第3章 存储系统结构

3.1 地址映像和变换

在一台计算机中，通常有多种存储器

种类：主存储器、Cache、通用寄存器、缓冲存储器、磁盘存储器、磁带存储器、光盘存储器等

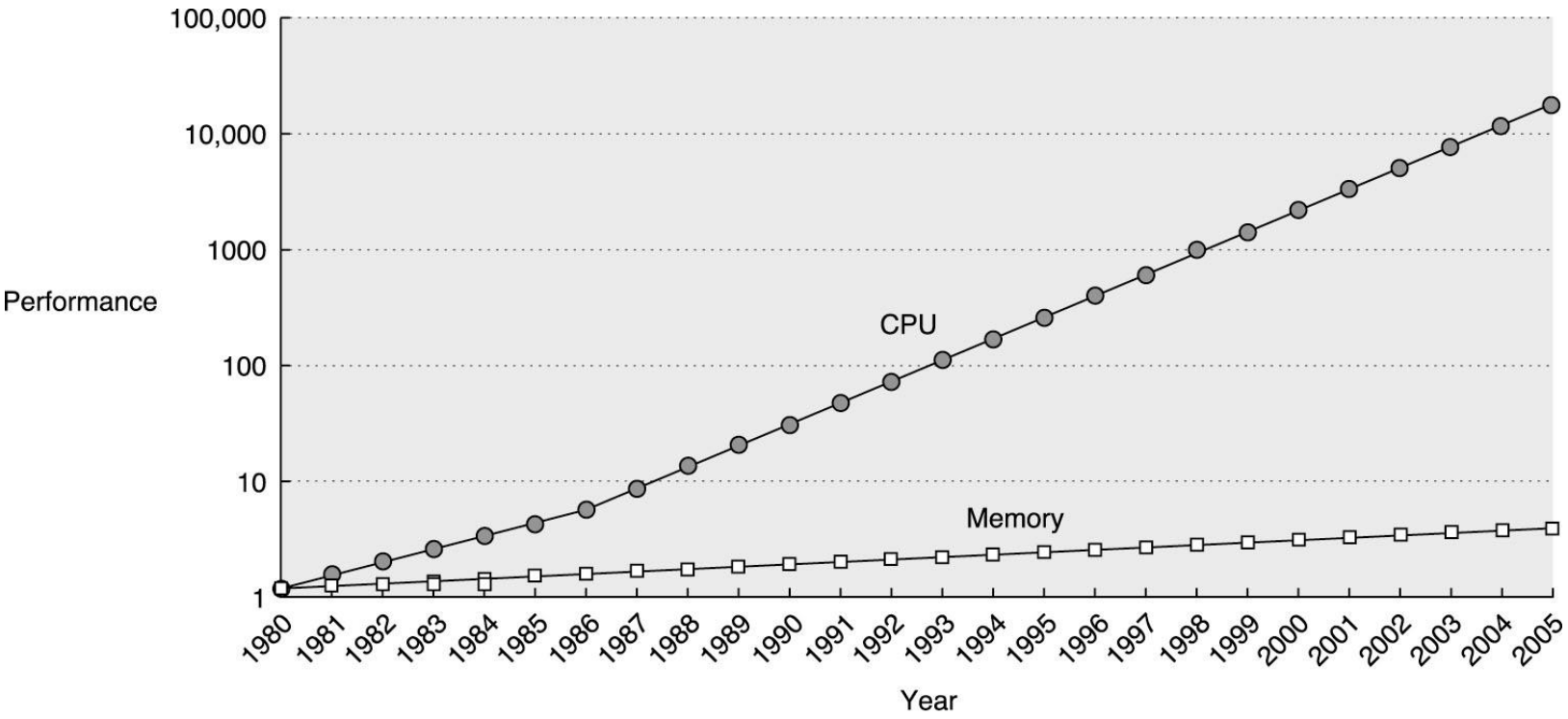
访问方式：随机访问、直接译码、先进先出、相联访问、 块传送、文件组

组成存储系统的关键：速度、容量、价格

把速度、容量和价格不同的多个物理存储器组织成一个存储器

速度快，存储容量大，单位容量的价格便宜

现代计算机系统以存储器为中心



- Clearly , there is a processor-memory performance gap—must try to close
- How?

