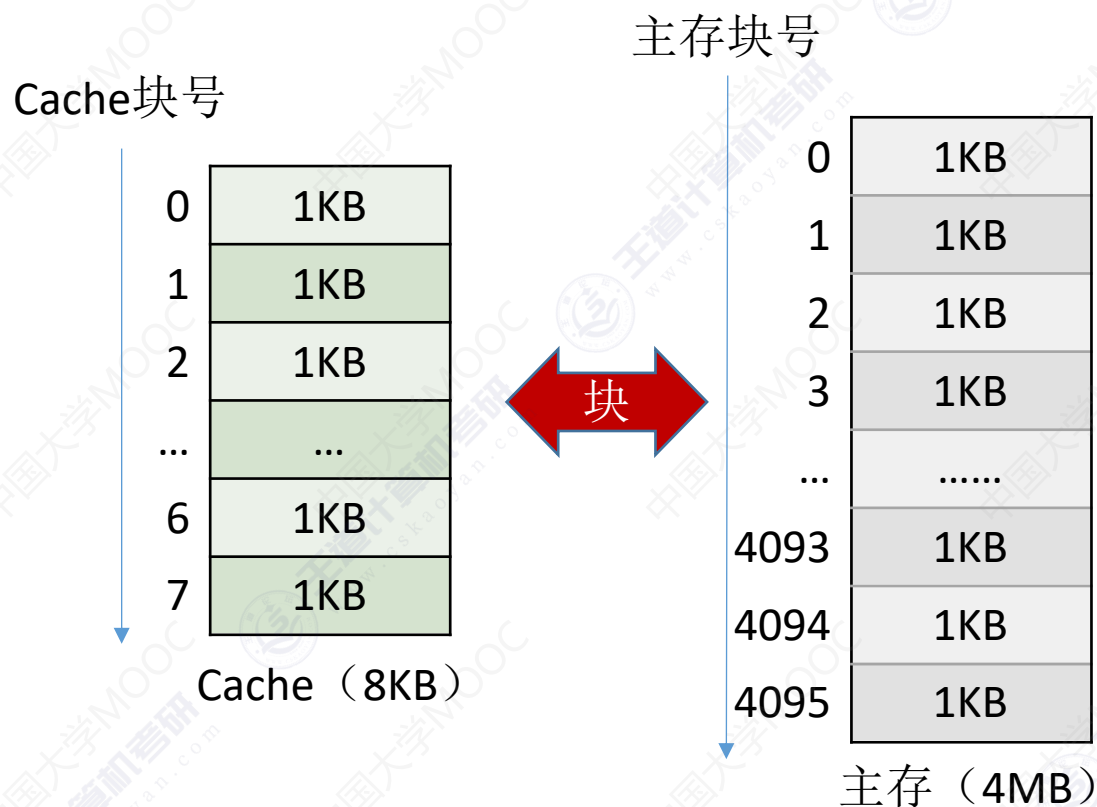


本节内容

# Cache 写策略

## 有待解决的问题



注意：每次被访问的主存块，一定会被立即调入Cache

主存的地址共22位：

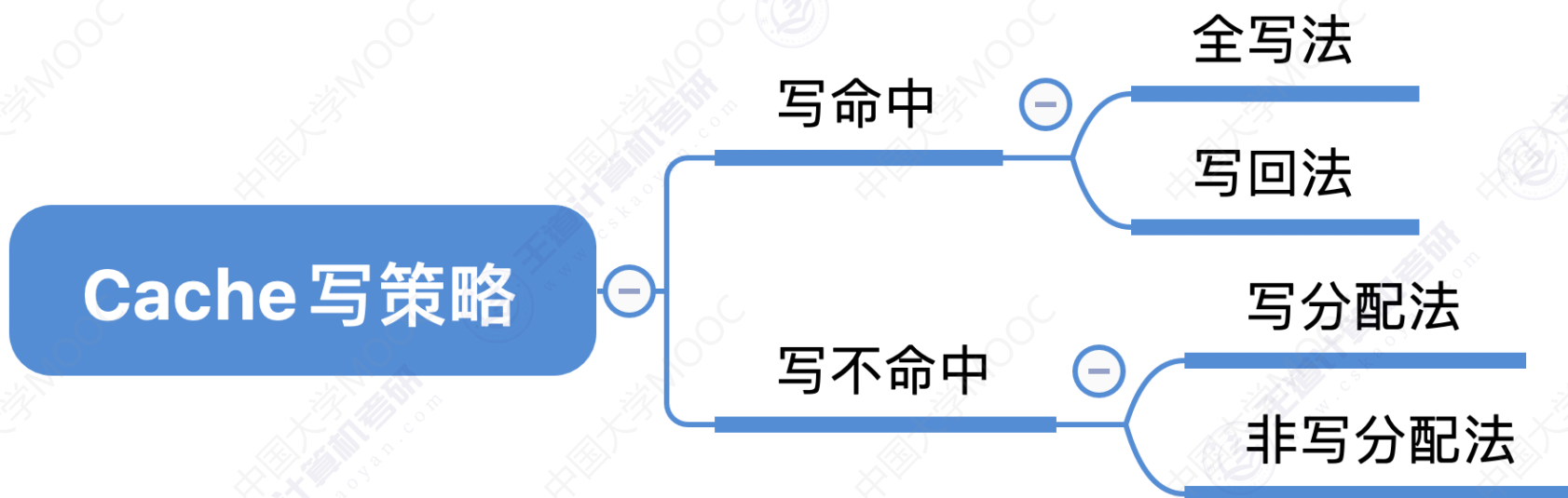
块号	块内地址
12位	10位

$$4M=2^{22}, 1K=2^{10}$$

整个主存被分为  $2^{12} = 4096$  块

- 如何区分 Cache 与 主存 的数据块对应关系？——Cache和主存的映射方式
- Cache 很小，主存很大。如果Cache满了怎么办？——替换算法
- CPU修改了Cache中的数据副本，如何确保主存中数据母本的一致性？——Cache写策略

