



第七章 存储系统(三)

秦磊华 计算机学院

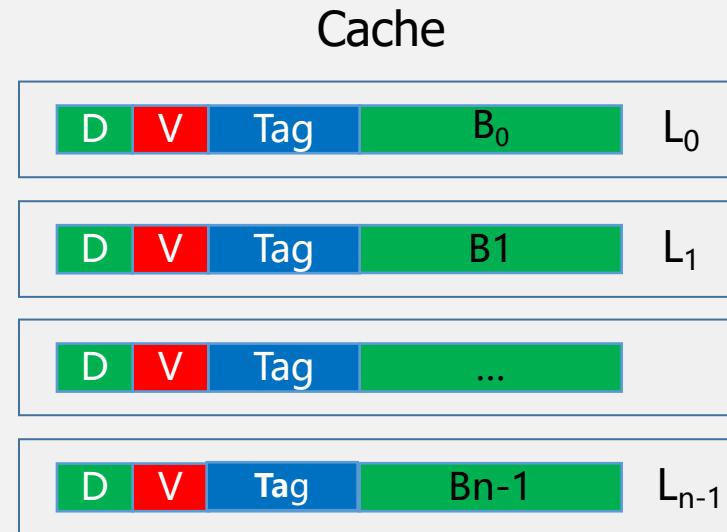
7.7 替换算法

7.8 Cache和调度算法应用举例

7.9 虚拟存储器



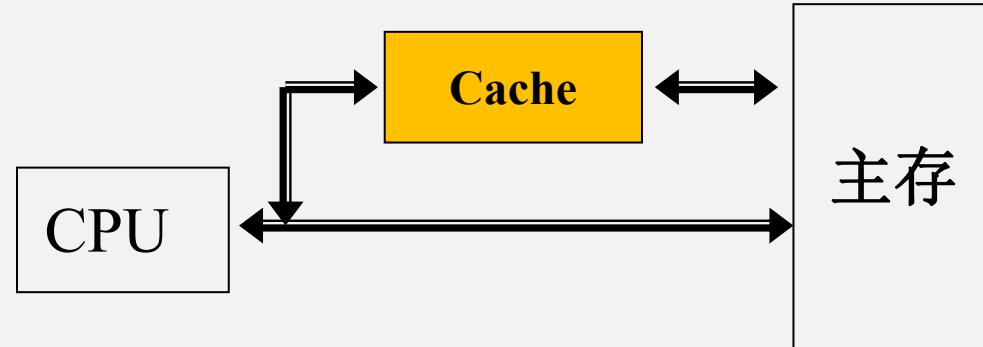
1. 需要替换算法的原因



程序运行一段时间后，Cache存储空间被占满，当有新数据被调入时，就需要借助某种机制决定替换的对象。

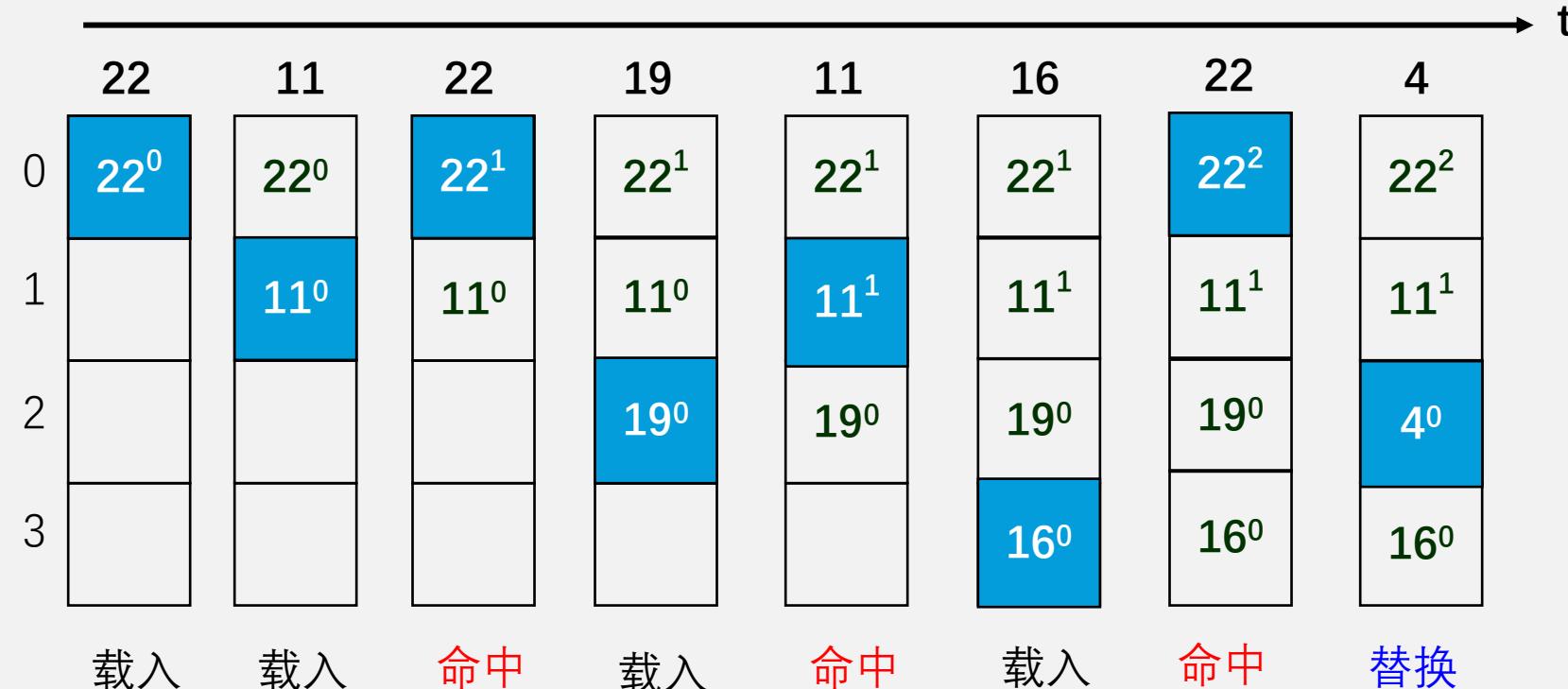
2. 几种常见的调度算法

- ◆ 先进先出法-FIFO
- ◆ 最不经常使用---LFU
- ◆ 最近最少用--- LRU
- ◆ 随机替换
- ◆ 最优替换算法



2. 几种常见的调度算法

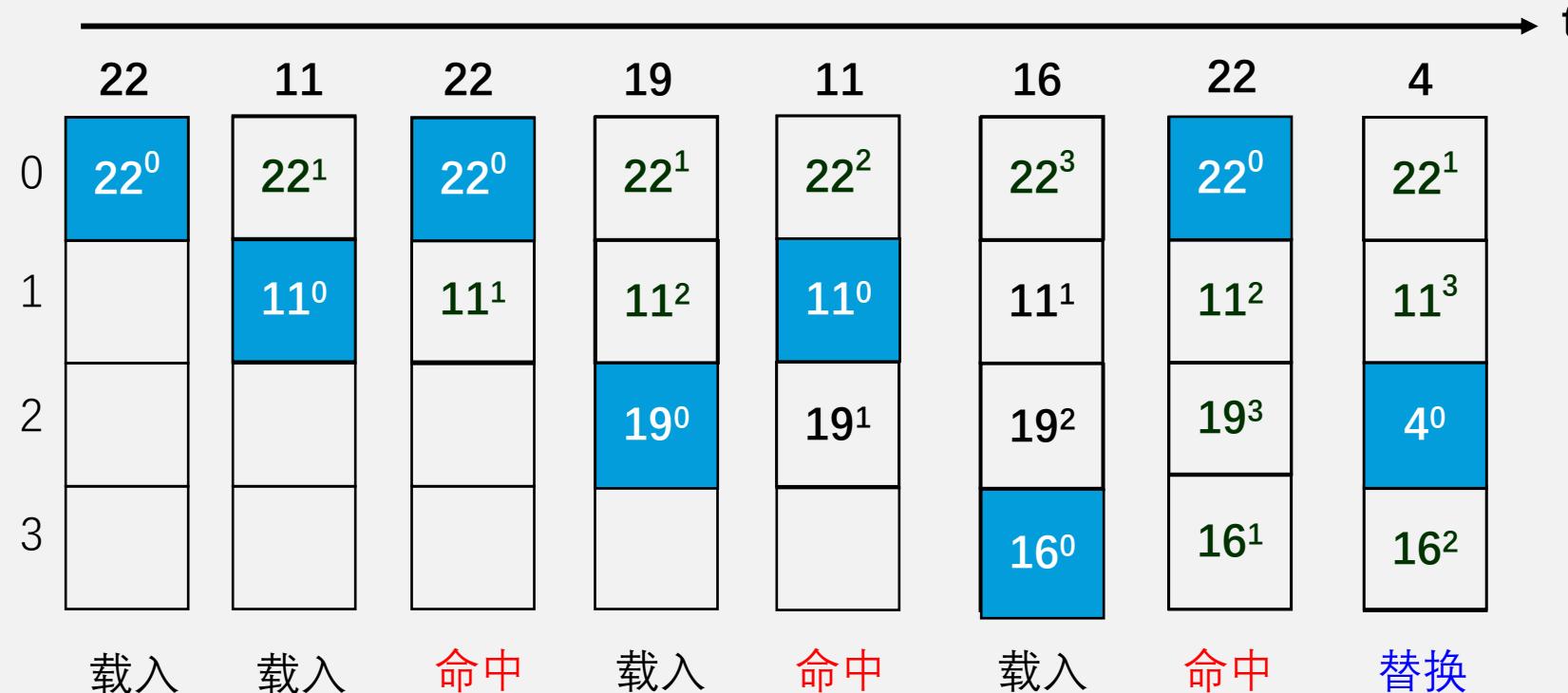
(1) 最不经常使用LFU(least frequently used)



