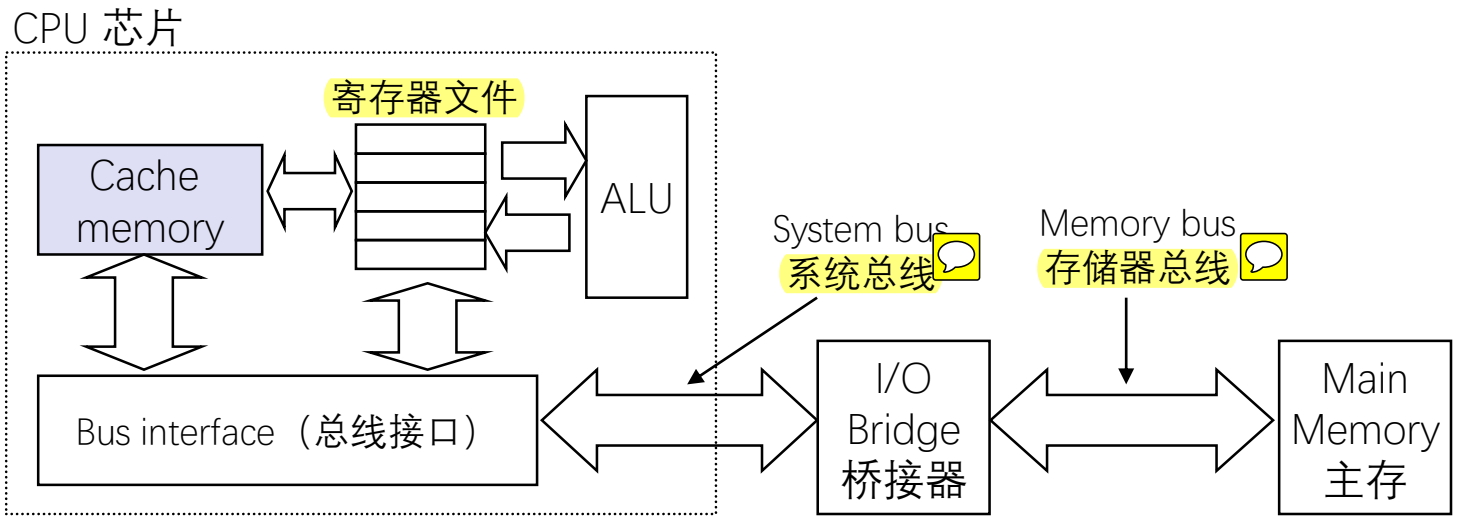
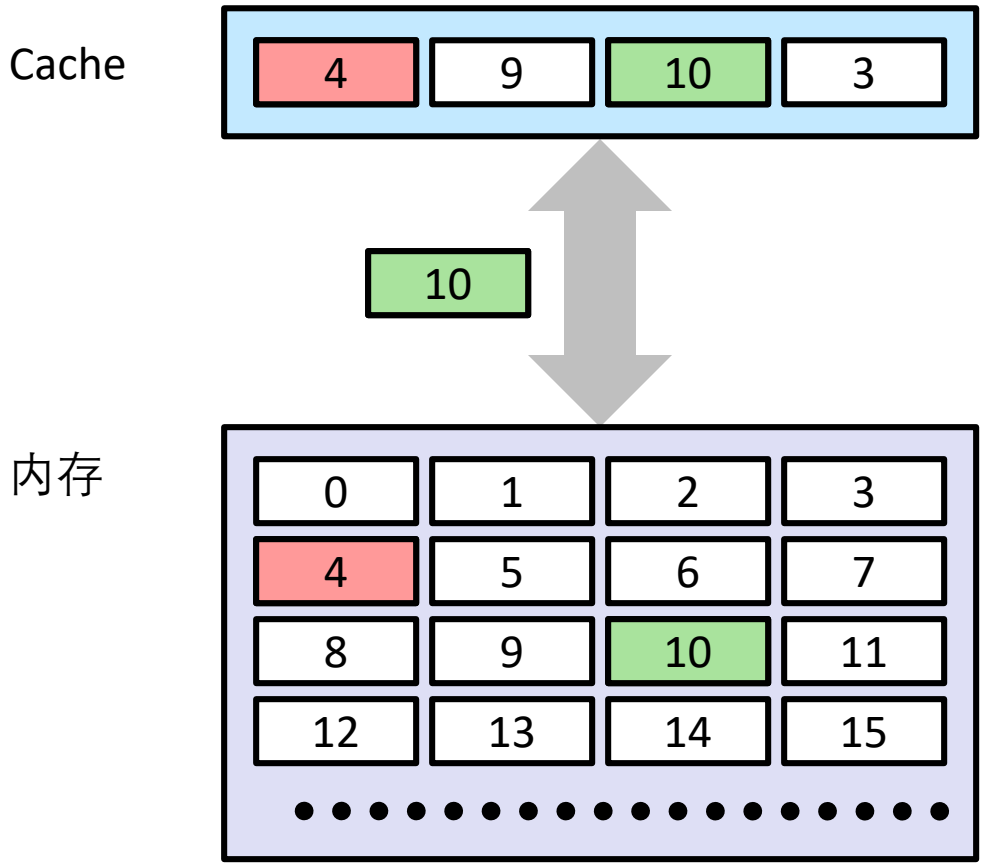
高速缓存(cache) 概念和原理

