4.3 高速缓冲存储器

- 一、概述
  - 1、为什么用Cache
  - 2、Cache的工作原理
    - 主存和缓存的编址
    - 命中与未命中
    - Cache的命中率
    - Cache-主存系统的效率
  - 3、Cache的基本结构
  - 4、Cache的读写操作
  - 5、Cache的改进
- 二、Cache-主存的地址映射
- 三、替换算法

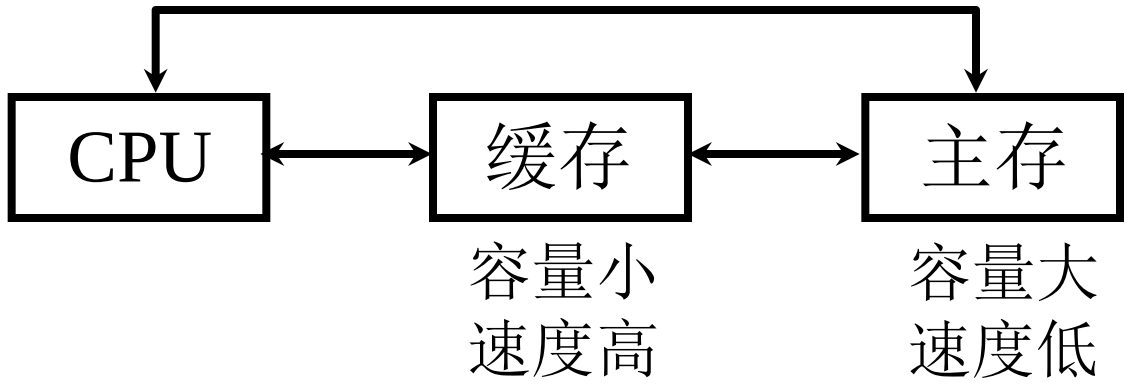
4.3 高速缓冲存储器

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

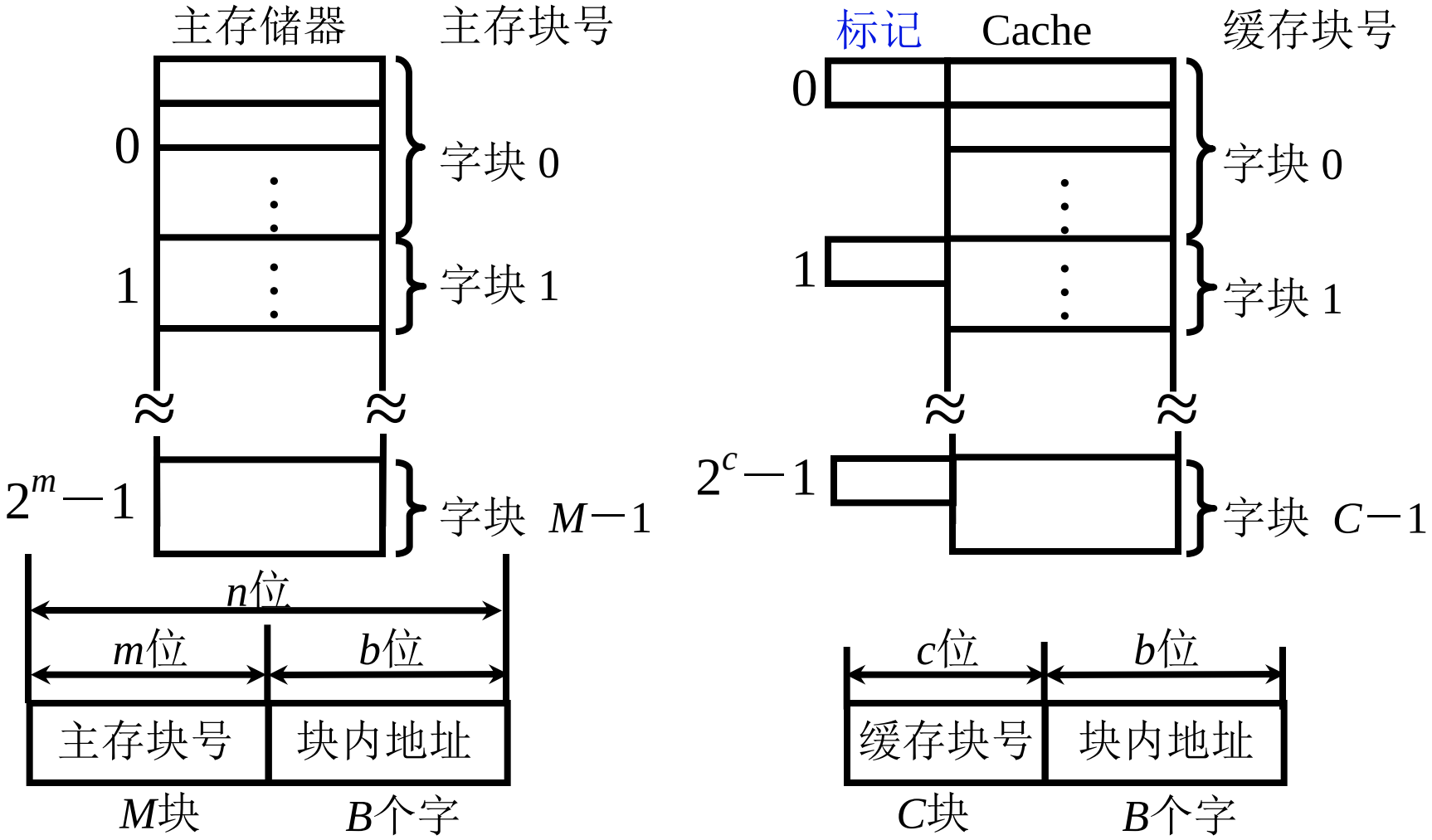


