

计算机组成原理与系统结构

第五章 存储体系

http://www.icourses.cn/coursestatic/course_2859.html



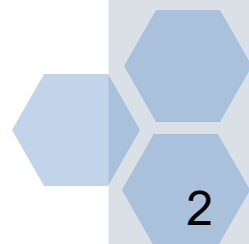


2、全相联映射

❖ **特点：**是多对多的映射关系：对于主存的任何一块均可以映射到 Cache 的任何一行。

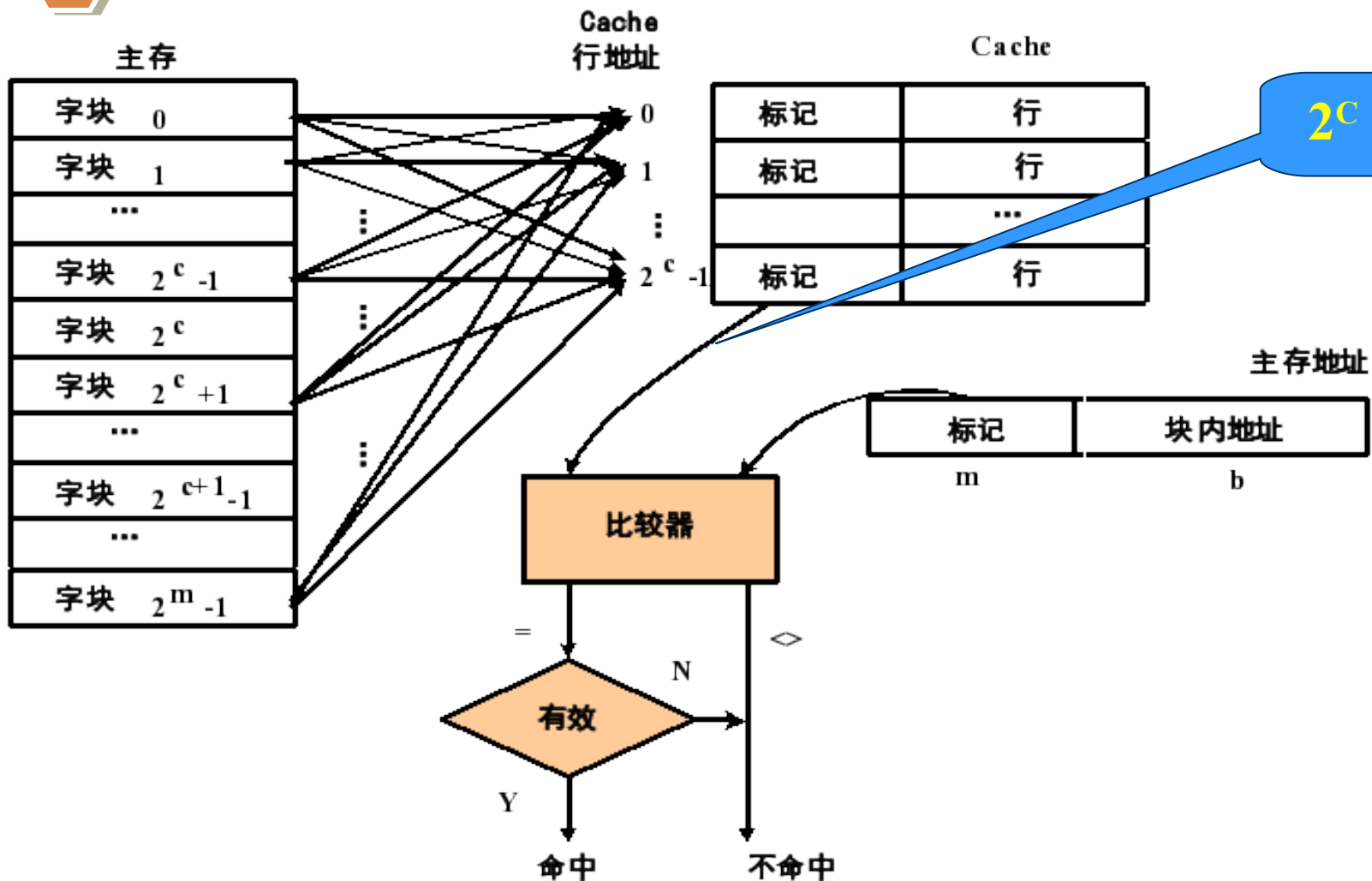
① **优点：**机制灵活，命中率高。

② **缺点：**比较器电路难于设计和实现，因此只适合于小容量的 Cache 。





2、全相联映射





全相联映象法举例

❖ 例如：主存 4GB（按字节编址），Cache 512KB，块大小 16B；读主存 234589ABH 的过程（假设 Cache 空白）。

主存划分格式

主存地址： 32 位

块大小 $16\text{B}=2^4\text{B}$ ，按字节编址，所以块内地址：

| 标记 | 块内地址 |
|---------------------------------------|------|
| 28 位 | 4 位 |
| 0010 0011 0100 0101 1000 1001 1010 | 1011 |





3、组相联映射

❖ **特点：**将 Cache 的行分成 2^{c-r} 组，每组 2^r 行。主存的字块存放到 Cache 中的哪个组是固定的，至于映射到该组哪一行是灵活的，即有如下函数关系：

$$j = (i \bmod 2^{c-r}) \times 2^r + k$$

$$\text{其中 } 0 \leq k \leq 2^{r-1}$$

- ① 优点：大大增加了映射的灵活性，主存中一块可映射到 Cache 的 2^r 块，提高了命中率。每次比较只是进行 2^r 路比较， r 较小时，硬件开销不是很大。
- ② 组相联映像通常采用 2 路、4 路和 8 路比较，即取 $r=1, r=2, r=3$ 。



3、组相联映射





组相联映象法举例

❖ 例如：主存 4GB（按字节编址），Cache 512KB，块大小 16B；4 路组相联，读主存 234589ABH 的过程（假设 Cache 空白）。

主存划分格式

主存地址：__ 32 __ 位

块大小 $16B = 2^4B$ ，按字节编址，故块内地址：__ 4 __ 位

Cache 分为：__ $512KB \div 16B = 2^{15}$ __ 行（块）

4 路组相联，即 4 块 / 组，Cache 分为： $\underline{2^{15} \div 4 = 2^{13}}$