## 本节总览

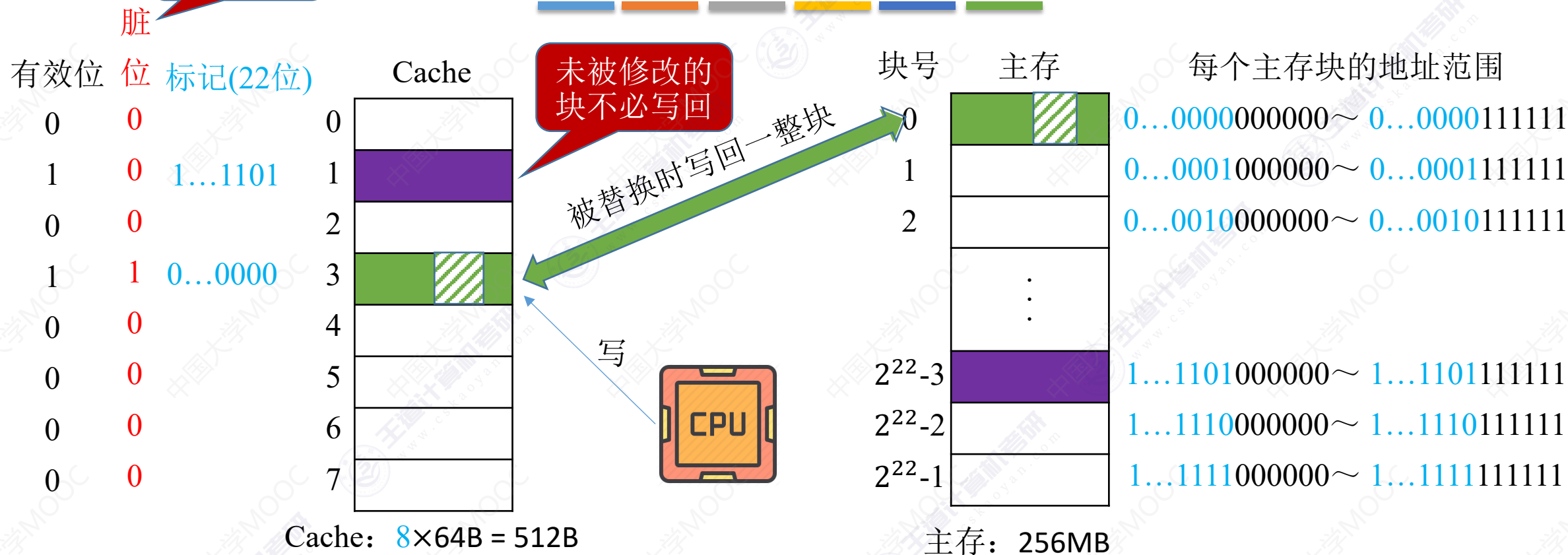


为何不讨论读命中、读不命中的情况？

读操作不会导致Cache和主存的数据不一致

## 写命中

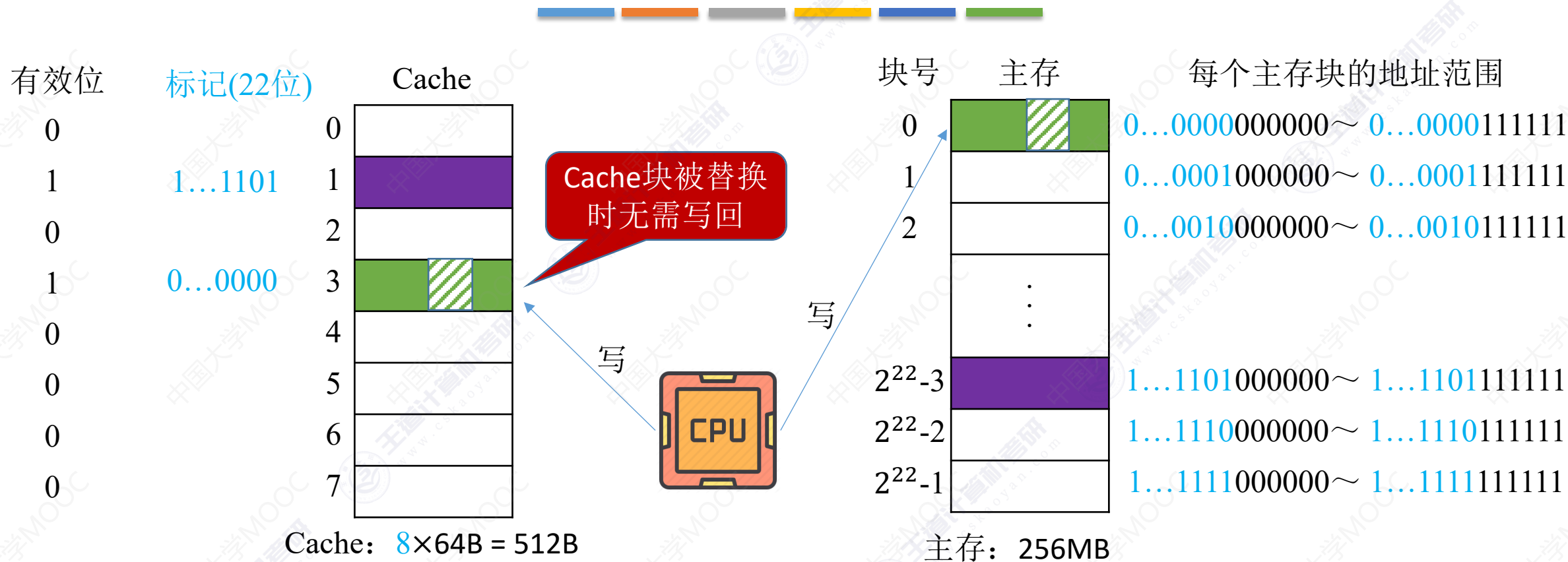
表示是否  
被修改过



写回法(write-back) —— 