命中率 = 3/8 = 37.5%

2. 几种常见的调度算法

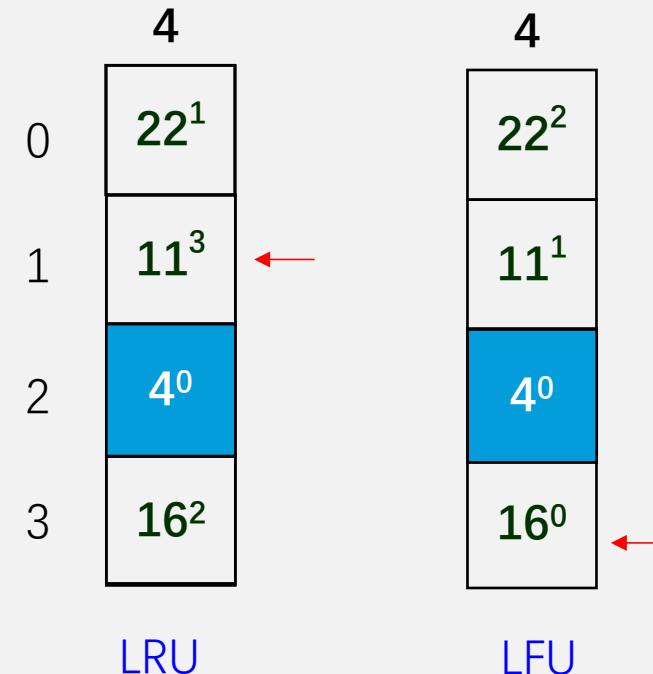
(2) 最近最少用LRU(Least Recently Used)



$$\text{命中率} = 3/8 = 37.5\%$$

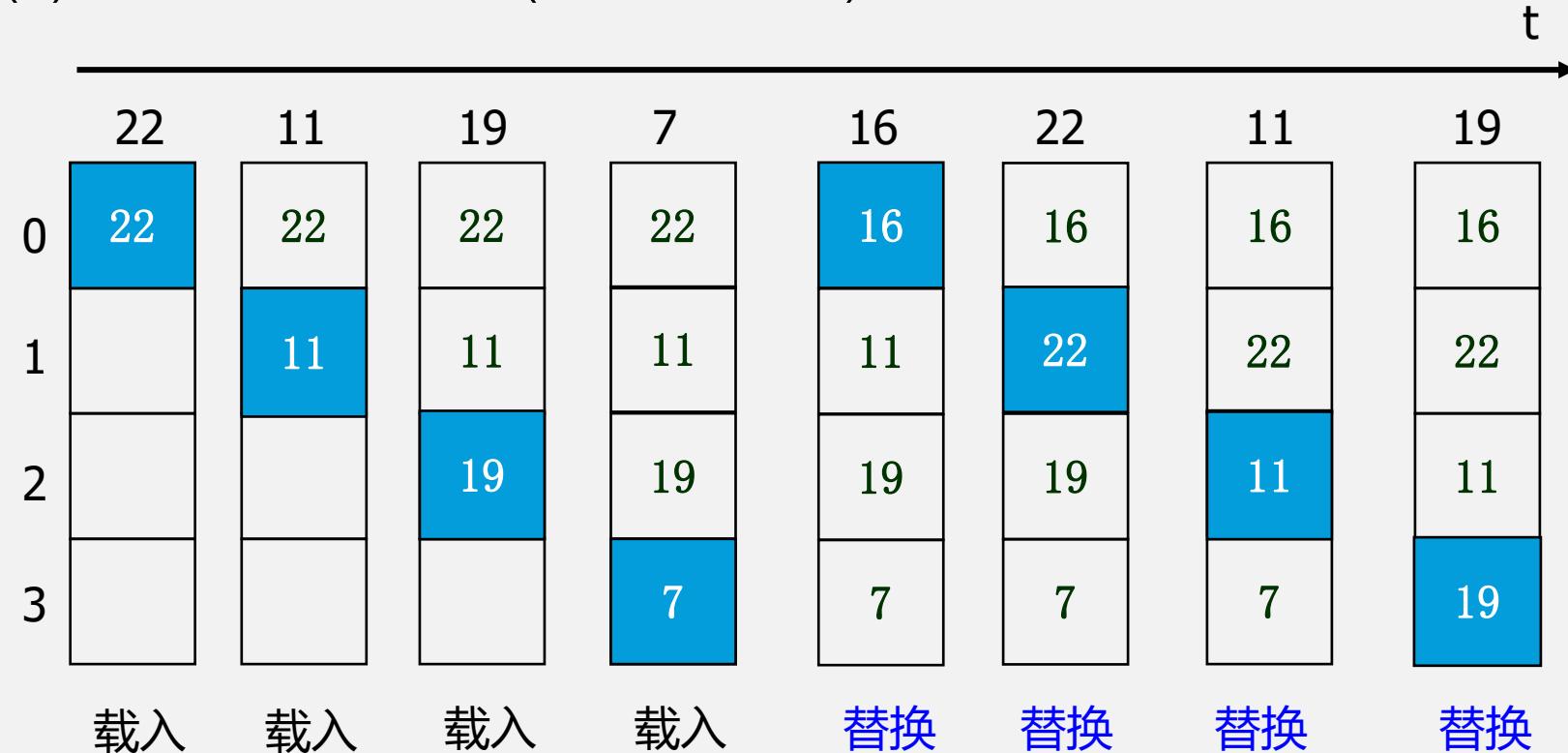
2. 几种常见的调度算法

(3) LRU 与 LFU 对比



2. 几种常见的调度算法

(4) 调度算法的抖动(以FIFO为例)

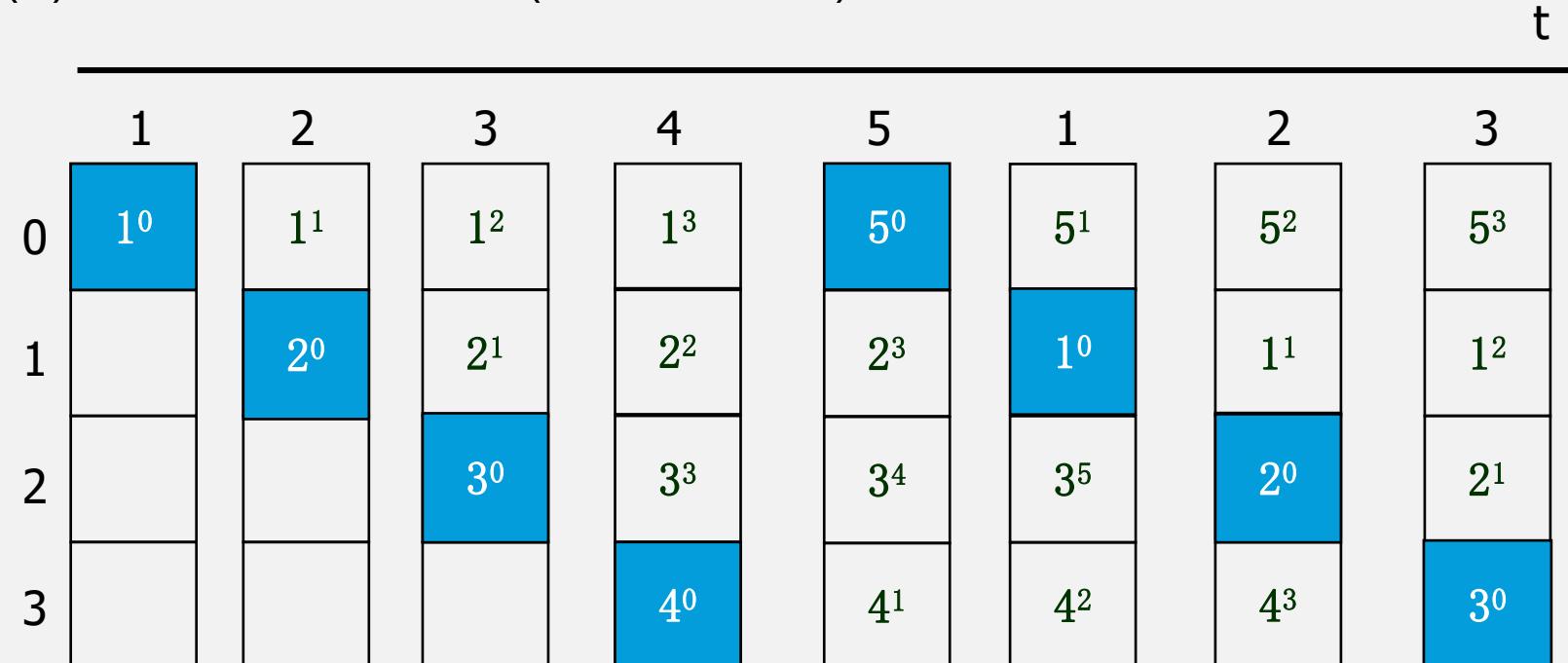


$$\text{命中率} = 0/8 = 0\%$$

- ◆ 抖动的原因?
- ◆ 抖动消除的方法?
- ◆ 其它替换算法会产生抖动吗?