程序访问的局部性原理

2. Cache 的工作原理

4.3

(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 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 ~ 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）

(4) Cache –主存系统的效率

效率 e 与 命中率 有关

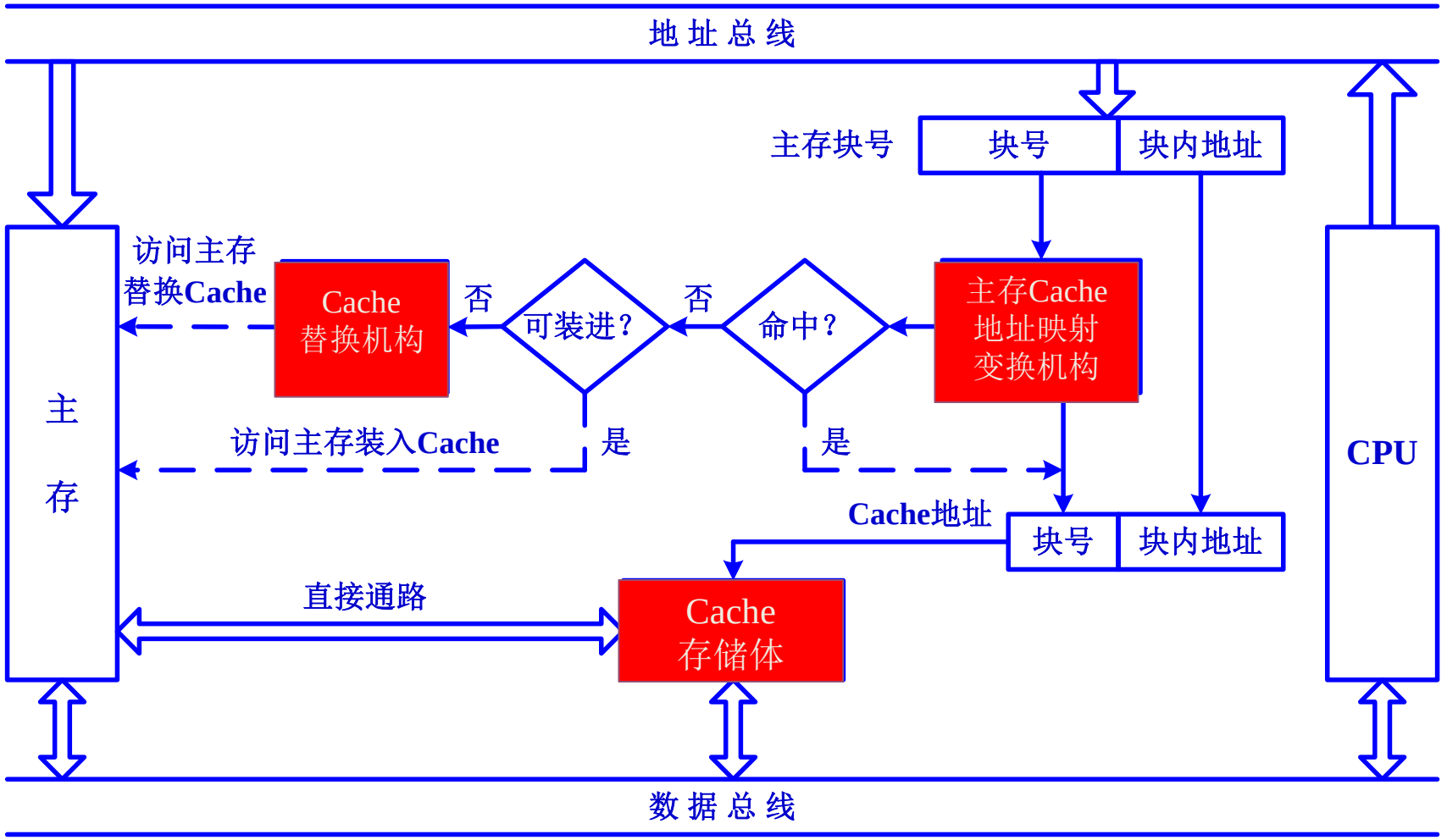
$$e = \frac{\text{访问 Cache 的时间}}{\text{平均访问时间}} \times 100\%$$