当CPU对Cache写命中时，只修改Cache的内容，而不立即写入主存，只有当此块被换出时才写回主存

减少了访存次数，但存在数据不一致的隐患。

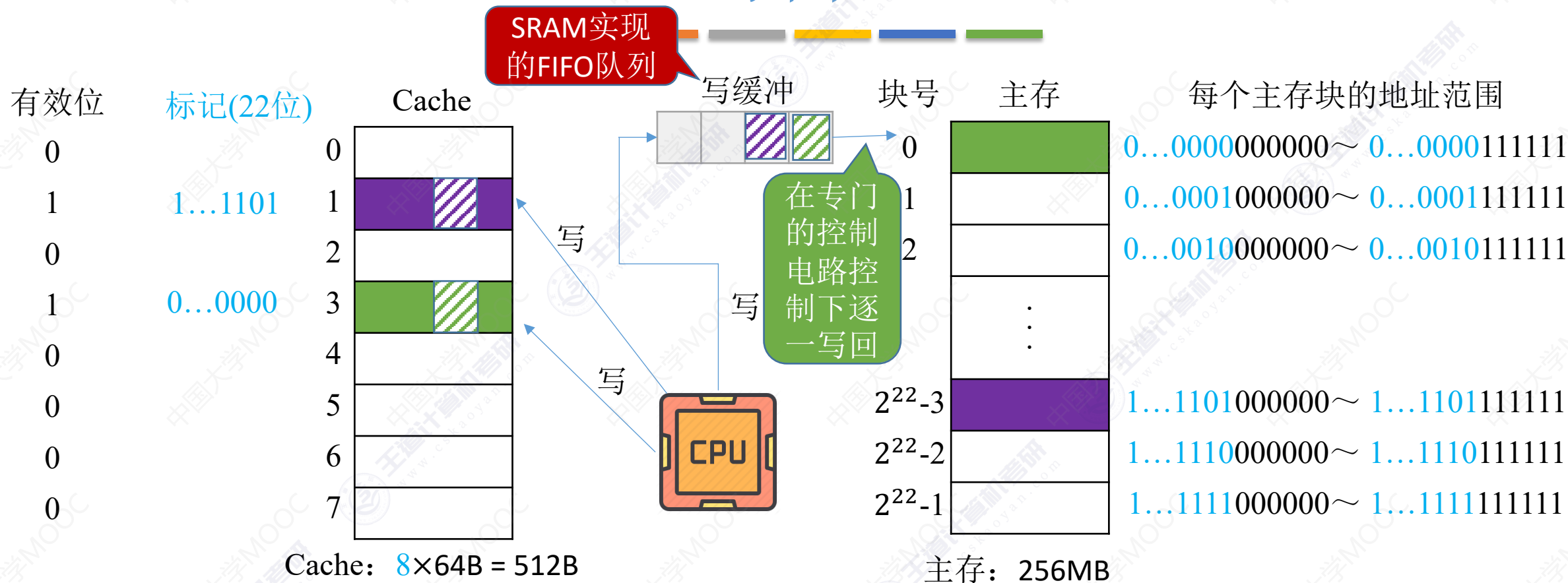
## 写命中



全写法(写直通法, write-through) —— 当CPU对Cache写命中时, 必须把数据同时写入Cache和主存, 一般使用写缓冲(write buffer)

