

本节内容

Cache

基本原理
基本概念

存储系统存在的问题

双端口RAM、多模块存储器提高存储器的工作速度

优化后速度与CPU差距
依然很大

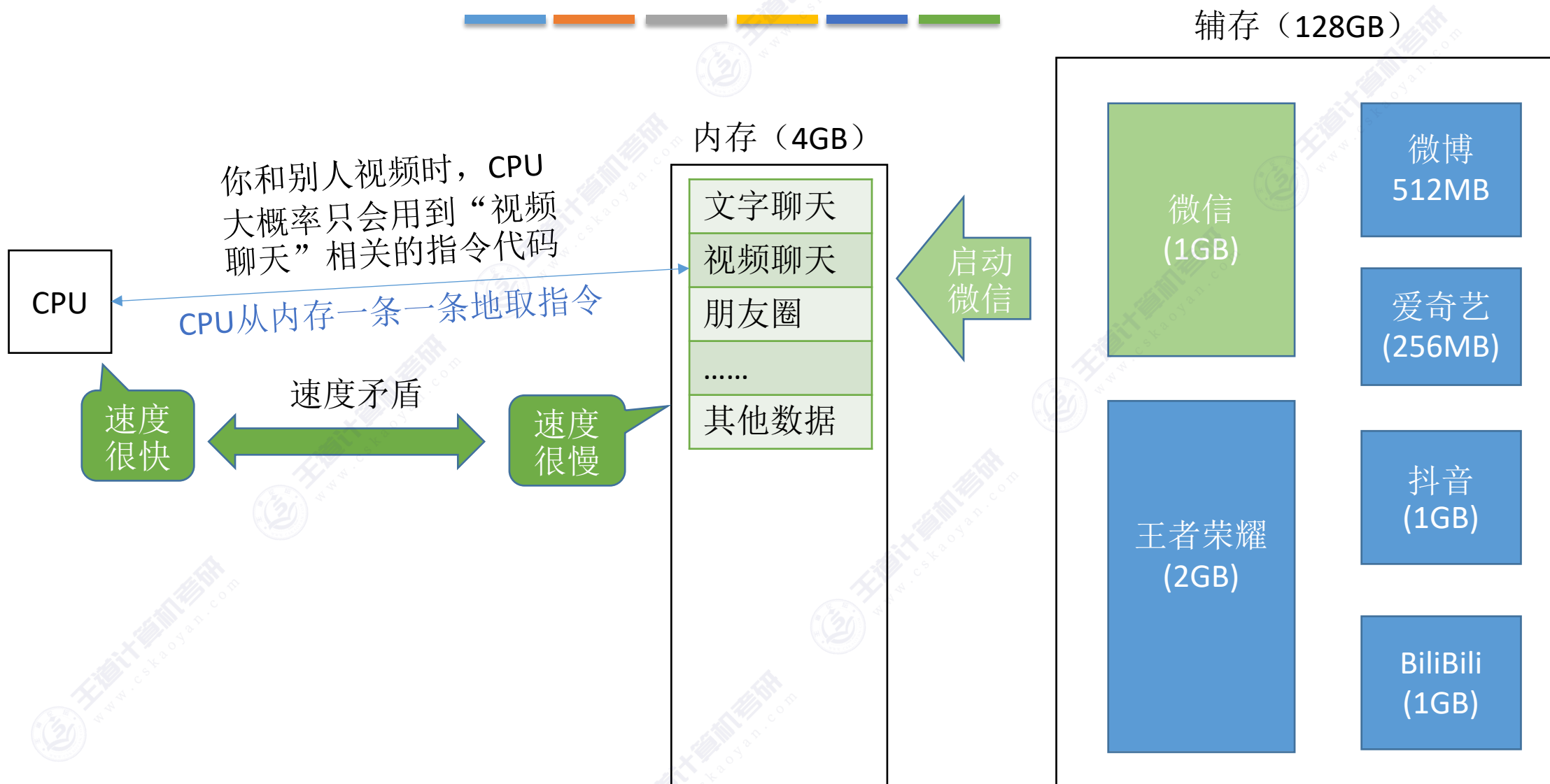
更高速的存储单元设计

存储器价格↑ 容量↓

存储体系的改善
“Cache-主存”层次

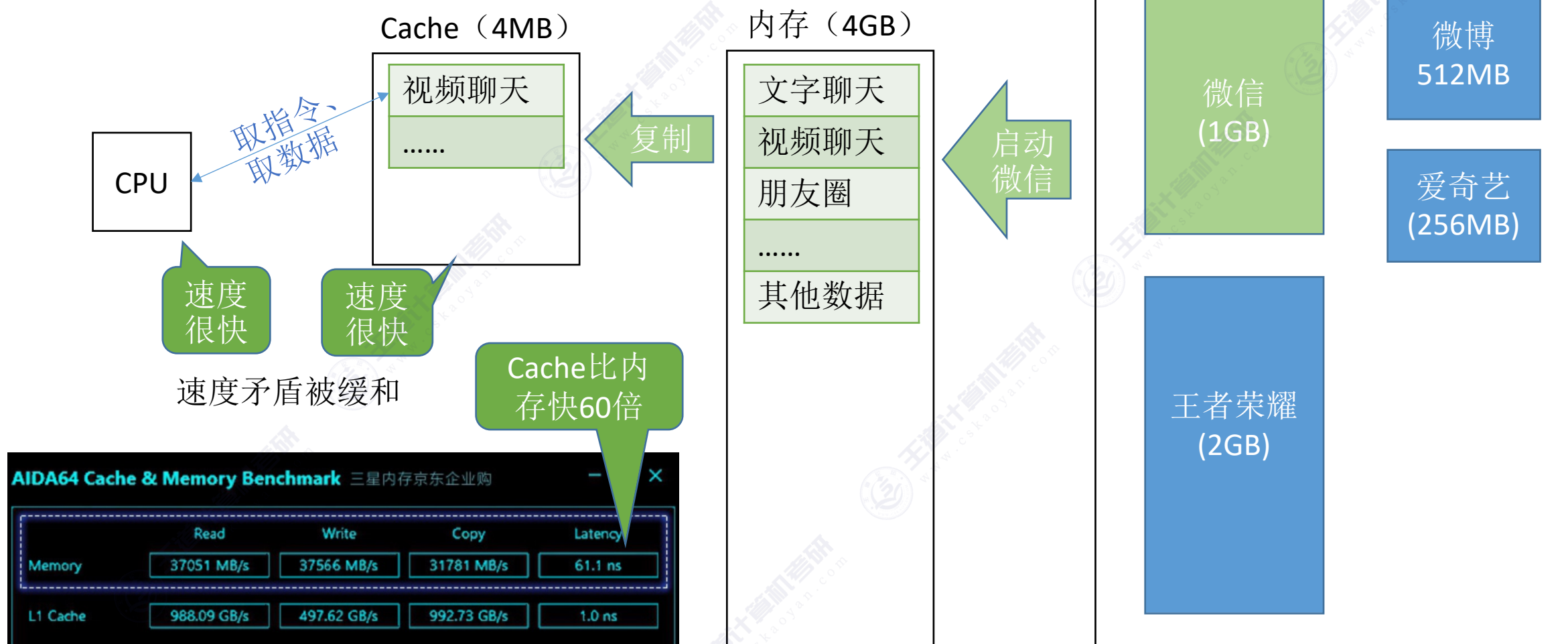
程序访问的局部性原理

Cache的工作原理



Cache的工作原理

注：实际上，Cache 被集成在CPU内部
Cache用SRAM实现，速度快，成本高



局部性原理

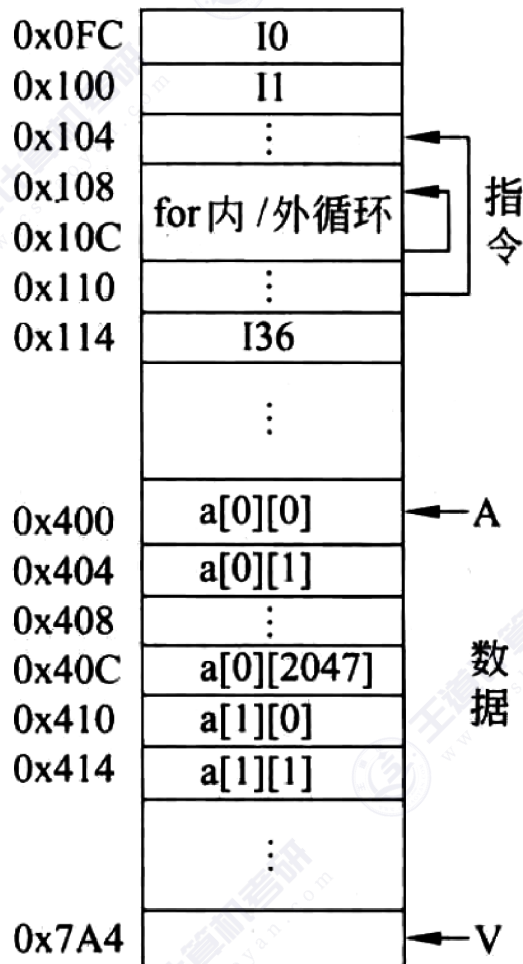
Eg: 数组元素、顺序执行的指令代码

程序A:

```
1  int sumarrayrows(int a[M][N])
2  {
3      int i, j, sum = 0;
4      for (i = 0; i < M; i++)
5          for (j = 0; j < N; j++)
6              sum += a[i][j];
7      return sum;
8  }
```

程序B:

```
1  int sumarraycols(int a[M][N])
2  {
3      int i, j, sum = 0;
4      for (j = 0; j < N; j++)
5          for (i = 0; i < M; i++)
6              sum += a[i][j];
7      return sum;
8  }
```