组

| 标记 | Cache 组地址 | 块内地址 |
|-----------------------|---------------------|------|
| 15 位 | 13 位 | 4 位 |
| 0010 0011 0100 010 | 1 1000 1001 1010 | 1011 |

$-13-4=$





三、替换算法

1. 随机替换算法

2. 先进先出算法（FIFO）

3. 最近最少使用算法（LRU）

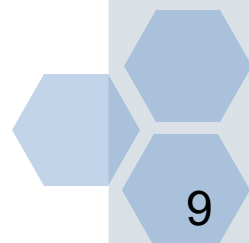
- 该算法统计哪一个 Cache 行是近段时间使用次数最少的 Cache 行，需替换时就将它替换出去。
- LRU 替换算法可以通过为每个 Cache 行设置一个计数器来实现 LRU 替换算法，Cache 每命中一次，命中行的计数器被清零，其他行的计数器加 1，需要替换的话，就将计数器值最大的行替换出去。



三、替换算法

| | | | | | | | | |
|-------|---|--|--|--|--|--|---|---|
| 访问顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 地址块号 | 2 | 11 | 2 | 9 | 7 | 6 | 4 | 3 |
| 块分配情况 | <div><div>2</div><div>-</div><div>-</div><div>-</div></div> | <div><div>2</div><div>11</div><div>-</div><div>-</div></div> | <div><div>2</div><div>11</div><div>-</div><div>-</div></div> | <div><div>2</div><div>11</div><div>9</div><div>-</div></div> | <div><div>2</div><div>11</div><div>9</div><div>7</div></div> | <div><div>6</div><div>11</div><div>9</div><div>7</div></div> | <div><div>6</div><div>4</div><div>9</div><div>7</div></div> | <div><div>6</div><div>4</div><div>3</div><div>7</div></div> |
| 操作状态 | 调进 | 调进 | 命中 | 调进 | 调进 | 替换 | 替换 | 替换 |