访存次数增加, 速度变慢, 但更能保证数据一致性

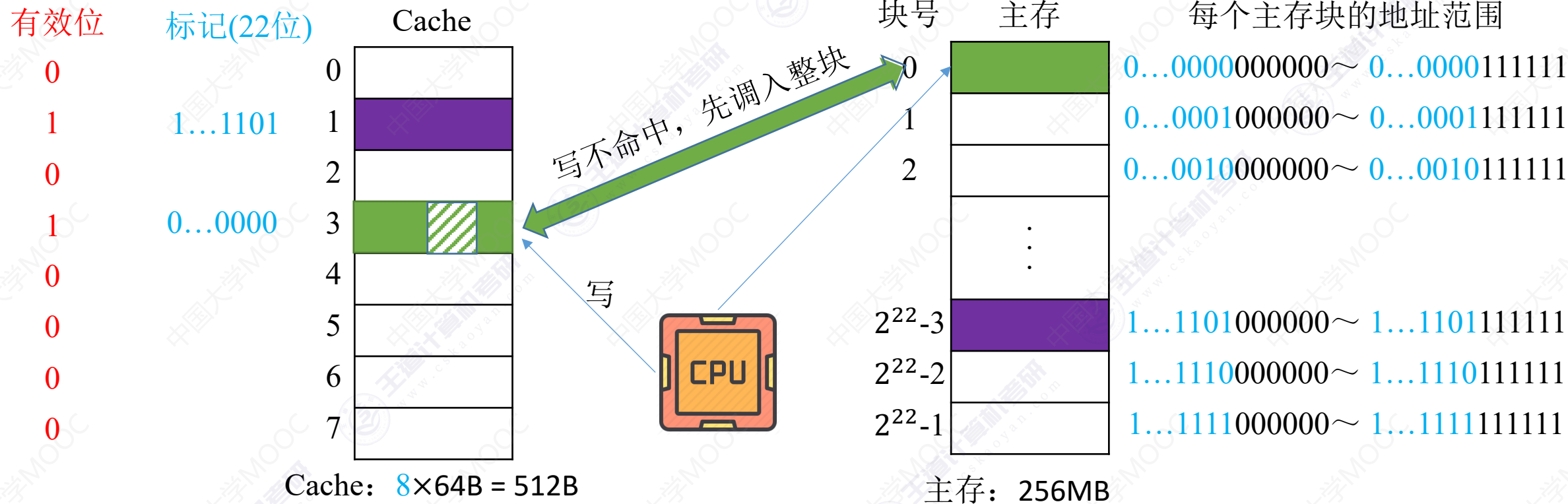
## 写命中



全写法(写直通法, write-through) —— 当CPU对Cache写命中时, 必须把数据同时写入Cache和主存, 一般使用写缓冲(write buffer)