空间局部性: 在最近的未来要用到的信息(指令和数据), 很可能与现在正在使用的信息在存储空间上是邻近的

时间局部性: 在最近的未来要用到的信息, 很可能是现在正在使用的信息

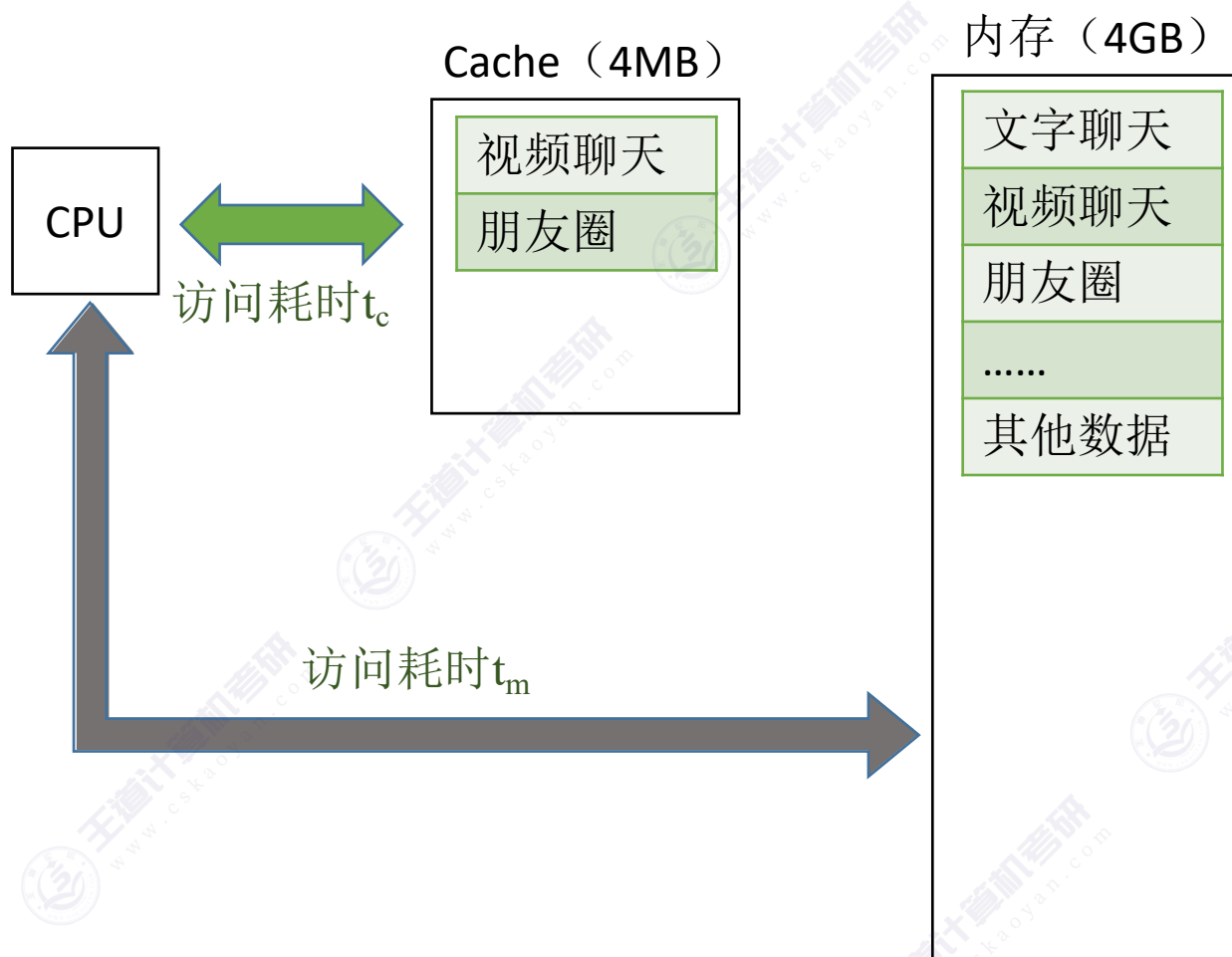
Eg: 循环结构的指令代码

基于局部性原理, 不难想到, 可以把CPU目前访问的地址“周围”的部分数据放到Cache中

程序B 按“列优先”访问二维数组, 空间局部性更差

指令和数据在内存中的存储

性能分析



设 t_c 为访问一次Cache所需时间,
 t_m 为访问一次主存所需时间

命中率 H : CPU欲访问的信息已在
Cache中的比率

缺失 (未命中) 率 $M = 1 - H$

Cache—主存 系统的**平均访问时间 t** 为

$$t = Ht_c + (1 - H)(t_c + t_m)$$

先访问Cache, 若Cache未命中再访问主存

或 $t = Ht_c + (1 - H)t_m$

同时访问 Cache和主存, 若Cache
命中则立即停止访问主存

性能分析

【例3-2】 假设Cache的速度是主存的5倍，且Cache的命中率为95%，则采用Cache后，存储器性能提高多少（设Cache和主存同时被访问，若Cache命中则中断访问主存）？