cache基本思想

- cache 存储器（Cache memories）
  - 在处理器附近增加一个小容量快速存储器(cache)
  - 基于SRAM，由硬件自动管理
- cache基本思想:
  - 频繁访问的数据块存储在cache中
  - CPU 首先在cache中查找想访问的数据，而不是直接访问主存
  - 我们希望被访问数据存放在cache中



Cache 基本概念：块（block）

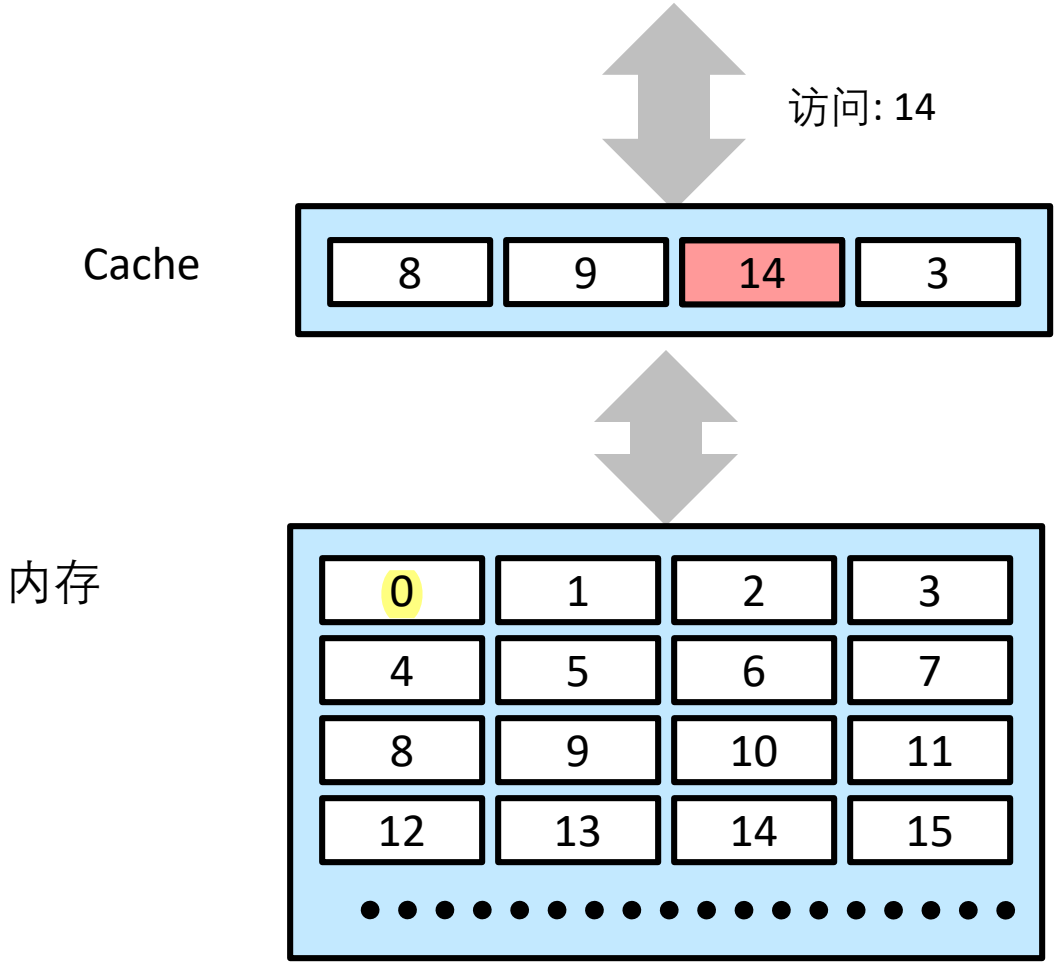