使用写缓冲, CPU写的速度很快, 若写操作不频繁, 则效果很好。若写操作很频繁, 可能会因为写缓冲饱和而发生阻塞

## 写不命中

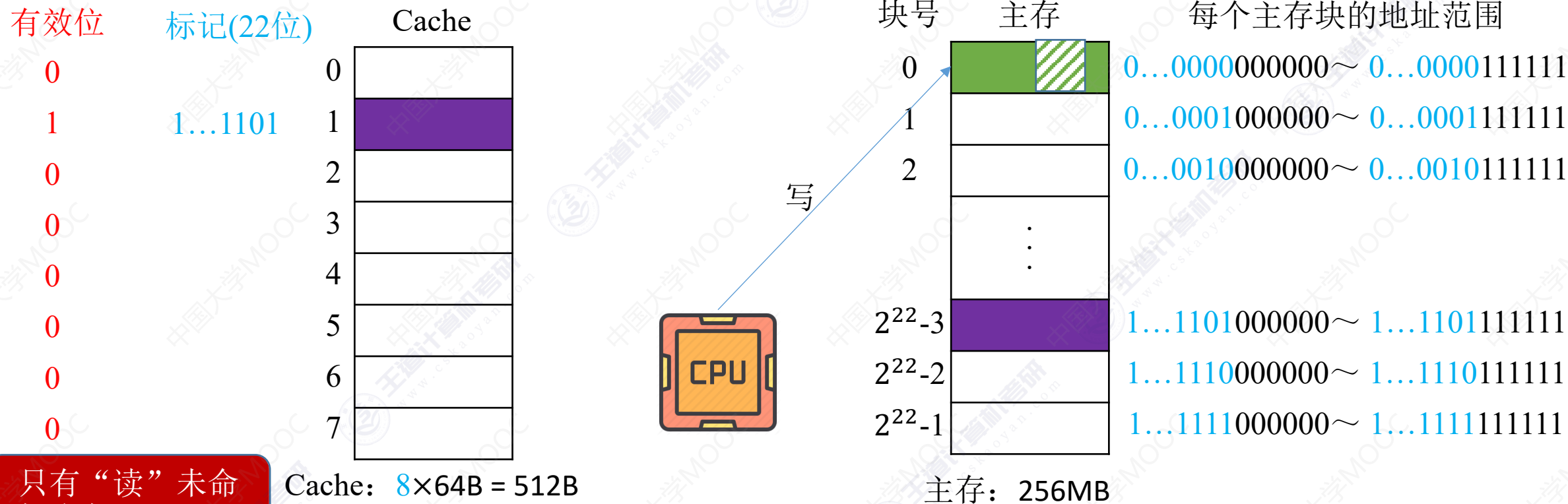


写分配法(write-allocate)——当CPU对Cache写不命中时, 把主存中的块调入Cache, 在Cache中修改。通常搭配写回法使用。

写回法(write-back)——当CPU对Cache写命中时, 只修改Cache的内容, 而不立即写入主存, 只有当此块被换出时才写回主存



## 写不命中

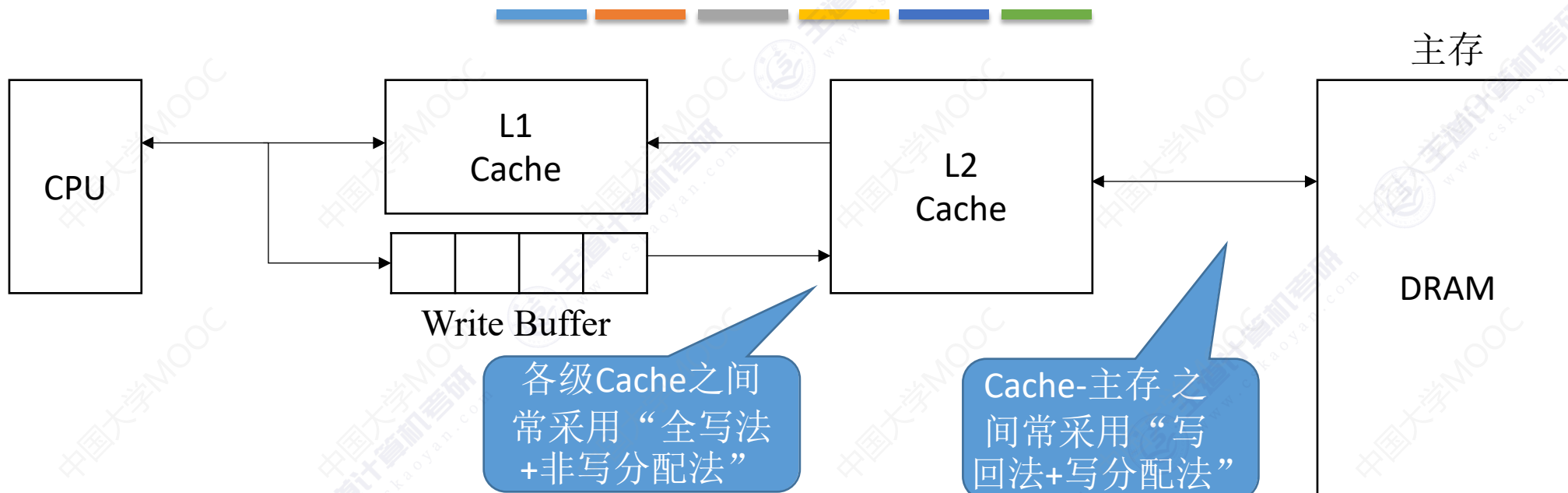


非写分配法(not-write-allocate)——当CPU对Cache写不命中时只写入主存，不调入Cache。  
搭配全写法使用。

全写法(写直通法, write-through) —— 当CPU对Cache写命中时，必须把数据同时写入Cache和主存，一般使用写缓冲(write buffer)