设Cache的存取周期为 t ，则主存的存取周期为 $5t$

若Cache和主存同时访问，命中时访问时间为 t ，未命中时访问时间为 $5t$

平均访问时间为 $0.95 \times t + 0.05 \times 5t = 1.2t$

故性能为原来的 $\frac{5t}{1.2t} \approx 4.17$ 倍

若先访问Cache再访问主存，命中时访问时间为 t ，未命中时访问时间为 $t+5t$

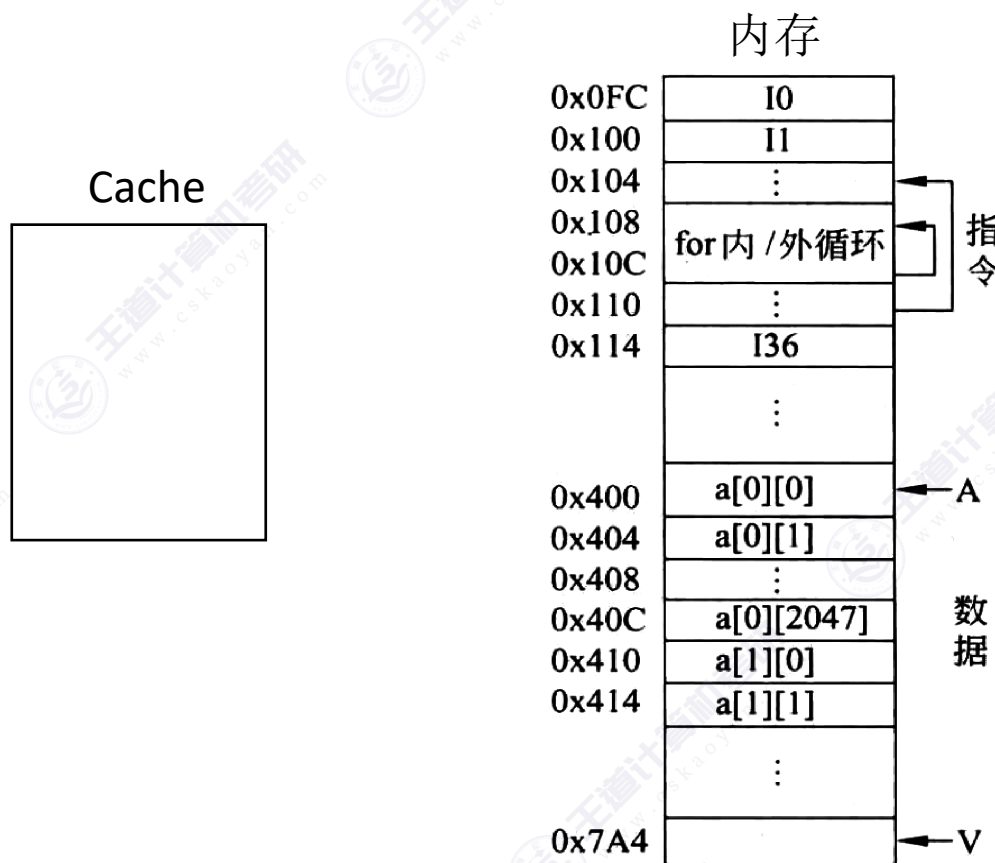
平均访问时间为 $T_a = 0.95 \times t + 0.05 \times 6t = 1.25t$

