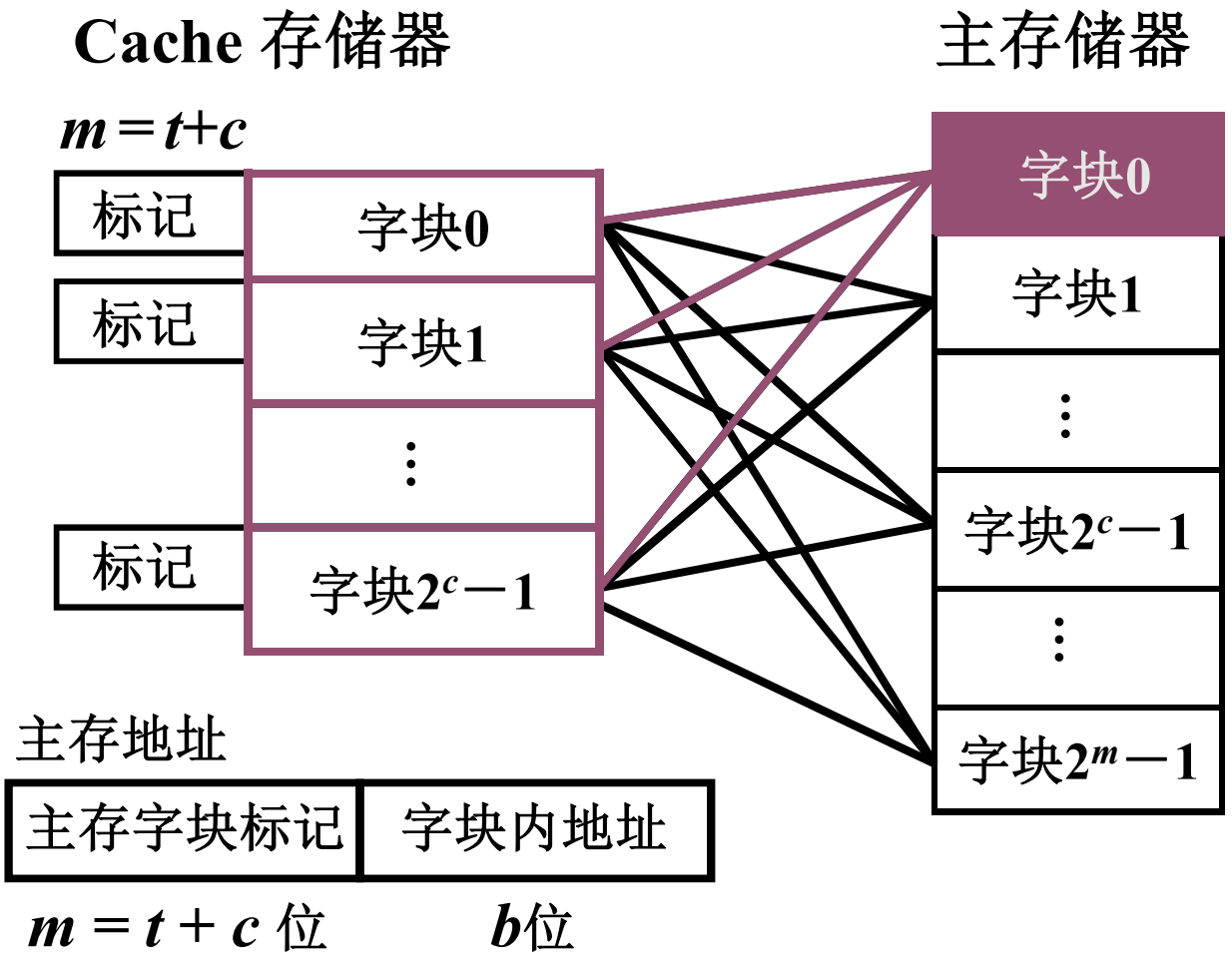
特点：　空间利用率最高，冲突概率最低，实现最复杂。

全相联映射

（举例）



2. 全相联映射

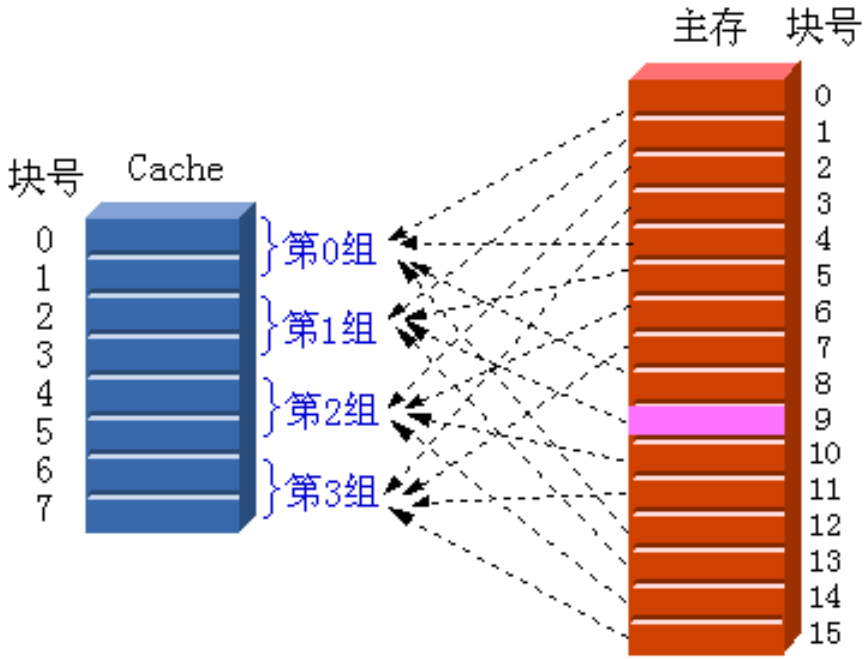


主存 中的 任一块 可以映射到 缓存 中的 任一块