## 多级Cache



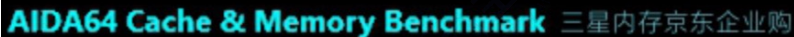
AIDA64 Cache & Memory Benchmark 三星内存京东企业购

	Read	Write	Copy	Latency
Memory	37051 MB/s	37566 MB/s	31781 MB/s	61.1 ns
L1 Cache	988.09 GB/s	497.62 GB/s	992.73 GB/s	1.0 ns
L2 Cache	399.35 GB/s	243.65 GB/s	296.69 GB/s	3.1 ns
L3 Cache	249.92 GB/s	163.07 GB/s	208.80 GB/s	13.5 ns
CPU Type	QuadCore Intel Core i5-9300H (Coffee Lake-H, 8GA1440)			

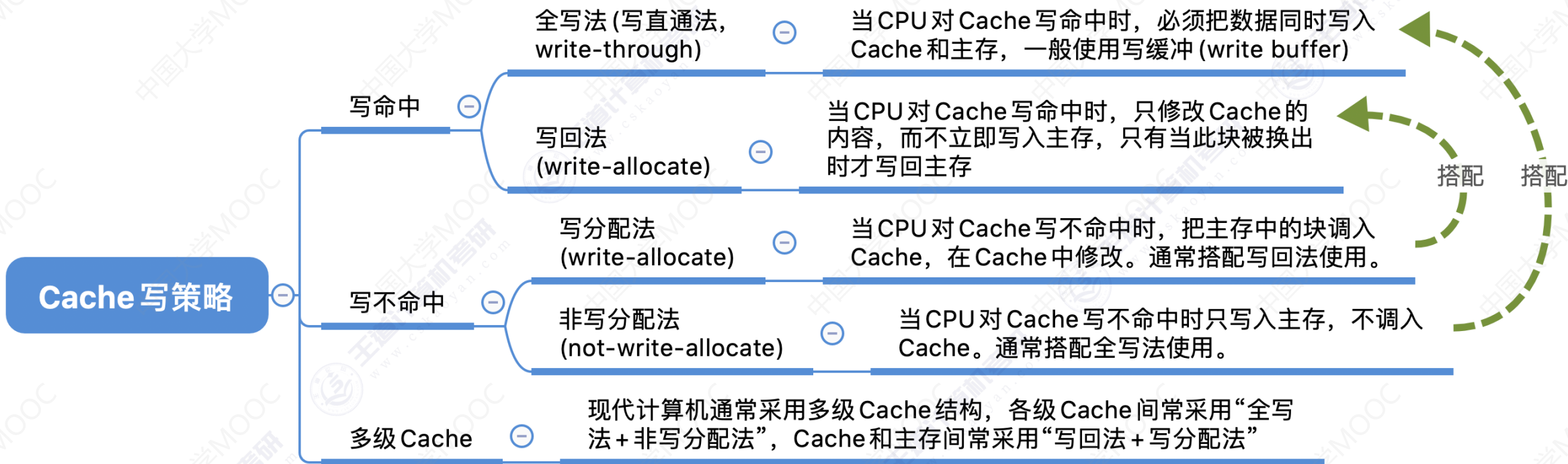
现代计算机常采用**多级Cache**  
离CPU越近的速度越快，容量越小  
离CPU越远的速度越慢，容量越大



各级Cache之间  
常采用“全写法  
+非写分配法”



# 知识回顾





公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研