更小、更快、更贵
cache 是内存以块为元素的一个子集

数据以块（block）为传输单位，
块一般为几个字，例如8个字。
块为什么包含多个字？局部性原理

更大、更慢、更便宜
内存可看作许多的块的集合

Cache 基本概念:命中 Hit

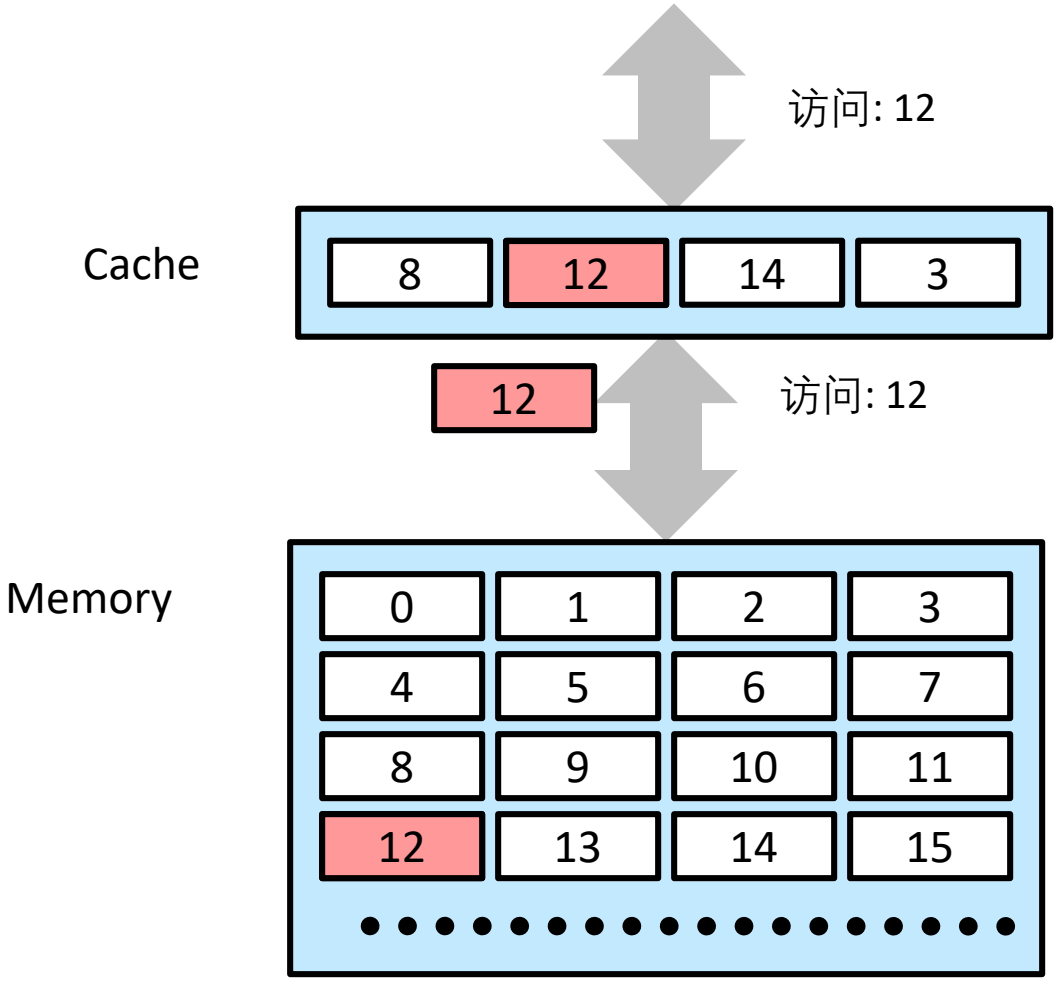


Block b 中的元素被访问

Block b 在 cache 中:

Hit!

Cache 基本概念: 失效 Miss