故性能为原来的 $\frac{5t}{1.25t} = 4$ 倍

有待解决的问题

基于局部性原理，不难想到，可以把CPU目前访问的地址“**周围**”的部分数据放到Cache中。如何界定“**周围**”？

将主存的存储空间“**分块**”，如：每 1KB 为一块。**主存与Cache之间以“块”为单位进行数据交换**



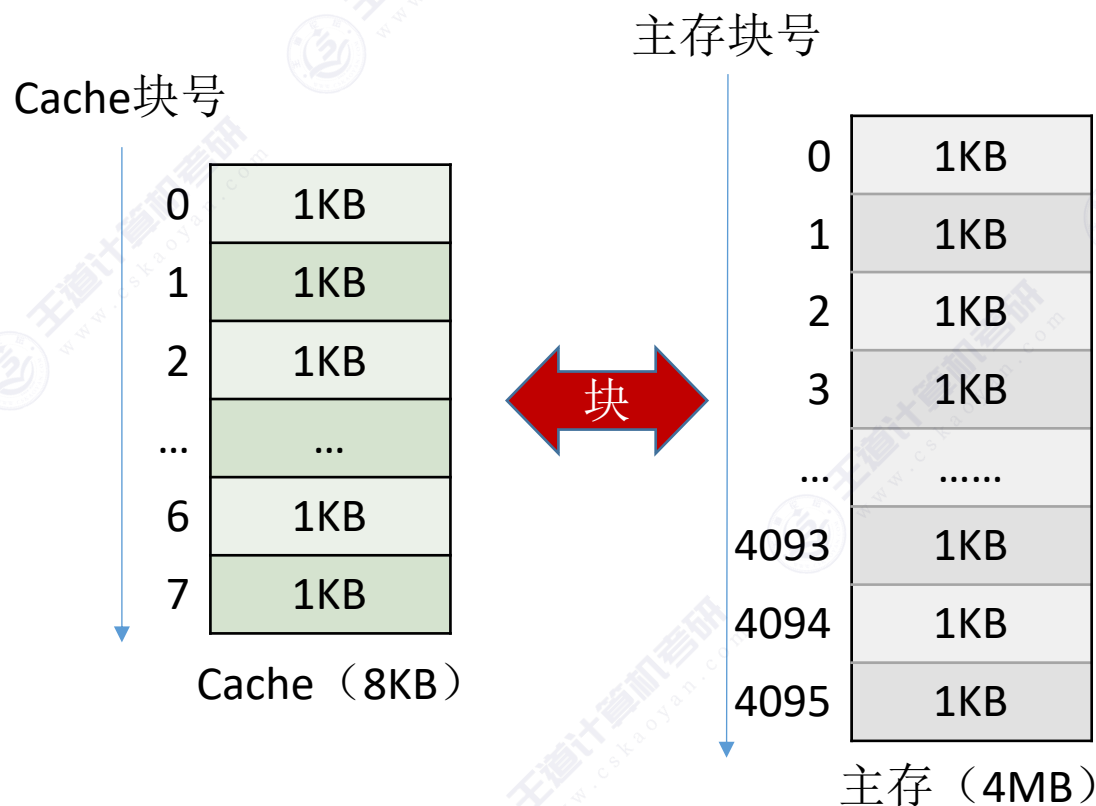
有待解决的问题

基于局部性原理，不难想到，可以把CPU目前访问的地址“**周围**”的部分数据放到Cache中。如何界定“**周围**”？

将主存的存储空间“**分块**”，如：每 1KB 为一块。主存与Cache之间以“**块**”为单位进行数据交换

注：操作系统中，通常将主存中的“一个**块**”也称为“一个**页/页面/页框**”

Cache中的“**块**”也称为“**行**”