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

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

3. Cache 的基本结构

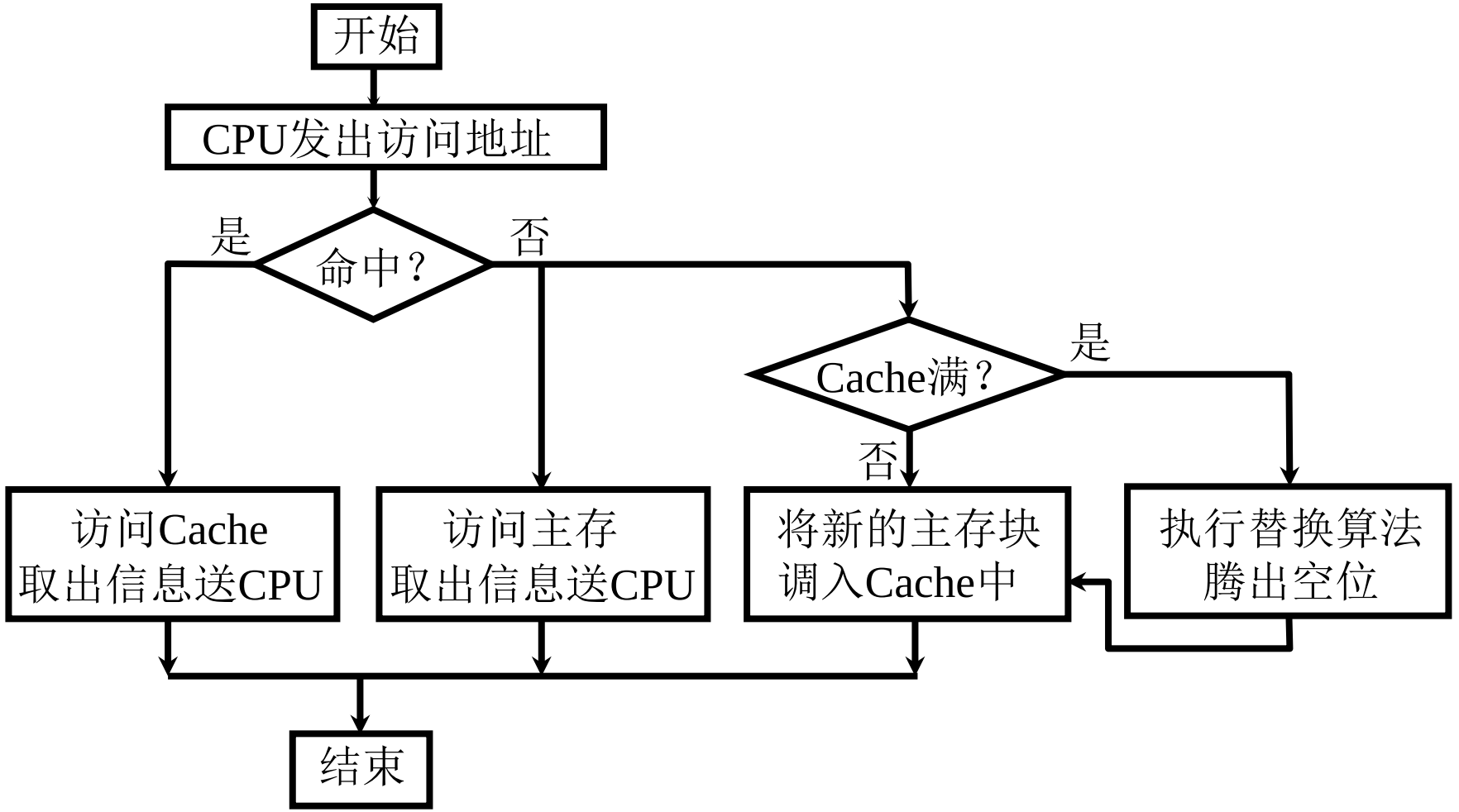
4.3



4. Cache 的 读写 操作

4.3

读



4. Cache 的 读写 操作

4.3

写　Cache 和主存的一致性