***Block b* 中的元素被访问**

***Block b* 不在 *Cache* 中：**
Miss!

从内存中取出 *block b*

将 *block b* 放置在 *cache* 中：

- 放置策略:
  决定 block b 将被放置在哪里
- 替换策略:
  决定哪个 block 将会被替换

Cache基本概念：命中率

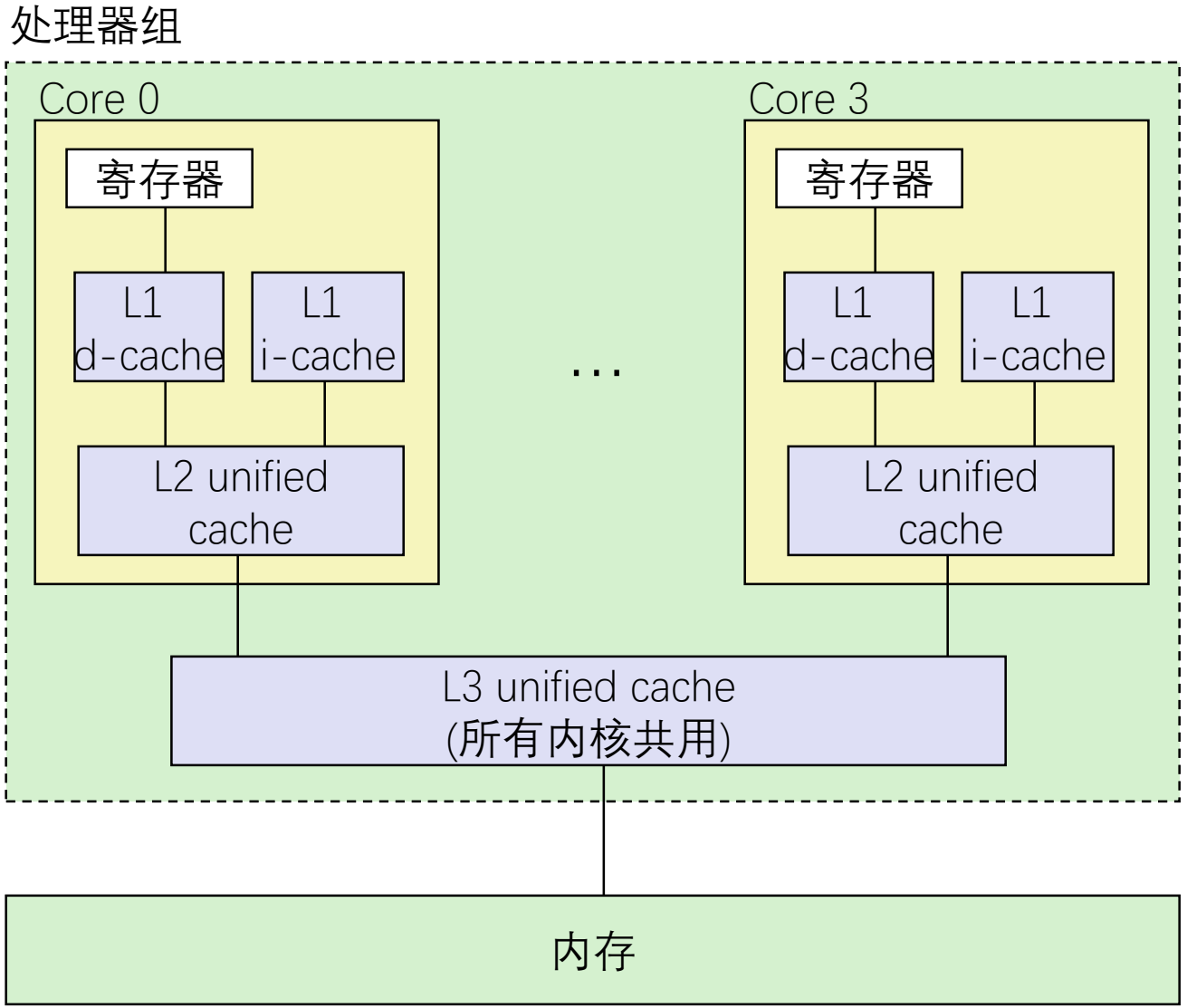
- N_c表示Cache完成存取访问的总次数
- N_m表示主存完成存取访问的总次数
- Cache命中率h