先进先出替换方式下的 cache 内容变化情况

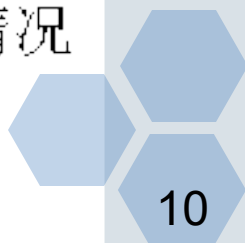




三、替换算法

| | | | | | | | | |
|-------|----|----|-----|-----|-----|----|----|----|
| 访问顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 地址块号 | 2 | 11 | 2 | 9 | 7 | 6 | 4 | 3 |
| 块分配情况 | 2* | 2* | 2 | 2 | 2 | 2* | 4 | 4 |
| | — | 11 | 11* | 11* | 11* | 6 | 6 | 6 |
| | — | — | — | 9 | 9 | 9 | 9* | 3 |
| | — | — | — | — | 7 | 7 | 7 | 7 |
| 操作状态 | 调进 | 调进 | 命中 | 调进 | 调进 | 替换 | 替换 | 替换 |

最久未使用替换方式下的 cache 内容变化情况





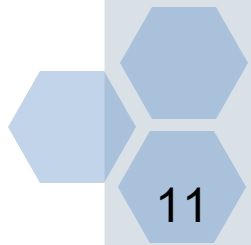
四、写策略

常用的写策略通常有写贯穿和写回两种

1. 写贯穿策略

当 CPU 写 Cache 命中时，所有写操作既对 Cache 也对主存进行；当 CPU 写 Cache 不命中时，直接写主存，有两种做法：

- 其一，不将该数据所在的块拷贝到 Cache 行，称为 **WTNWA** 法；
- 其二，将该数据所在块拷贝到 Cache 的某行，称为 **WTWA** 法。





四、写策略

2. 写回策略 (Write Back)

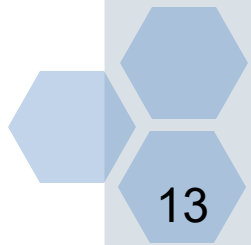
- 当 CPU 写 Cache 命中时，写操作只是对 Cache 进行，而不修改主存的相应内容，仅当此 Cache 行被换出时，相应的主存内容才被修改；当 CPU 写 Cache 不命中时，先将该数据所在块拷贝到 Cache 的某行，余下操作与 Cache 写命中时相同。
- 为了区别 Cache 行是否被改写过，应为每个 Cache 行设置一个修改位，CPU 修改 Cache 行时，标记其修改位，当此 Cache 行被换出时，判别此 Cache 行的修改位，从而决定是否将 Cache 行数据写回主存相应单元。



四、写策略

3. 两种写策略比较

- 写贯穿策略保证了主存数据总是有效，写回策略可能导致 Cache 和主存数据不一致；
- 写回策略的效率高于写贯穿策略；
- 写回策略的控制比写贯穿策略的控制复杂。

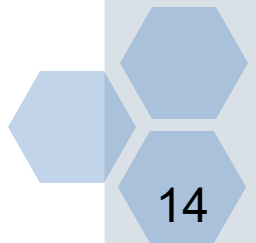




五、Cache 的多层次设计

设计 Cache 主要考虑五个问题：

- 第一，容量。
- 第二，Cache 中行的大小。
- 第三，Cache 的组织（地址映射方式）。
- 第四，指令和数据共用同一个 Cache 还是分享不同 Cache。
- 第五，Cache 的层次。





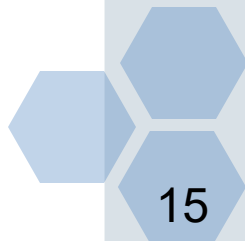
五、Cache 的多层次设计

1. 统一 Cache 和分离 Cache

- 统一 Cache：只有一个 Cache，指令和数据混放。
- 分离 Cache：分为指令 Cache 和数据 Cache。它消除了流水线中指令处理器和执行单元间的竞争，因此，特别适用于 Pentium II 和 Power PC 这样的超标量流水线中；是 Cache 结构发展的趋势。

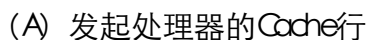
2. 单级 Cache 与两级 Cache

- 一级 Cache 和二级 Cache
- 采用两级 Cache 结构可以提高性能





3. Cache 一致性问题



RH——读命中；
RMS——读不命中，
共享； RME——读
不命中，专有； WH
——写命中； WM
——写不命中；
SHR——读监听命中
； SHW——写监听
命中或读是用于修
改； +——无效处
理； ↑——Cache
行填入； ↓——无
效行拷回； ×——
读用于修改

MESI 协议的状态图