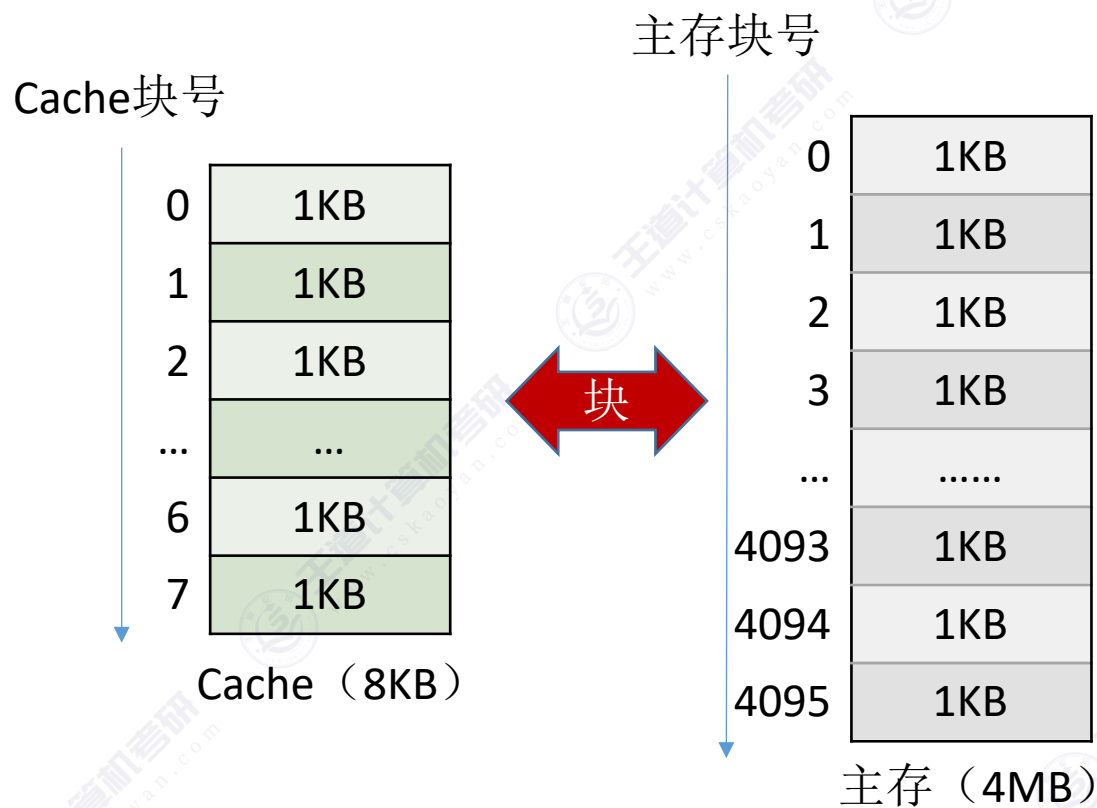


主存的地址共22位：

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

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

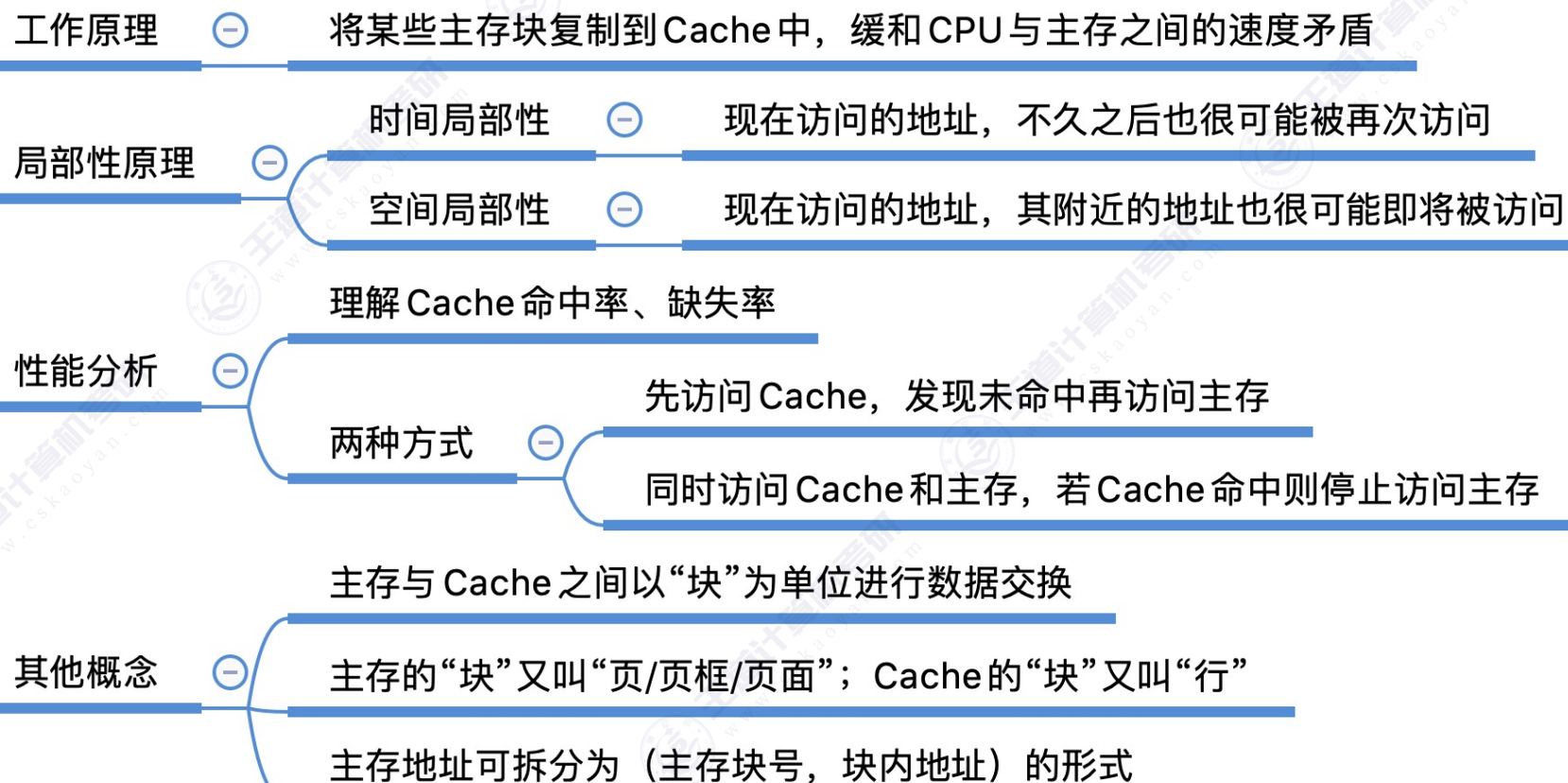
$$4M=2^{22}, 1K=2^{10}$$

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

知识回顾

高速缓冲存储器 Cache



每次被访问的主存块，
一定会被立即调入Cache

- 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



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