$$h=N_c/(N_c+N_m)$$

- t_c表示命中Cache时的访问时间
- t_m表示命中主存时的访问时间
- t_a平均访问时间

$$t_a=ht_c+(1-h)t_m$$

Cache 层次结构举例： Intel Core i7



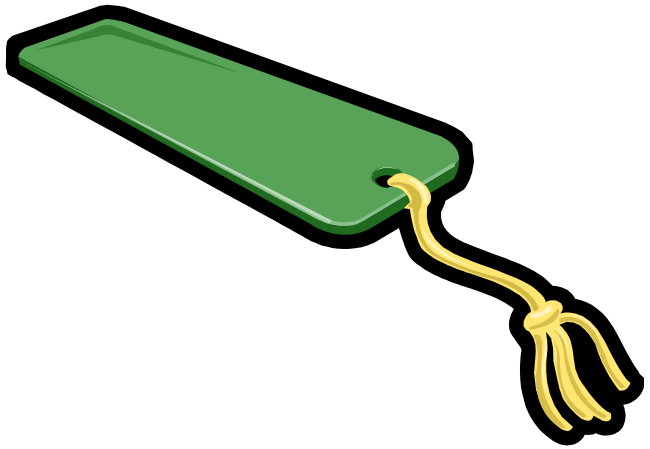
L1 i-cache 和 d-cache:
32 KB, 8-way,
访问: 4 个周期

L2 unified cache:
256 KB, 8-way,
访问: 10 个周期

L3 unified cache:
8 MB, 16-way,
访问: 40-75 个周期

块大小: 每个cache都是64 bytes。

Cache 的关键问题



- 如何判断一个数据在cache中
  - 数据查找　Data Identification
- 如需访问的数据在cache中，存放在什么地方
  - 地址映射　Address Mapping
- Cache满了以后如何处理
  - 替换策略　Placement Policy
- 如何保证cache与memory的一致性
  - 写入策略　Write Policy

将在以后几节介绍!

小结

- Cache 的基本思想和工作原理
- Cache的基本概念
  - 块
  - 命中
  - 失效
  - 命中率
  - 存储器平均访问时间
- 多级高速缓存