2. 几种常见的调度算法

(4) 调度算法的抖动(以LRU为例)



2. 几种常见的调度算法

(5) Cache不同映射方式的调度算法特征

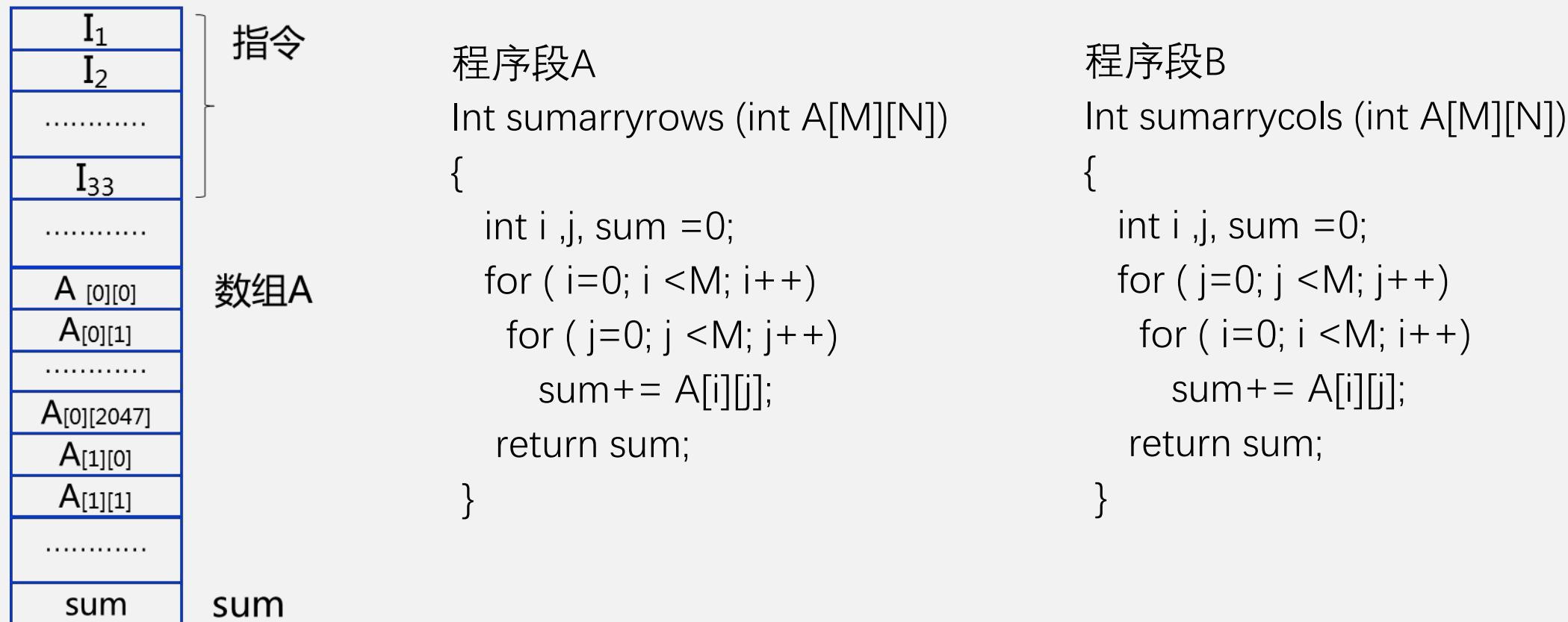


- ◆全相联映射没有Index字段，各种调度算法要比较全部Cache行；
- ◆组相联映射中，各种调度算法要比较Index所指特定组的全部Cache行；
- ◆直接映射中，各种调度算法要比较Cache哪些行？

7.8 Cache和调度算法应用举例

例1 局部性分析。以下程序A和B中，哪一个对数组A[2047][2047]引用的**空间局部性**更好？**时间局部性**呢？变量sum的空间局部性和时间局部性如何？

数组和指令**行优先**顺序存放



```
int sumarryrows (int A[M][N])
{
    int i ,j, sum =0;
    for ( i=0; i <M; i++)
        for ( j=0; j <M; j++)
            sum+= A[i][j];
    return sum;
}
```

```
int sumarrycols (int A[M][N])
{
    int i ,j, sum =0;
    for ( j=0; j <M; j++)
        for ( i=0; i <M; i++)
            sum+= A[i][j];
    return sum;
}
```

例1 局部性分析。以下程序A和B中，哪一个对数组A[2048][2048]引用的**空间局部性**更好？**时间局部性**呢？变量sum的空间局部性和时间局部性如何？

时间局部性的程序结构



循环



循环体中的**同一对象具有时间局部性**

空间局部性的程序结构

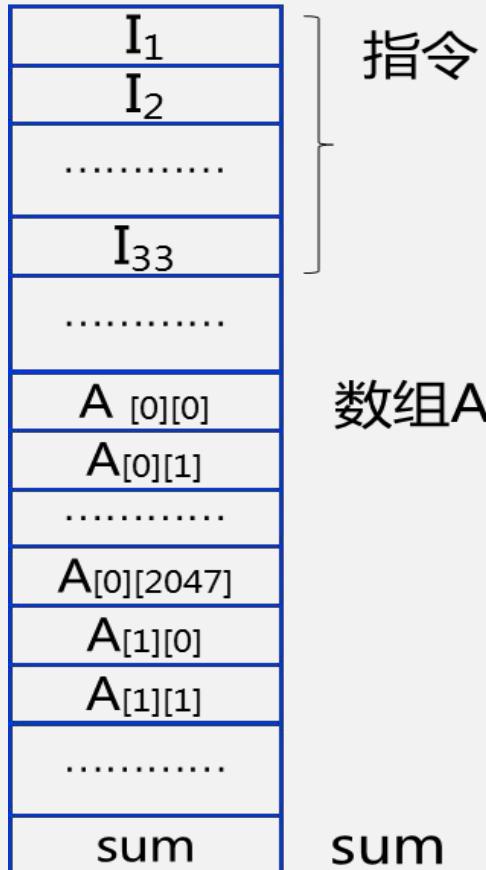


顺序



顺序结构中的**不同对象**间具有**空间局部性**

例1 局部性分析。以下程序A和B中，哪一个对数组A[2048][2048]引用的**空间局部性**更好？**时间局部性**呢？变量sum的空间局部性和时间局部性如何？



程序段A

```
Int sumarryrows (int A[M][N])
{
    int i ,j, sum =0;
    for ( i=0; i <M; i++)
        for ( j=0; j <M; j++)
            sum+= A[i][j];
    return sum;
}
```

- 数组访问顺序与存放顺序一致，空间局部性好！
- 该循环体中的数组时间局部性好吗？
- 变量sum在循环体中，时间局部性好！
- 变量sum只有一个变量，空间局部性差！

7.8 Cache和调度算法应用举例

例1 局部性分析。以下程序A和B中，哪一个对数组A[2048][2048]引用的**空间局部性**更好？**时间局部性**呢？变量sum的空间局部性和时间局部性如何？



程序段B

```
Int sumarrycols (int A[M][N])
{
    int i, j, sum = 0;
    for (j=0; j < M; j++)
        for (i=0; i < M; i++)
            sum += A[i][j];
    return sum;
}
```

- 数组访问顺序与存放顺序不一致，空间局部性差！？
- 数组虽在循环体中，但每个元素只用一次，时间局部性差！
- 变量sum在循环体中，时间局部性好！
- 变量sum只有一个变量，空间局部性差！

7.8 Cache和调度算法应用举例

例2 主存和Cache之间采用直接映射方式，块大小为16B。Cache数据区容量为64KB，主存地址为32位，按字节编址，数据字长32位。要求

- 1)给出直接映射方式下主存地址划分；
- 2)完成Cache访问的硬件实现；
- 3)计算Cache总容量多大。

解：

Tag	Index	块内偏移
-----	-------	------

 \rightarrow 求解有顺序

数据块大小16B \rightarrow 块内偏移地址4位；

Cache数据区容量为64KB \rightarrow $64KB / 16B = 4096$ 行 (Cache)；

Tag字段的位数为 $\rightarrow 32 - 12 - 4 = 16$ 位

 Index字段12位

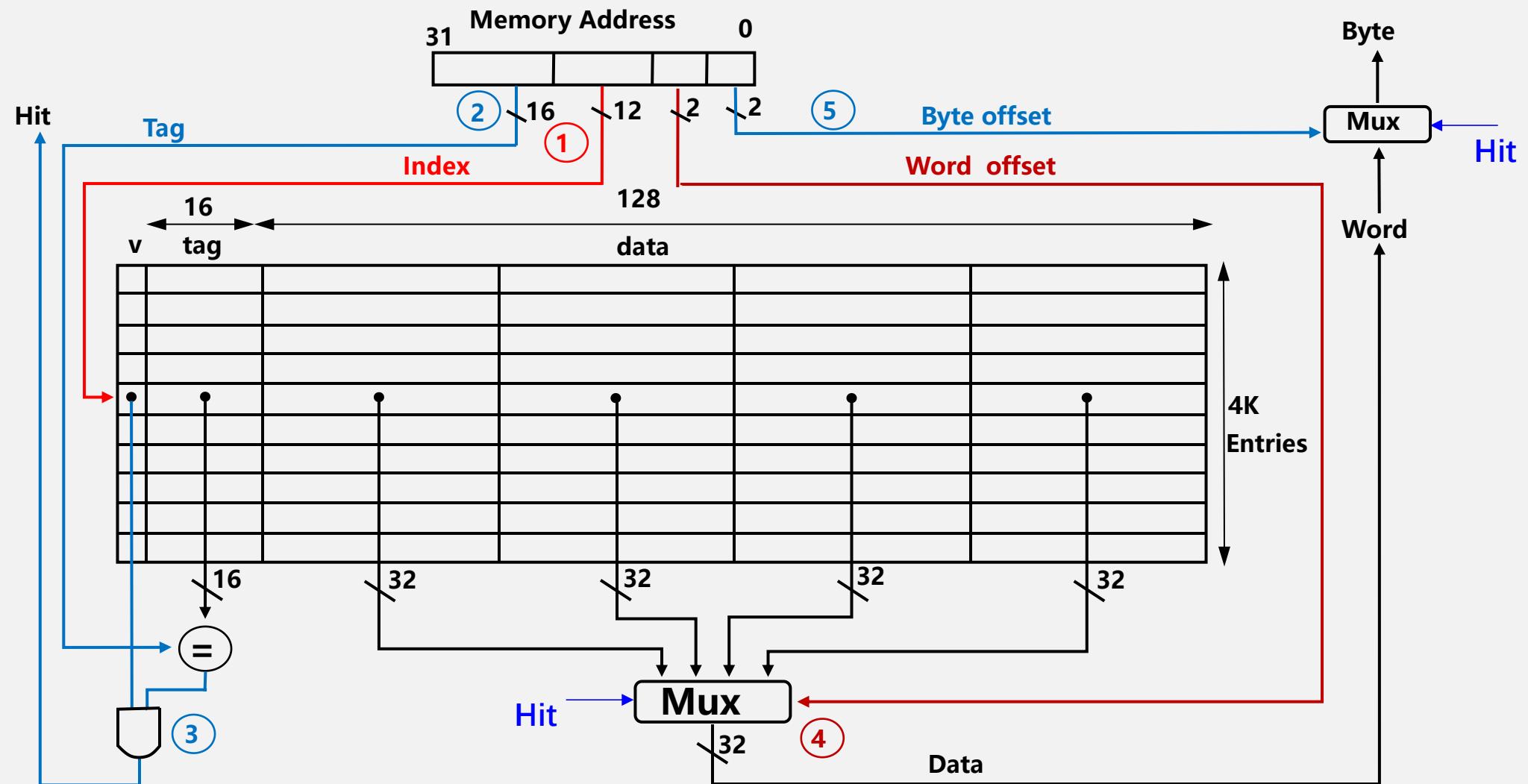
7.8 Cache和调度算法应用举例

2)