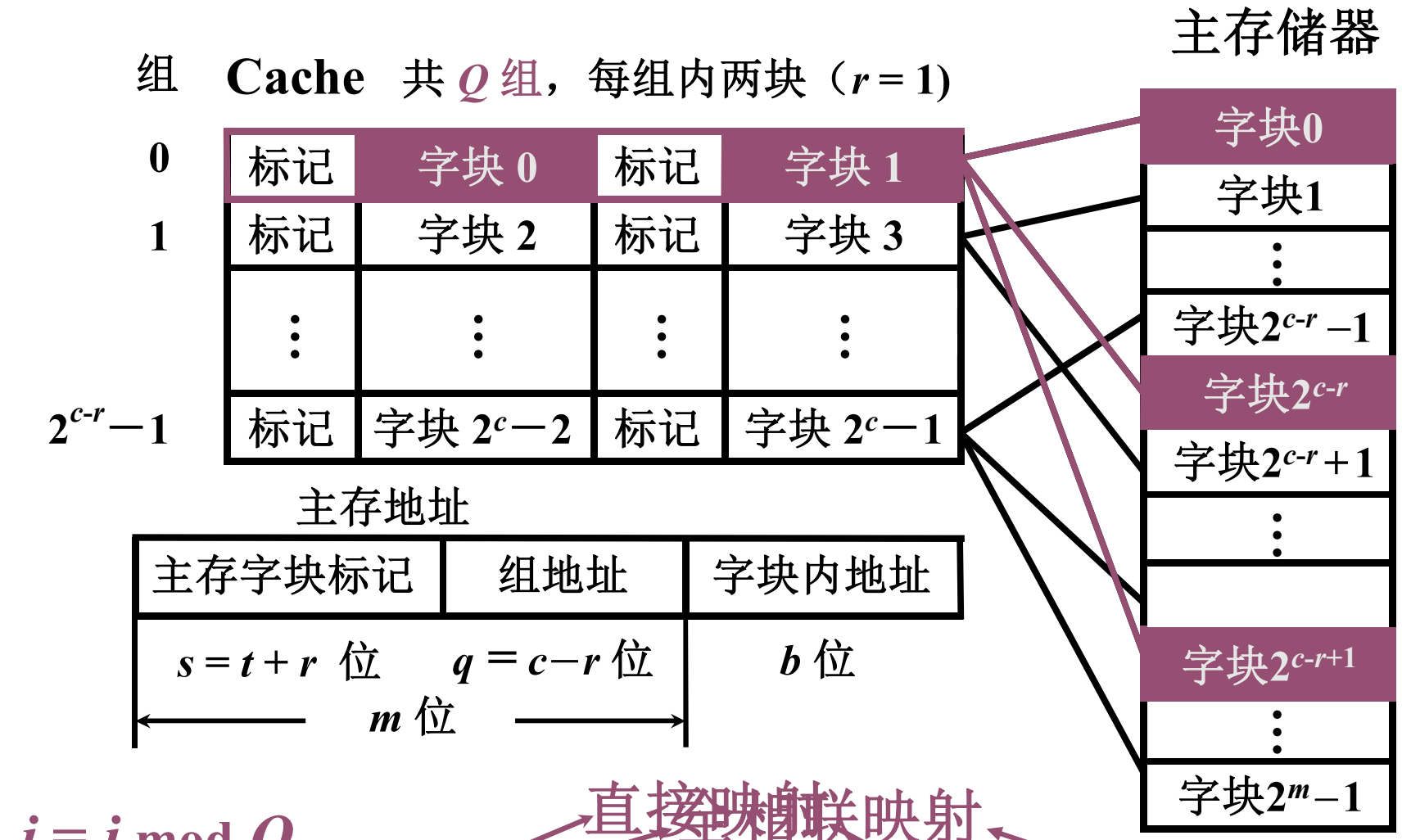
3. 组相联映射

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。

组相联是直接映射和全相联的一种折中



4. 组相联映射



$i = j \bmod Q$

直接映射 全相联映射

某一主存块 j 按模 Q 映射到 缓存 的第 i 组中的 任一块

- ***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
直接映射	1	***M***
组相联	$1<n<M$	$1<G<M$