- 写直达法（Write – through）

  写操作时数据既写入Cache又写入主存

  写操作时间就是访问主存的时间，Cache块退出时，不需要对主存执行写操作，更新策略比较容易实现

- 写回法（Write – back）

  写操作时只把数据写入 Cache 而不写入主存

  当 Cache 数据被替换出去时才写回主存

  写操作时间就是访问 Cache 的时间，

  Cache块退出时，被替换的块需写回主存，增加了 Cache 的复杂性

5. Cache 的改进

(1) 增加 Cache 的级数

片载（片内）Cache

片外 Cache

(2) 统一缓存和分立缓存

指令 Cache　　数据 Cache

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

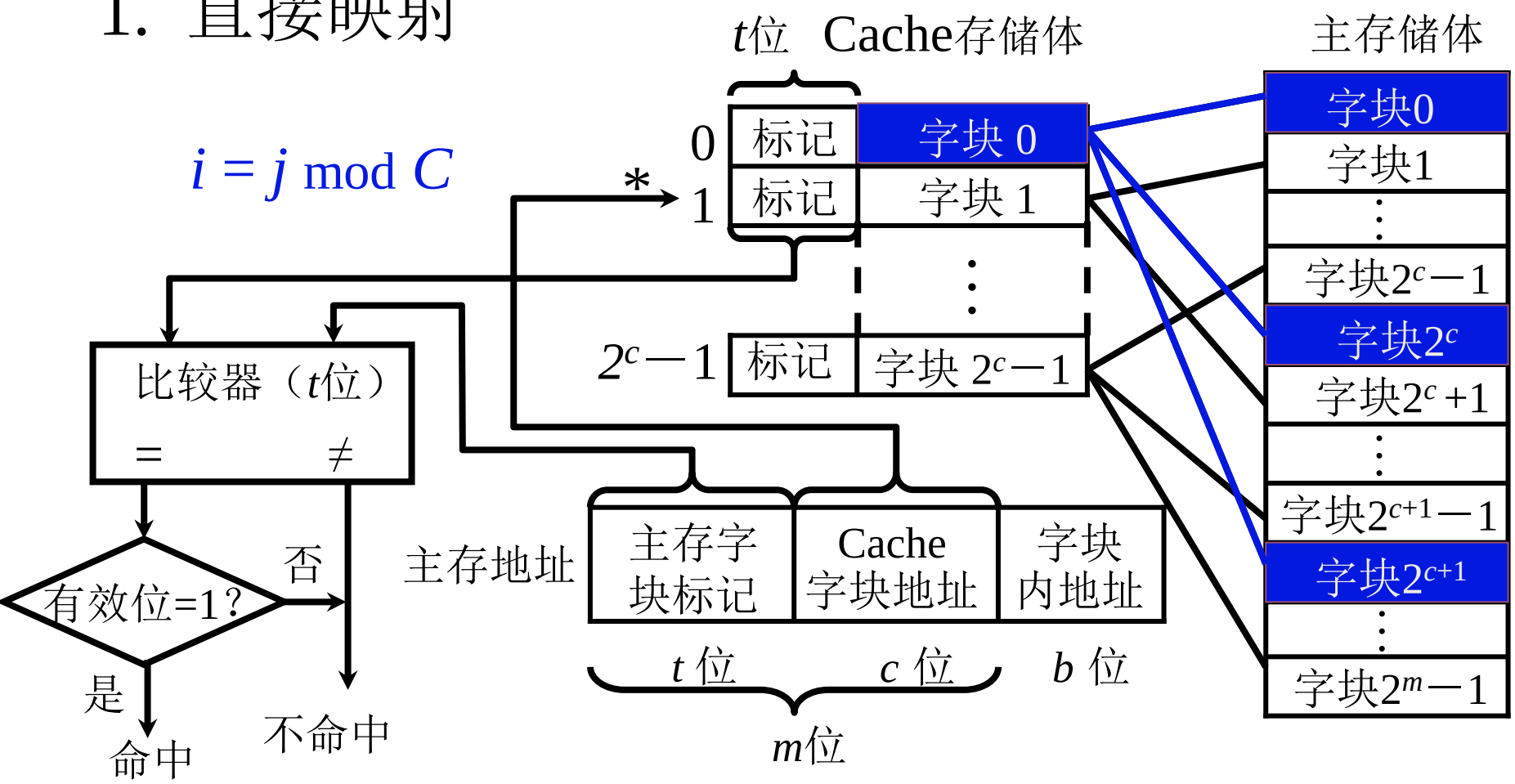
4.3 高速缓冲存储器

- 一、概述
- 二、Cache-主存的地址映射
  - 1、直接映射
  - 2、全相联映射
  - 3、组相联映射