Tag(16)

Index(12)

块内偏移(4)



例2 主存和Cache之间采用直接映射方式，块大小为16B。Cache数据区容量为64KB，主存地址为32位，按字节编址，数据字长32位。

3)计算Cache总容量多大 (假定Cache使用1位有效位)

Tag : 16 bit

Cache行数据存储体容量为: $16 \times 8 = 128$ bit

有效位: 1 bit

Cache 一行的总容量= $(16+1+ 128)$ bit = 145 bit

Cache 总容量为 = 4096×145 bit = 580Kbit

7.8 Cache和调度算法应用举例

例3 某机内存为16MB, Cache容量16KB, 每块8个字, 每个字32位, 采用四路组相联。

- 1) 求该组相联映射的主存地址字段位数;
- 2) 设Cache初始状态为空, 若CPU顺序访问0-99号单元, 并从中读出100个字, CPU每次读一个字, 并重复此顺序10次, 计算Cache命中率?
- 3) 若Cache的速度是主存速度的6倍, 求存储系统访问的加速比?

解:

Tag	Index	块内偏移
-----	-------	------

 \rightarrow 求解有顺序

解: 1) 每块 8×32 位 = 32B \rightarrow $\left\{ \begin{array}{l} \text{块内偏移地址: 5位} \\ \text{Cache行数: } \rightarrow 16\text{KB}/32\text{B} = 512 \text{行} \end{array} \right.$

Cache的组数为: $\rightarrow 512/4 = 128$ 组 \rightarrow Index : 字段 7位

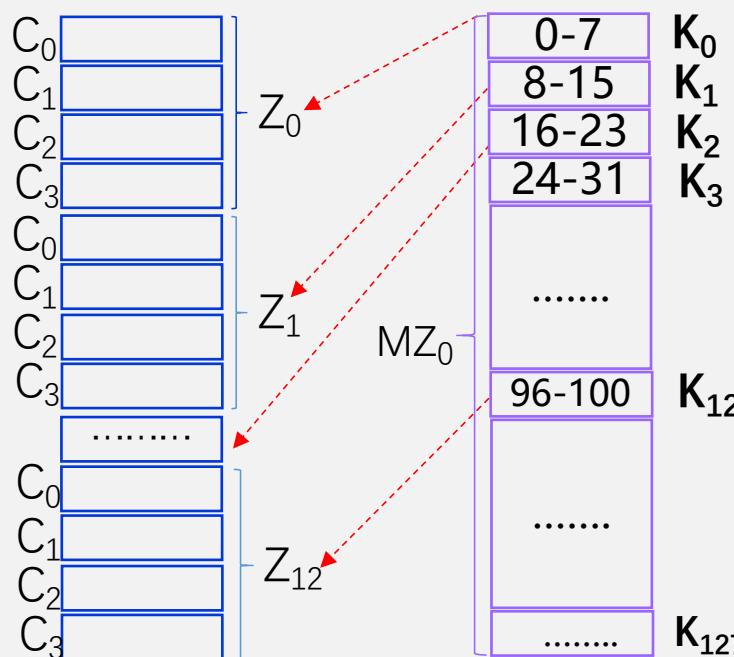
Tag : $24 - 5 - 7 = 12$ 位

7.8 Cache和调度算法应用举例

例3 某机内存为16MB,Cache容量16KB,每块8个字,每个字32位,采用四路组相联。

2)设Cache初始状态为空,若CPU顺序访问0-99号单元,并从中读出100个字,CPU每次读一个字,并重复此顺序10次,计算cache命中率?

组相联关键: 主存根据Cache的组数再分组



- Cache有 $512/4=128$ 组, 主存每组有128块, 主存前100个单元分13块, 都处于主存0组, 故访问主存前100号单元不发生页面调度

- 初态为空,每块第一个字不命中, 后7个字访问均命中

- 100号单元对应13块(编号从0-12),重复10次访问中的第一轮访问共有13次不命中,后9轮访问均命中

- 循环10次的总命中率为: $(100*10-13)/(10*100) = 98.7\%$

7.8 Cache和调度算法应用举例

例3 某机内存为16MB,Cache容量16KB,每块8个字,每个字32位,采用四路组相联。

3)若Cache的速度是主存速度的6倍,求存储系统访问的加速比?

设Cache的存取周期为t, 则主存存取周期为6t

100字直接从内存读取10次, 所需总时间为:

$$T_{nc} = 100 * 10 * 6t = 6000t$$

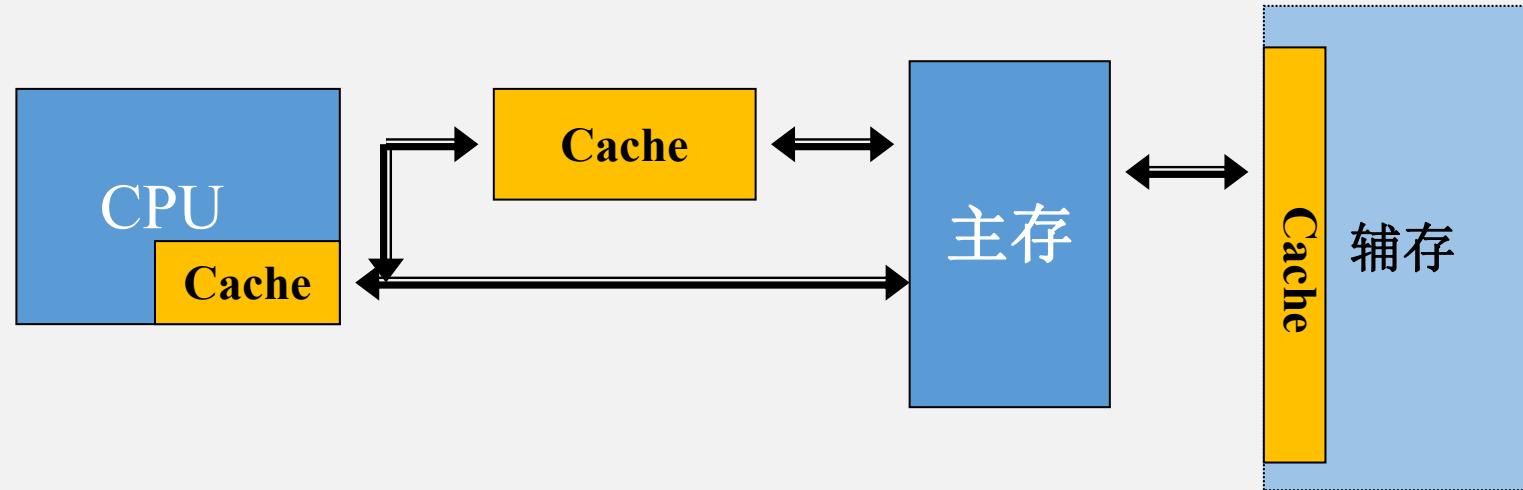
通过高速缓存体系访问1000个数据的时间为:

$$T_c = 13 * 6t + (1000 - 13) * t = 1065t$$

存储系统访问的加速比

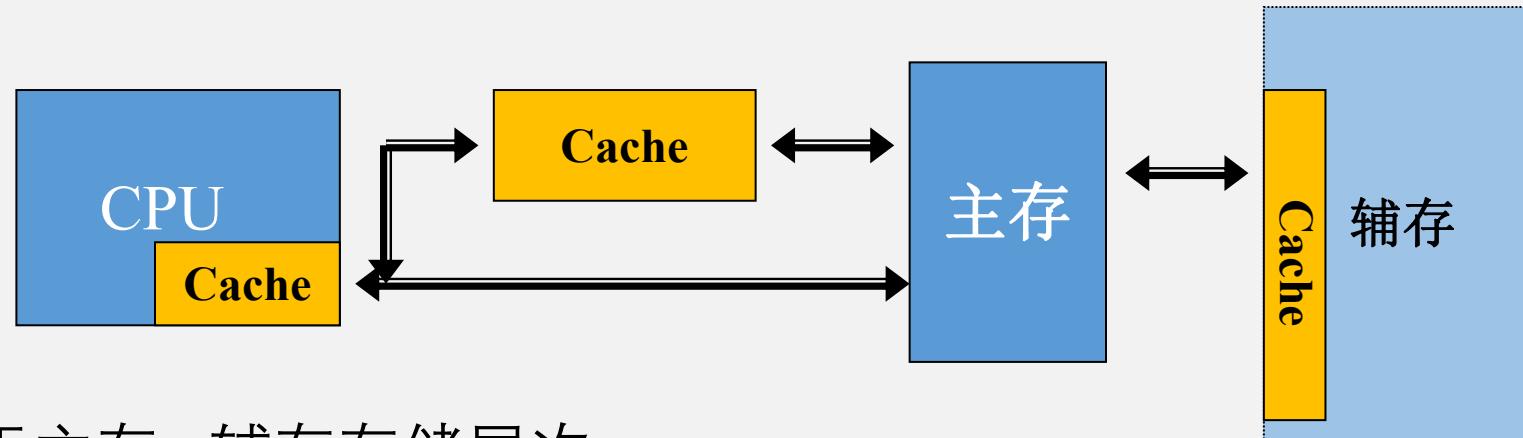
$$S = T_c / T_{nc} = (6000t / 1065t) = 5.6$$