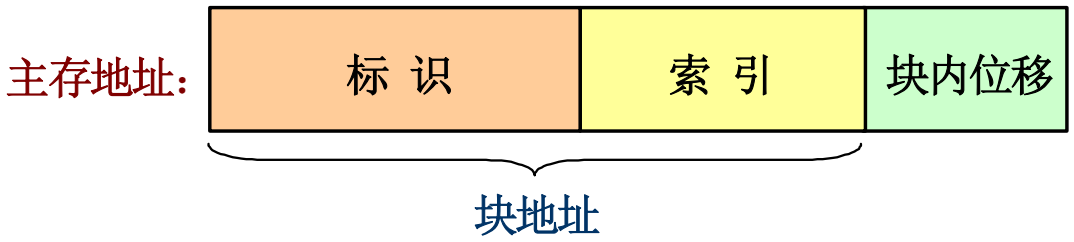
- 大多数计算机的Cache: $n \leq 4$

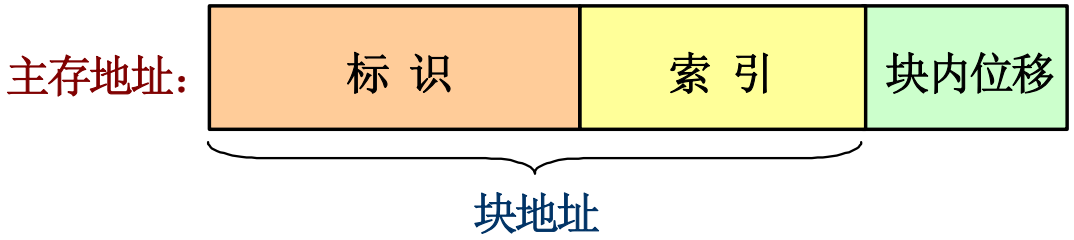
想一想：相联度是否越大越好？

四、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

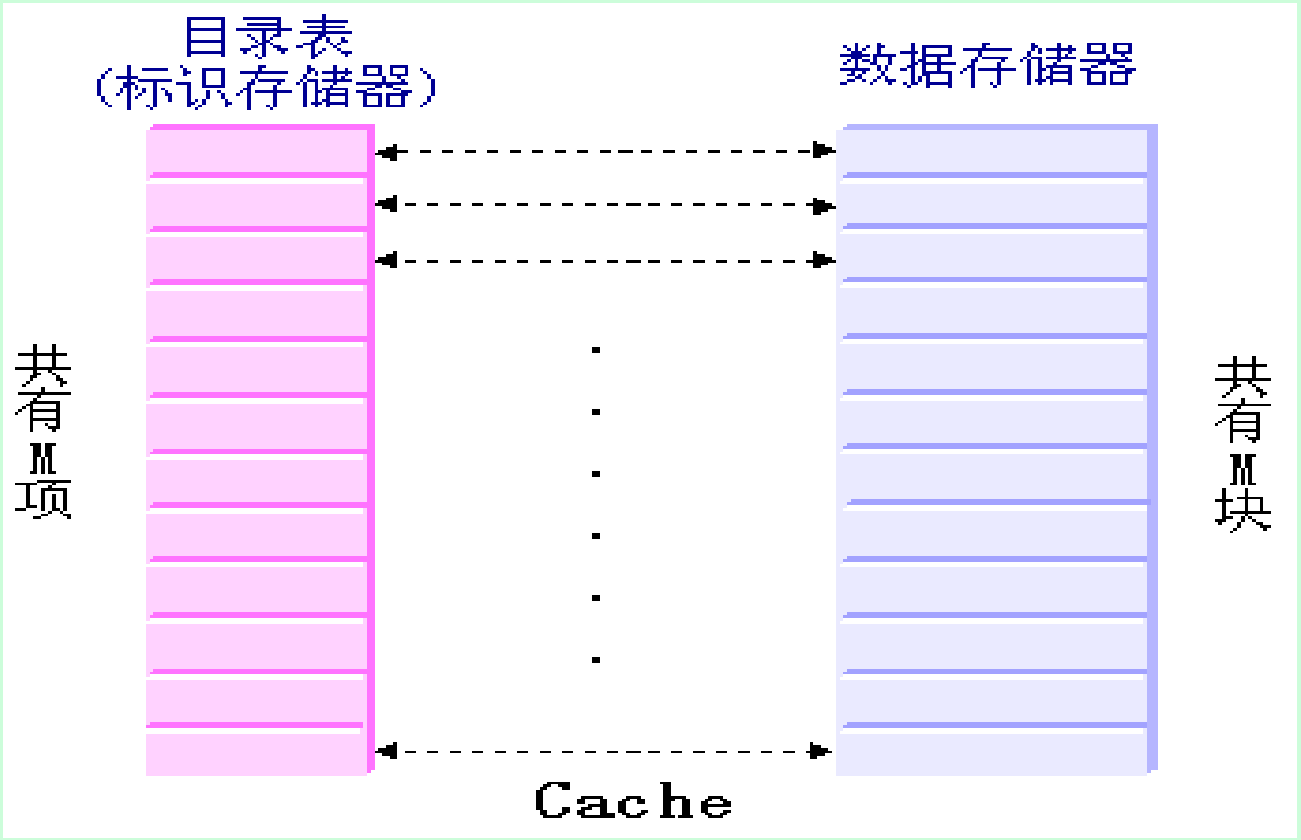




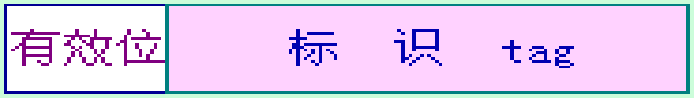
- 块内地址偏移用来从块中选出需要的数据
- 索引域用来选择组
- 通过比较标识域来判断是否发生命中