- 三、替换算法

二、Cache – 主存的地址映射

4.3

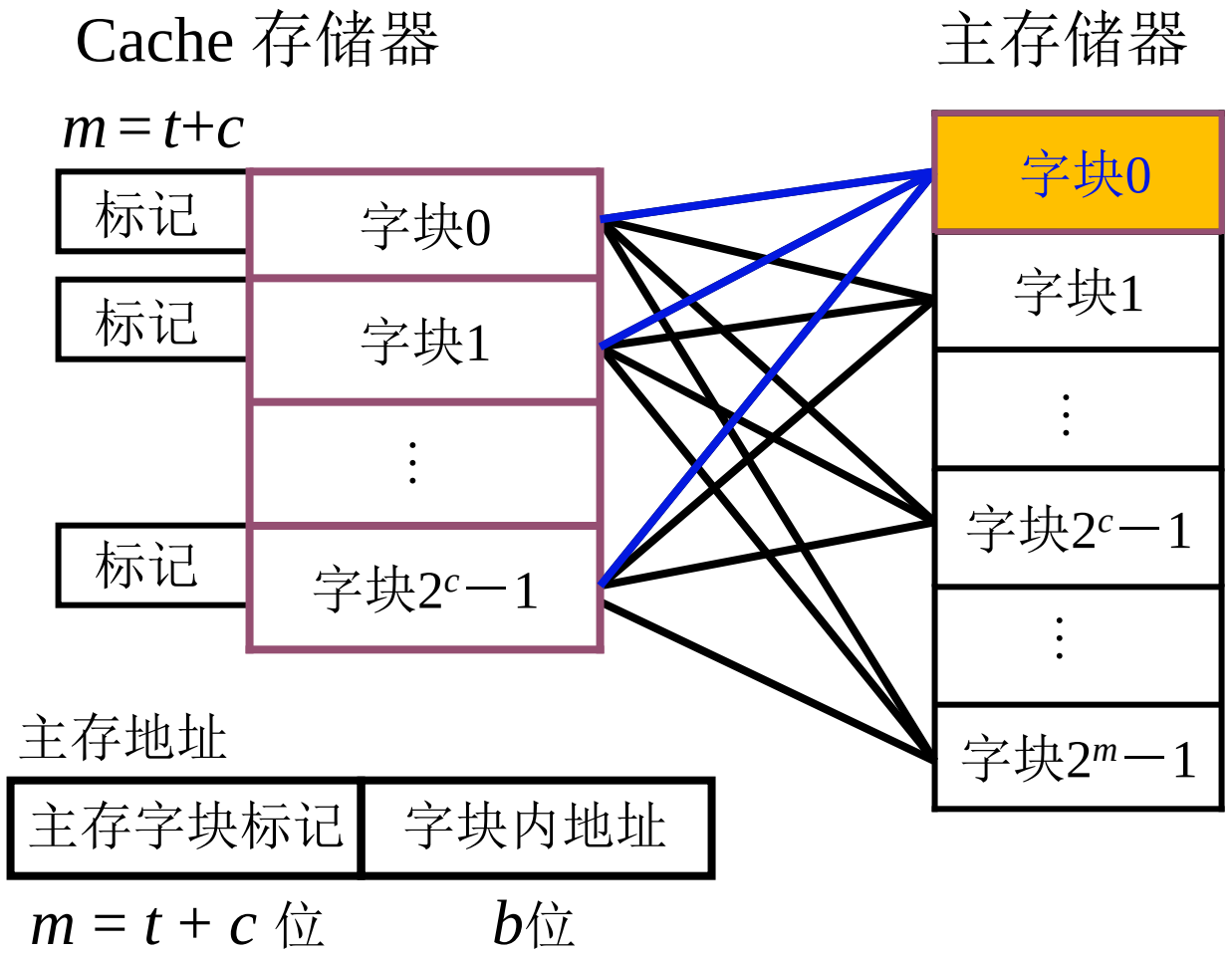
1. 直接映射



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

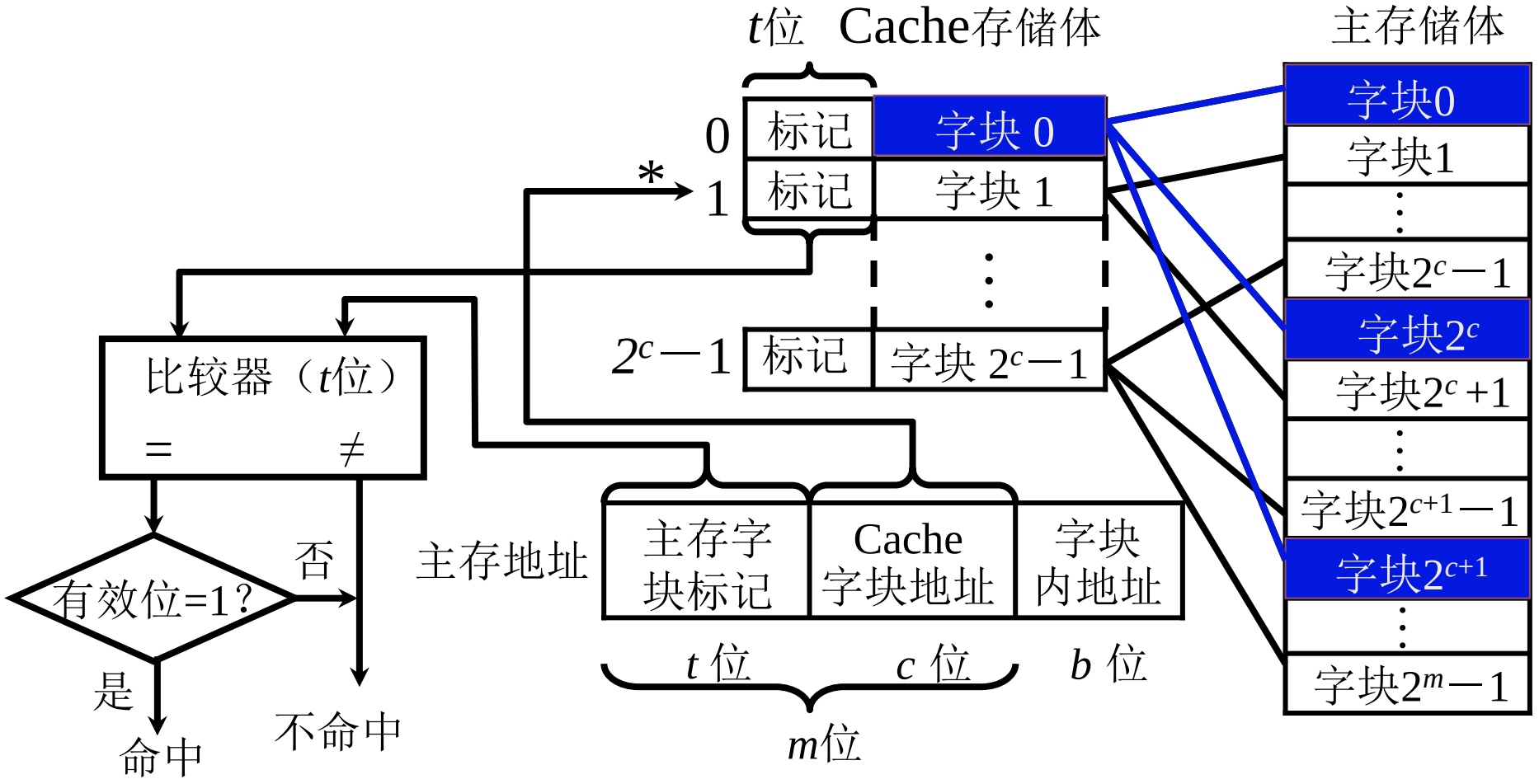


主存 中的 任一块 可以映射到 缓存 中的 任一块

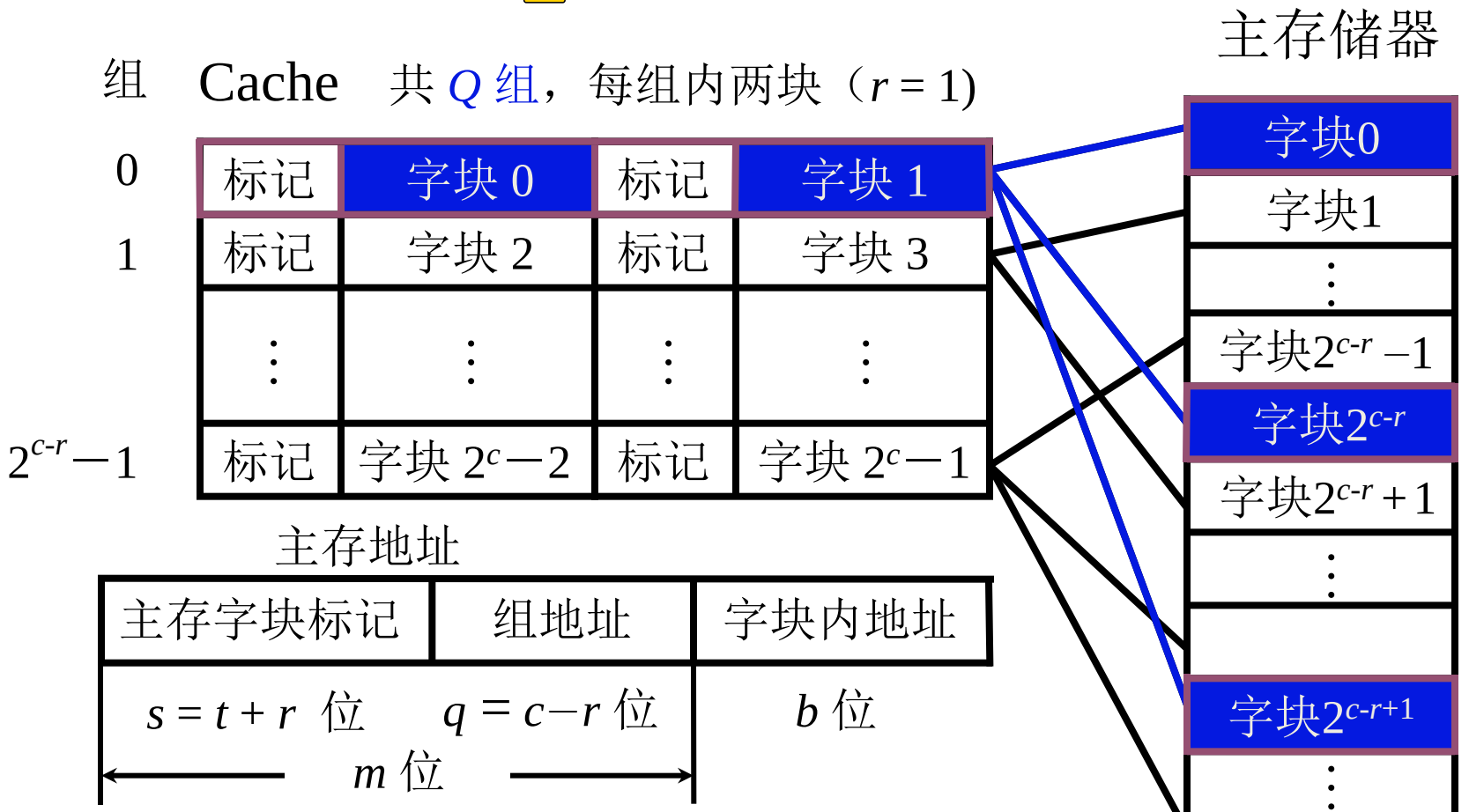
二、Cache – 主存的地址映射

4.3

刚刚讲过的直接相联映射



3. 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

三、替换算法

4.3

1. 先进先出（FIFO）算法

2. 近期最少使用（LRU）算法

小结

成本高

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 只能 映射到 某一 缓存 组 中的 任一块