1. 虚拟存储器概述



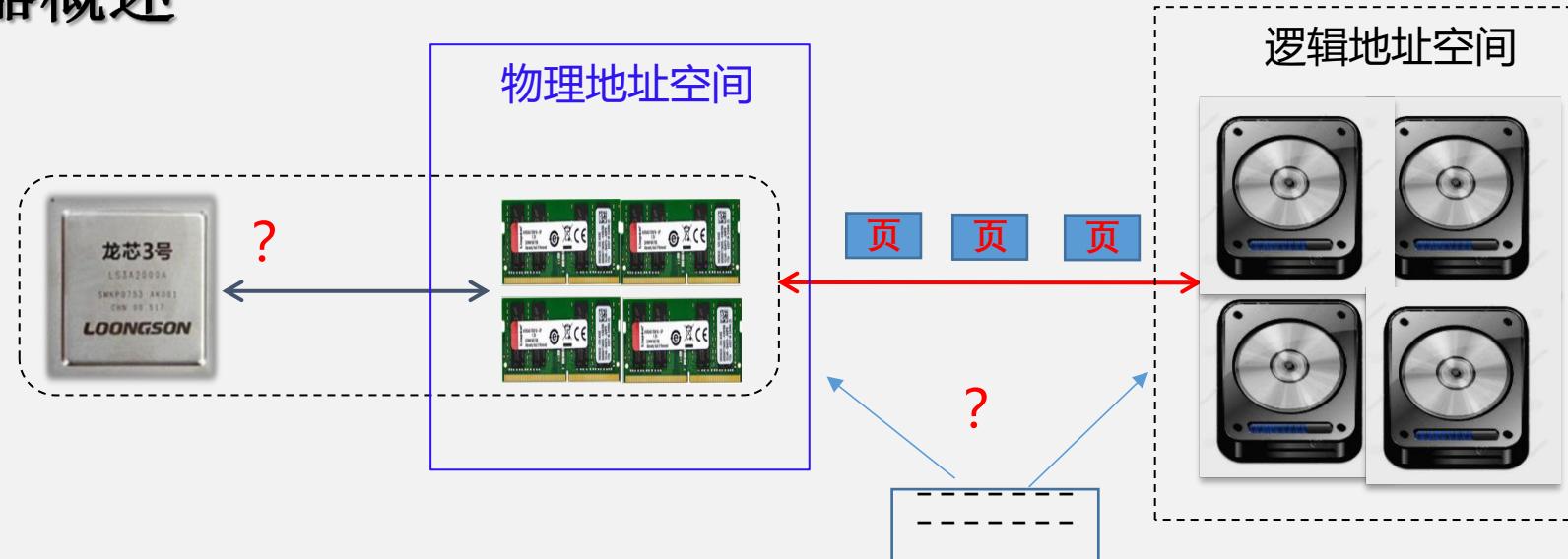
- ◆ 冯诺依曼计算机工作原理：存储程序、程序控制
- ◆ 计算机能执行比主存空间大的程序吗？
- ◆ 1961年，基尔伯恩(Tom Kilbrn)等提出虚拟存储器的概念；
- ◆ 比尔·盖茨(1981)“无论对谁来说,640K内存都足够了”

1. 虚拟存储器概述



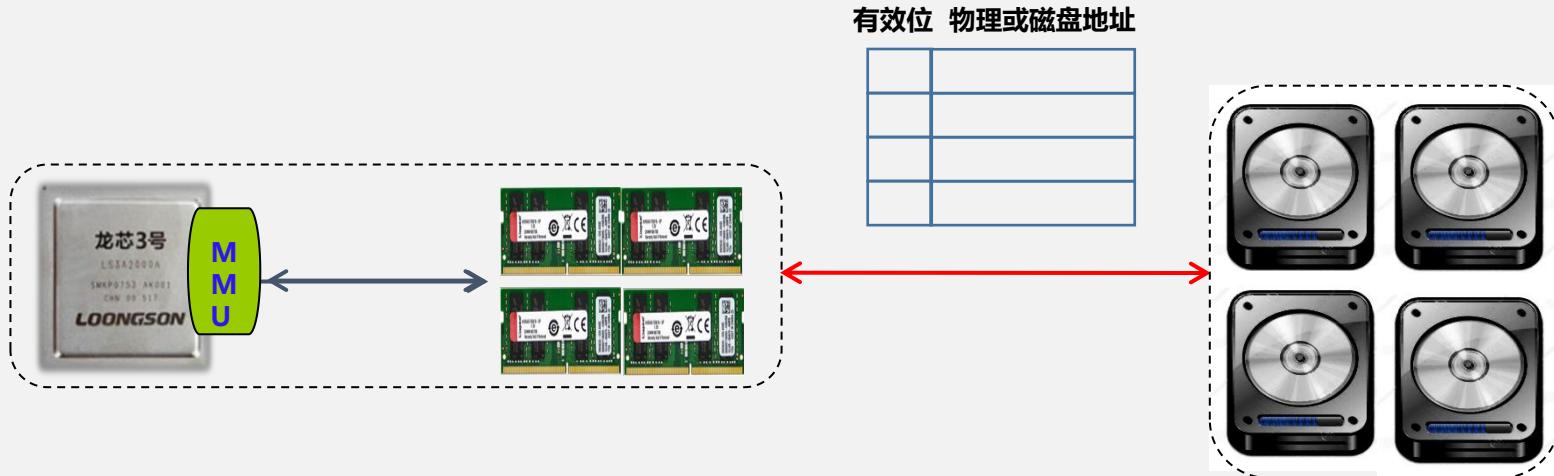
- ◆ 处于主存 - 辅存存储层次
- ◆ 解决主存容量不足问题，向程序员提供比主存空间大的编程空间，即程序员看到的内存空间(最大值与什么有关？)；
- ◆ 虚存空间由辅存（磁盘）来支持；
- ◆ 分类：页式虚拟存储器、段式虚拟存储器、段页式虚拟存储器

1. 虚拟存储器概述



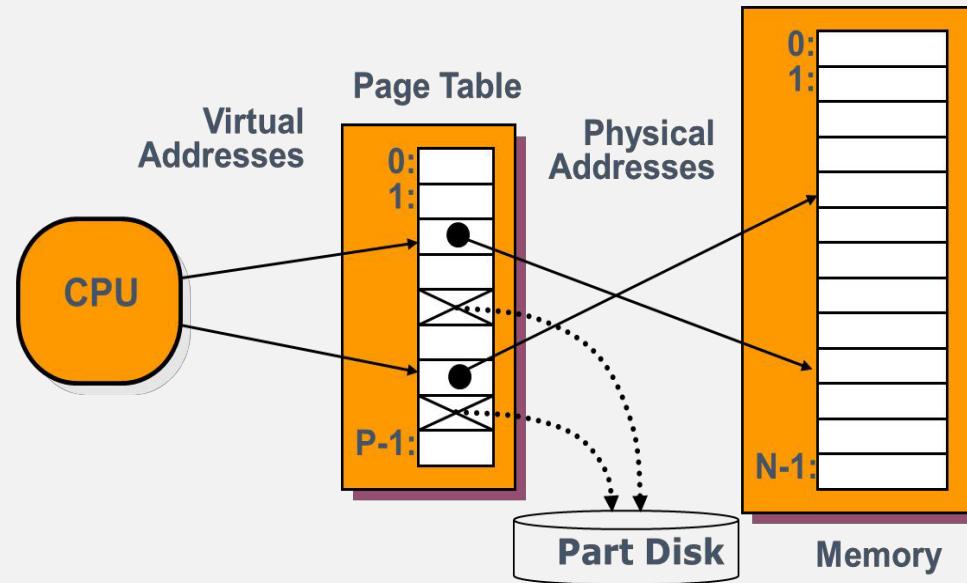
- ◆ 虚拟内存和物理内存分成大小相等的页，页内地址连续；
- ◆ 虚拟页动态调入物理内存空间； \curvearrowright Physical Address Virtual address
- ◆ CPU访问存储系统的地址属性(PA or VA)? 直面物理空间却要使用逻辑地址
- ◆ 如何判断CPU要访问的信息是否在主存中?

2. 页式虚拟存储器地址变换

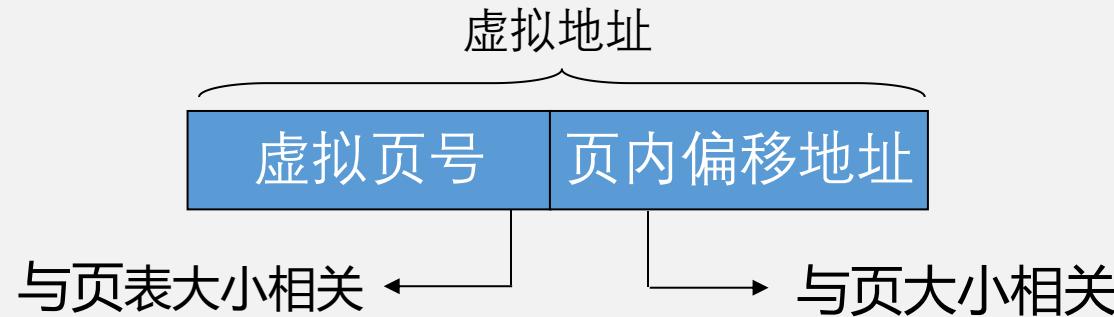


- ◆ 采用**MMU**(Memory Management Unit):管理虚拟存储器与物理存储器
- ◆ 采用**页表**判断CPU要访问的内容是否在主存，并与MMU配合实现逻辑地址与物理地址之间的转换。

2. 页式虚拟存储器地址变换



2. 页式虚拟存储器地址变换



假定主存页大小为4K, 虚存大小为4GB, 则:

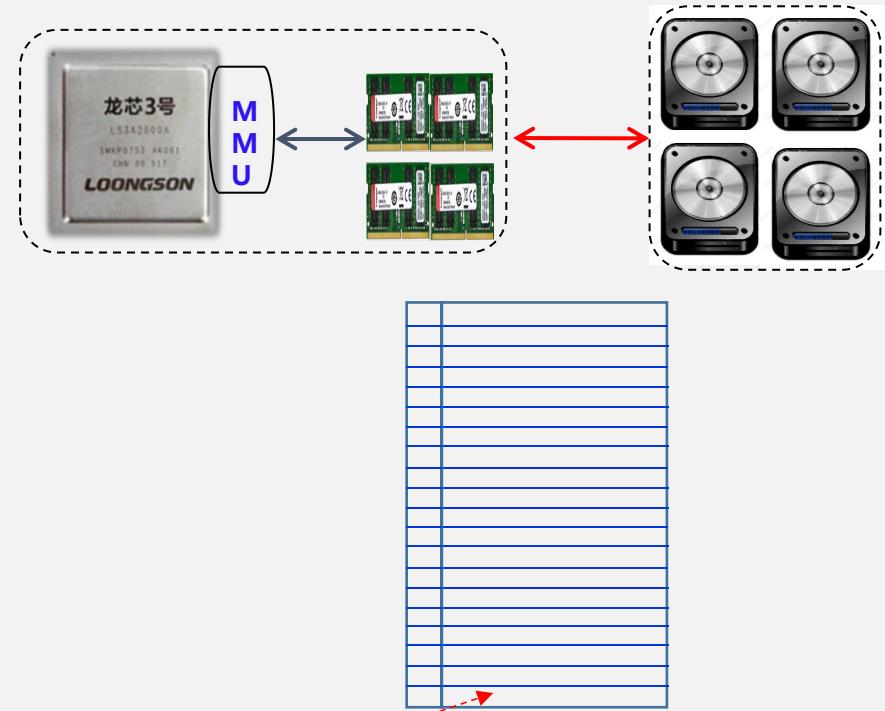
页内偏移量为: 12位;

虚拟页号: $32 - 12 = 20$ 位, 对应页表有 1024×1024 项(行),

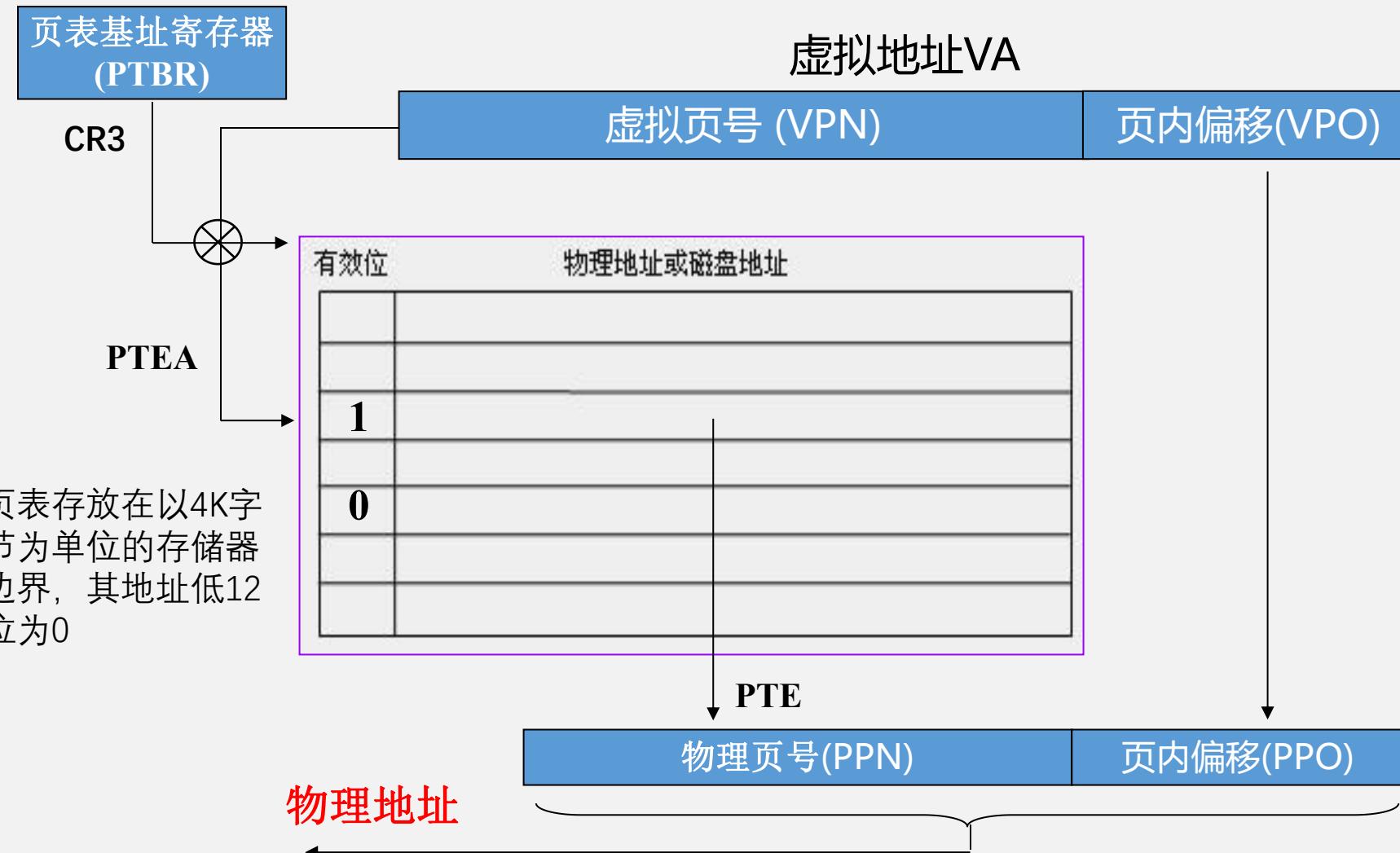
每行表示对应虚页是否调入主存及其对应的物理页号;

该页表共有 1024×1024 个页表项PTE(Page Table Entry)

页表存放在哪里?



2. 页式虚拟存储器地址变换



2. 页式虚拟存储器地址变换

在下表所示页式虚拟存储器页表中，页面大小为1024B，求对应于虚拟地址 $(2050)_{10}$ 和 $(3080)_{10}$ 的物理地址。(假定最大物理空间为64KB)

0	1	000010
1	1	000110
2	1	000111
3	0	000100

页表

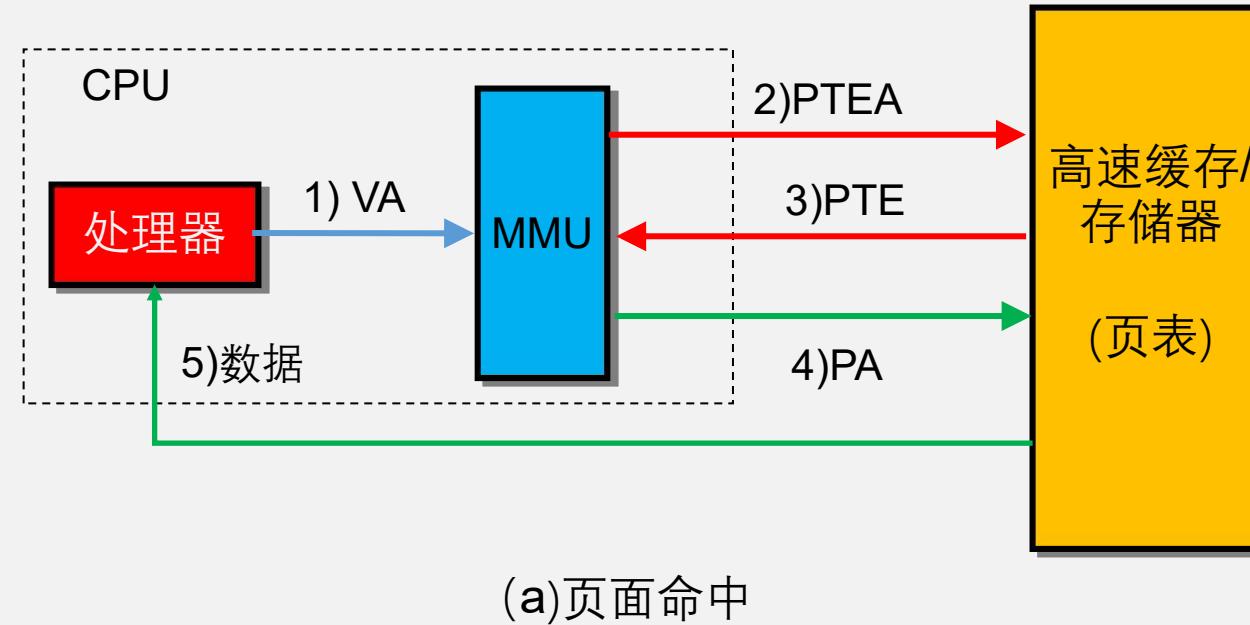
$$(2050)_{10} = 2048 + 2 = (10\ 0000000010)_2$$

虚页号为2,查页表可得物理页号为000111且有效位为1，则对应物理地址为:

000111	0000000010
--------	------------

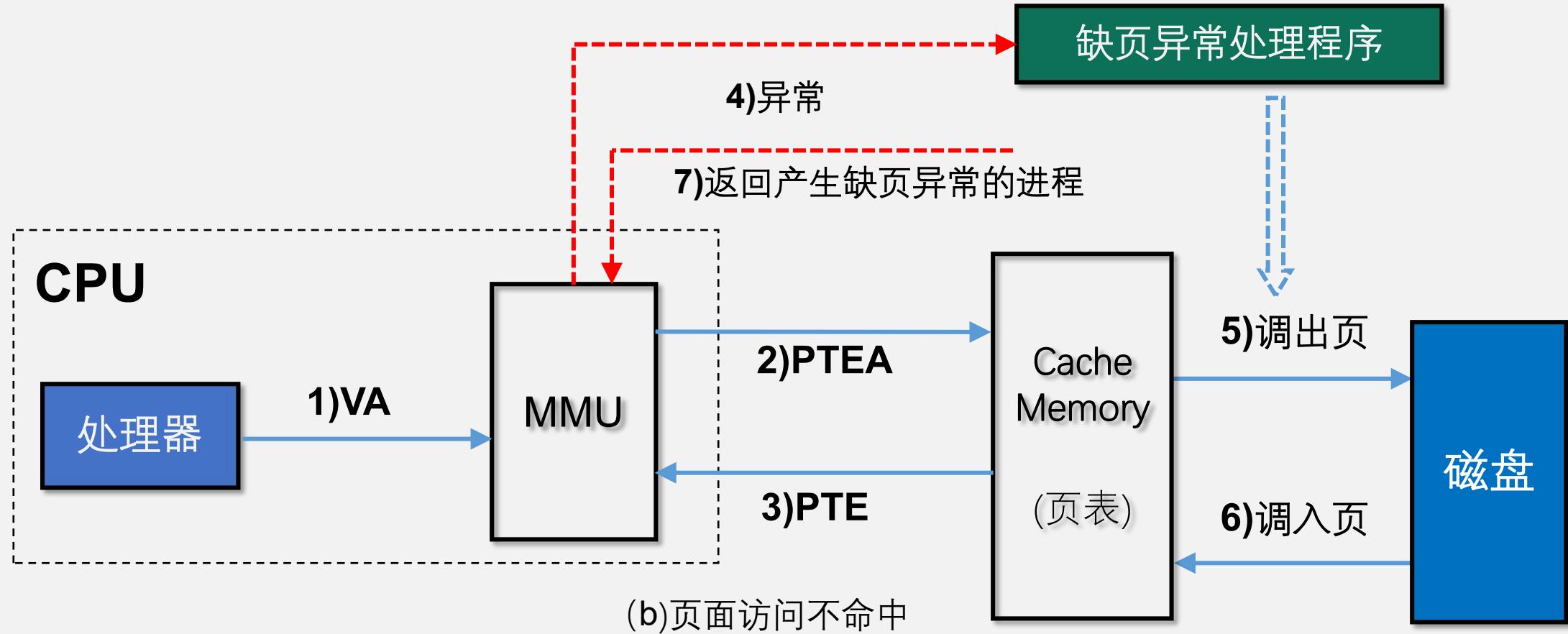
虚存3080号单元对应的物理地址为: **缺失, 虚页号3不在主存中**

3. 旁路转换缓存(TLB:Translation Lookaside Buffer)



虚实地址转换访问Cache/主存(2次)

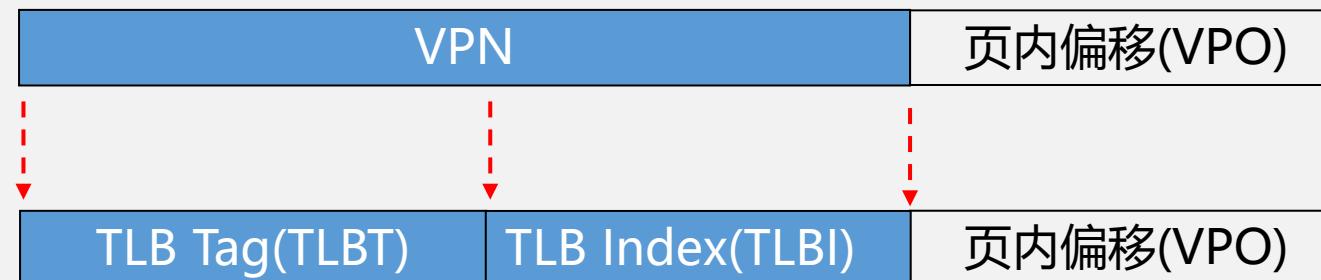
3. 旁路转换缓存(TLB:Translation Lookaside Buffer)



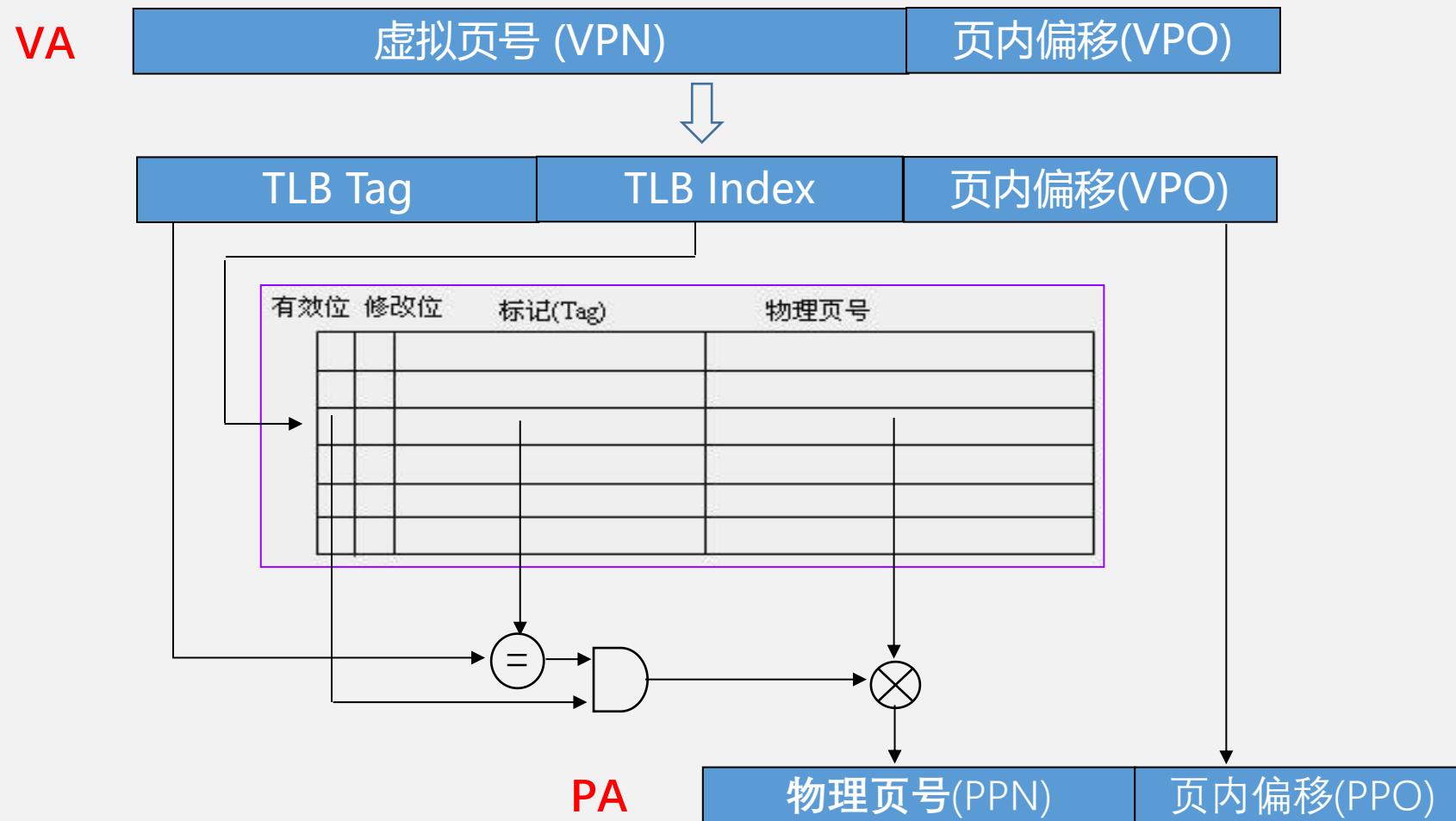
共访问Cache/主存多少次?

3. 旁路转换缓存(TLB:Translation Lookaside Buffer)

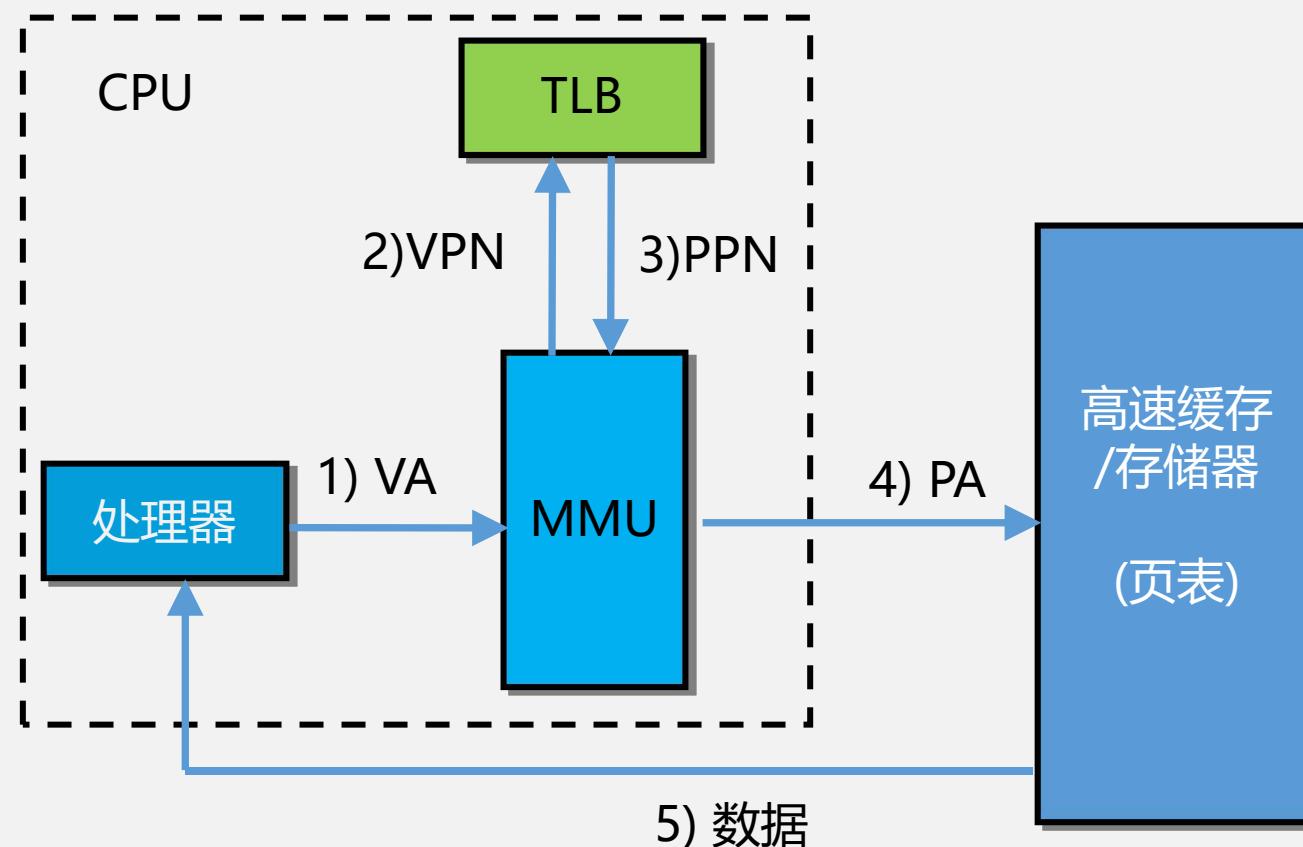
- 根据局部性原理，可增加一个小容量、高速存储部件存放当前访问页表地址变换条目，该存储部件称为TLB: 旁路转换缓存。
- TLB类似页表，也是PTE的集合。为实现对TLB的快速访问，类似Cache中映射方法，对来自于CPU的虚页号进行逻辑划分，得到相应的标记和索引字段。