- 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

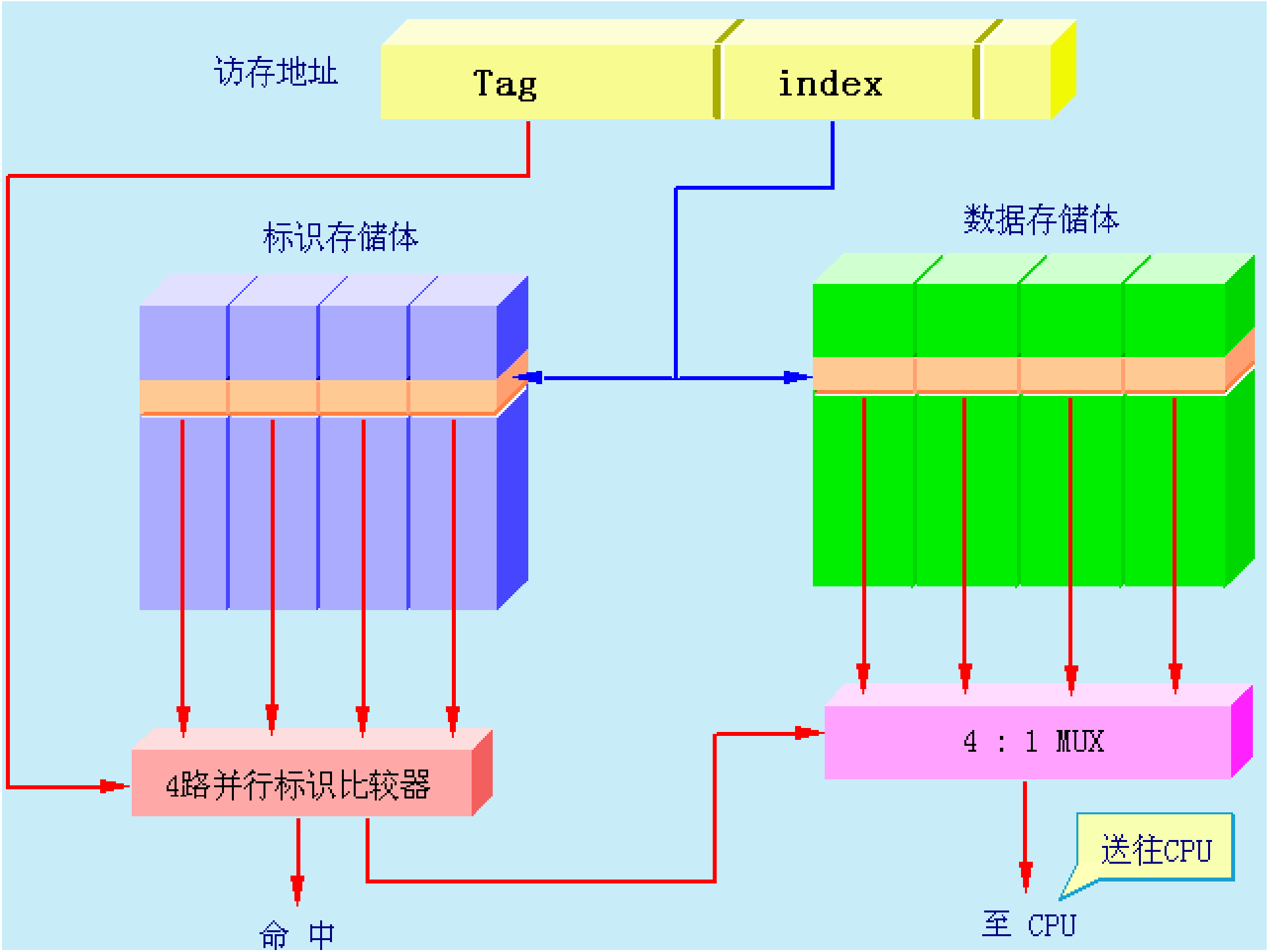


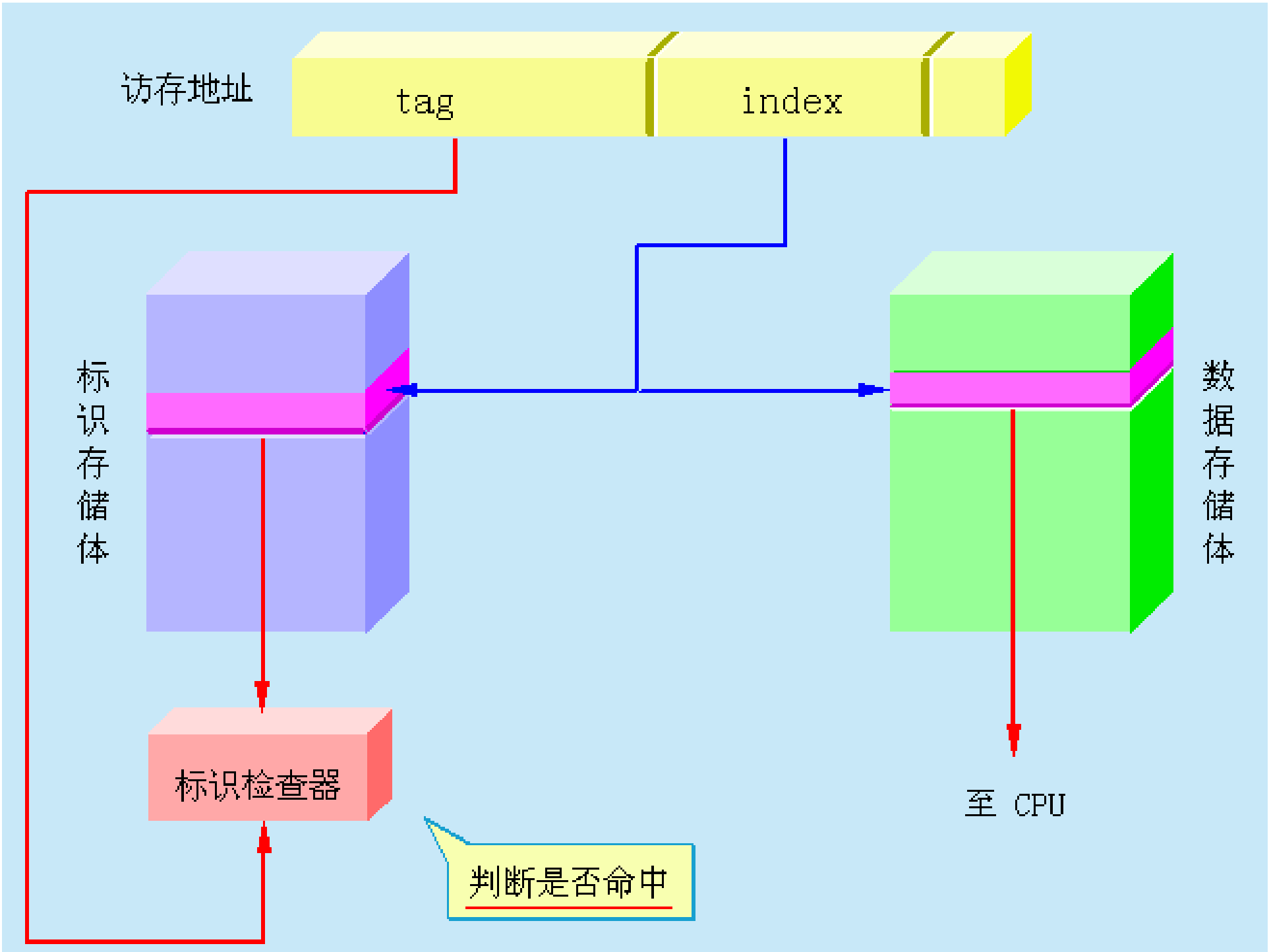
目录表项：



访存地址：

- 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

二、地址映射

小结

成本高　不灵活

直接　某一 主存块 只能固定 映射到 某一 缓存块

全相联　某一 主存块 能 映射到 任一 缓存块

组相联　某一 主存块 能 映射到 某一 缓存 组 中的 任一块