3. 旁路转换缓存(TLB: Translation Lookaside Buffer)

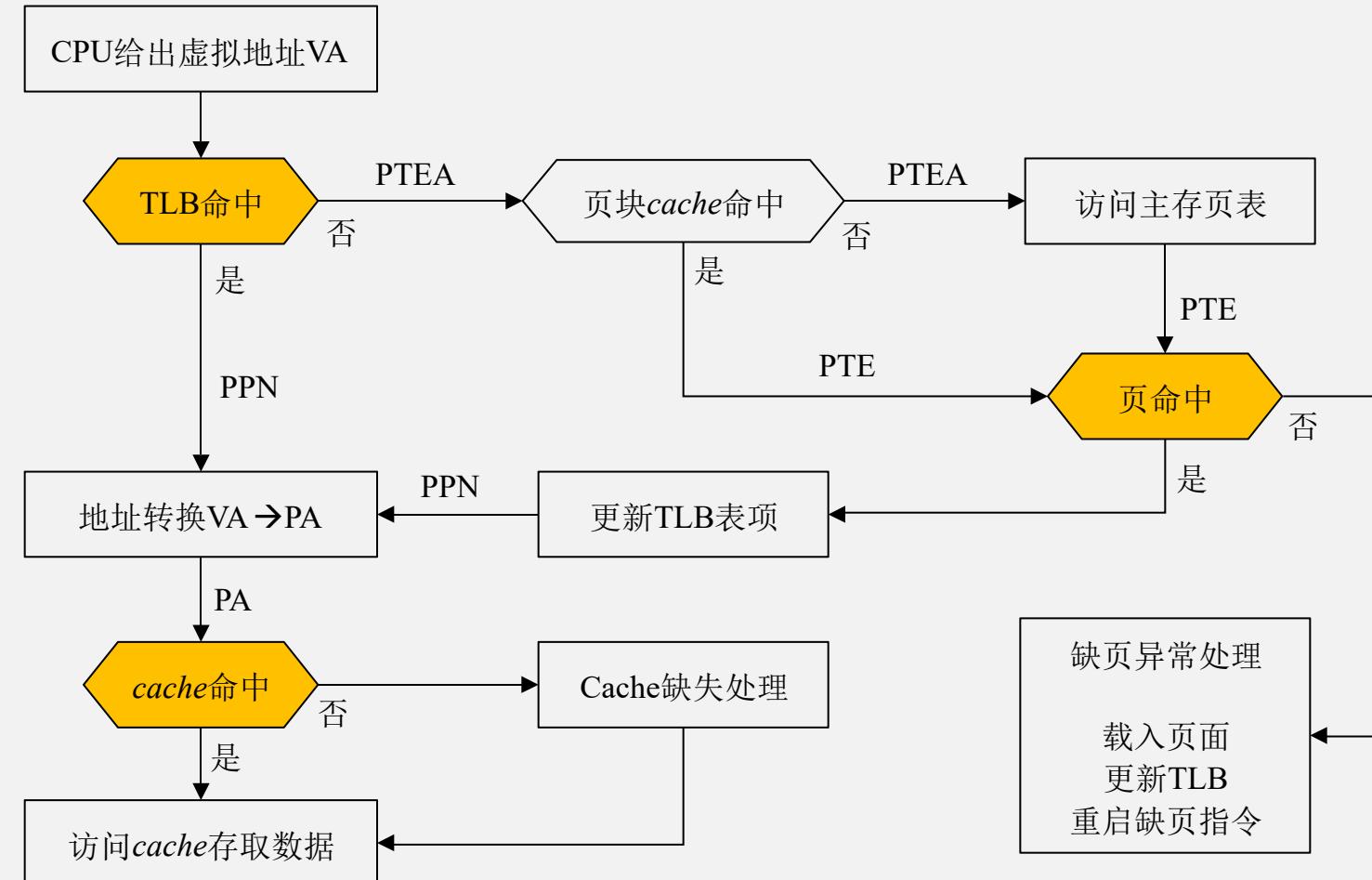


3. 旁路转换缓存(TLB: Translation Lookaside Buffer)



访问过程中存在更新TLB中局部页表的问题

3. 旁路转换缓存(TLB: Translation Lookaside Buffer)



包含TLB的虚拟地址转换物理地址流程

3. 旁路转换缓存 (TLB: Translation Lookaside Buffer)

◆ 包含Cache和TLB的存储层次访问

存储器系统页大小为4K,地址空间为32位,假设物理地址长度与虚拟地址相同,TLB包含64个表项,采用全相联.主存块大小为4B. Cache采用直接映像, 数据区容量为64KB,包含1位有效位.

$b=12$

$a=20$

$c=20$

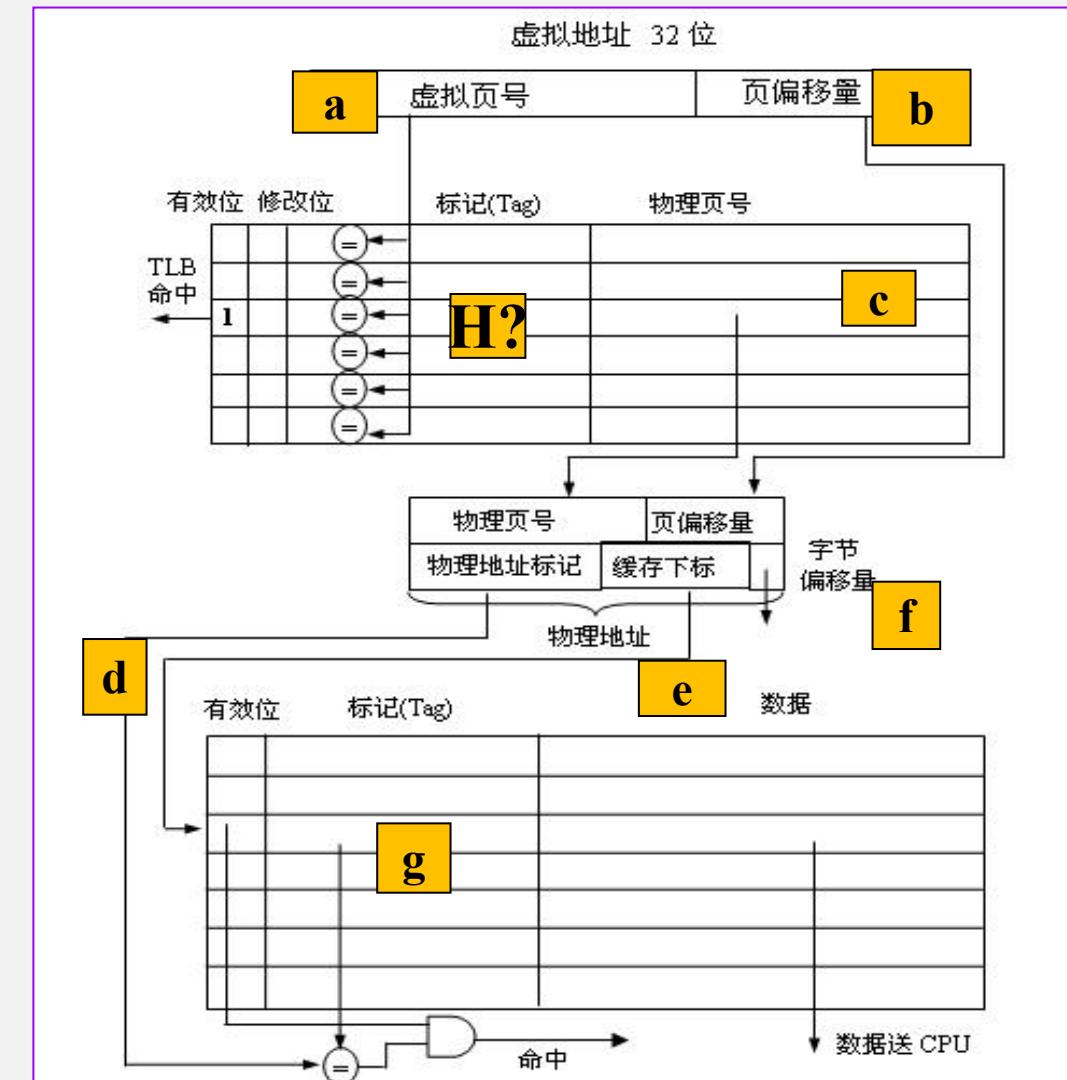
$f=2$

$e=14$

$d=16$

$g=16$

$H=20$



第三部分完