1. 存储系统的层次结构

多个层次的存储器：

第1层： Register Files(寄存器堆)

第2层： Buffers(Lookahead)(先行缓冲站)

第3层： Cache(高速缓冲存储器)

第4层： Main Memory(主存储器)

第5层： Online Storage(联机存储器)

第6层： Off-line Storage(脱机存储器)

用*i*表示层数，则有：工作周期T*i*＜T*i*+1，

存储容量：*Si*＜*Si*+1，单位价格：*Ci*＞*Ci+1*

问题：存储系统的频带平衡

例：Pentium4的指令执行速度为8GIPS，CPU取指令8GW/s，访问数据16GW/s，各种输入输出设备访问存储器1GW/s，三项相加，要求存储器的频带宽度不低于25GW/s。

如果采用PC133内存，主存与CPU速度差188倍

如果采用PC266内存，主存与CPU速度差94倍

解决存储器频带平衡方法

(1)多个存储器并行工作

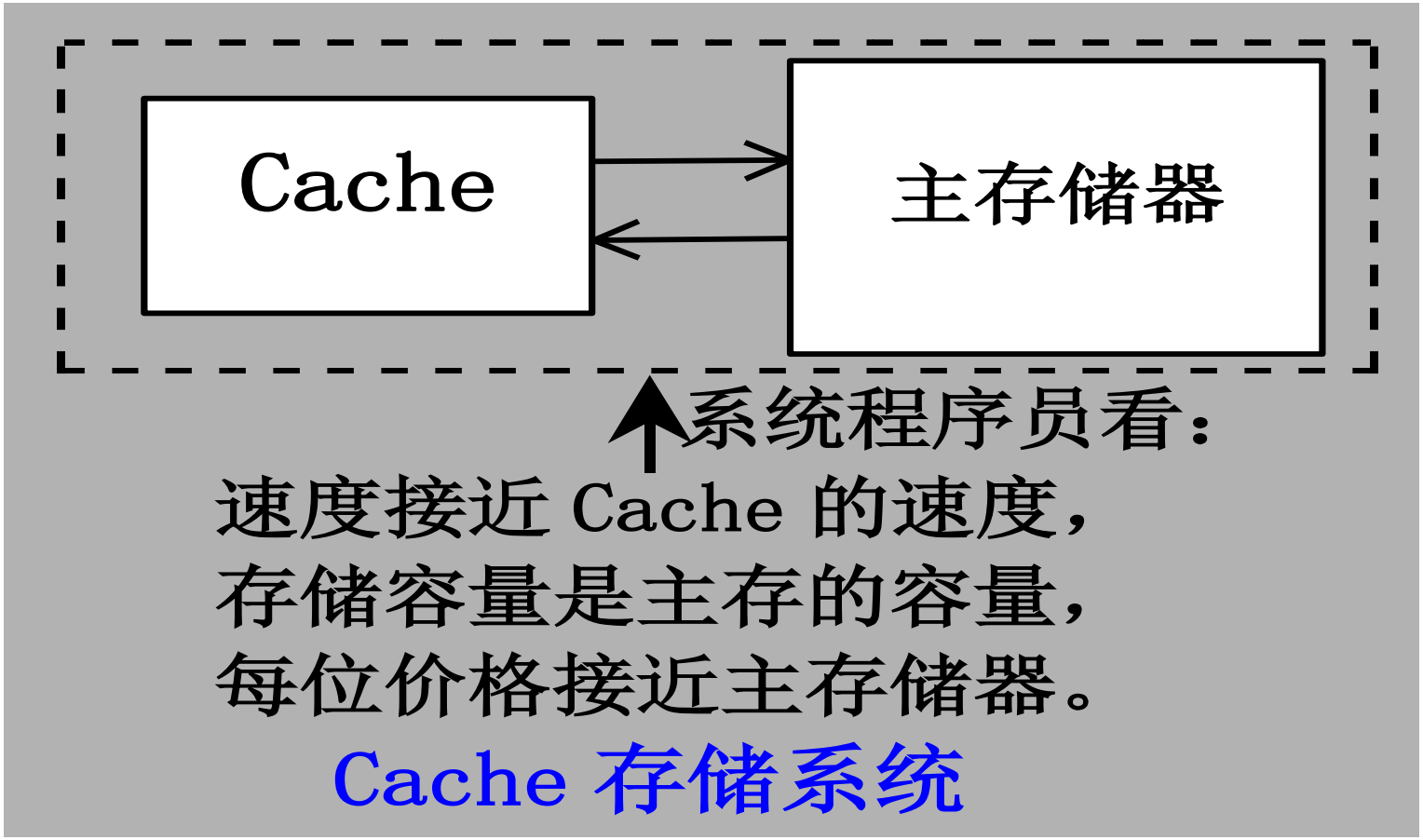
(2)设置各种缓冲存储器

(3)采用存储系统

- 在一般计算机系统中，有两种存储系统：

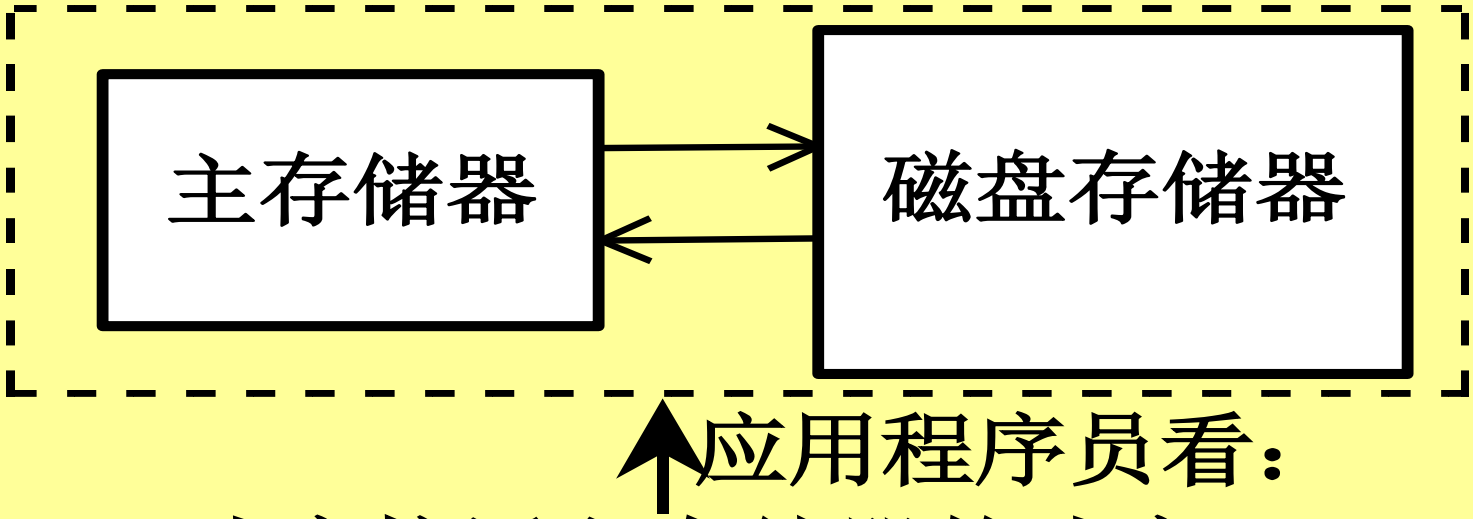
➢Cache存储系统：由Cache和主存储器构成

主要目的：提高存储器速度



➢虚拟存储系统：由主存储器和硬盘构成

主要目的：扩大存储器容量



2 地址的映象与变换

三种地址空间： 虚拟地址空间
主存储器地址空间
辅存地址空间

程序定位

如何把程序的逻辑地址变换成实际的主存物理地址
目的是提高主存空间利用率，及时释放主存中的无用区

地址映象：

把虚拟地址空间映象到主存地址空间

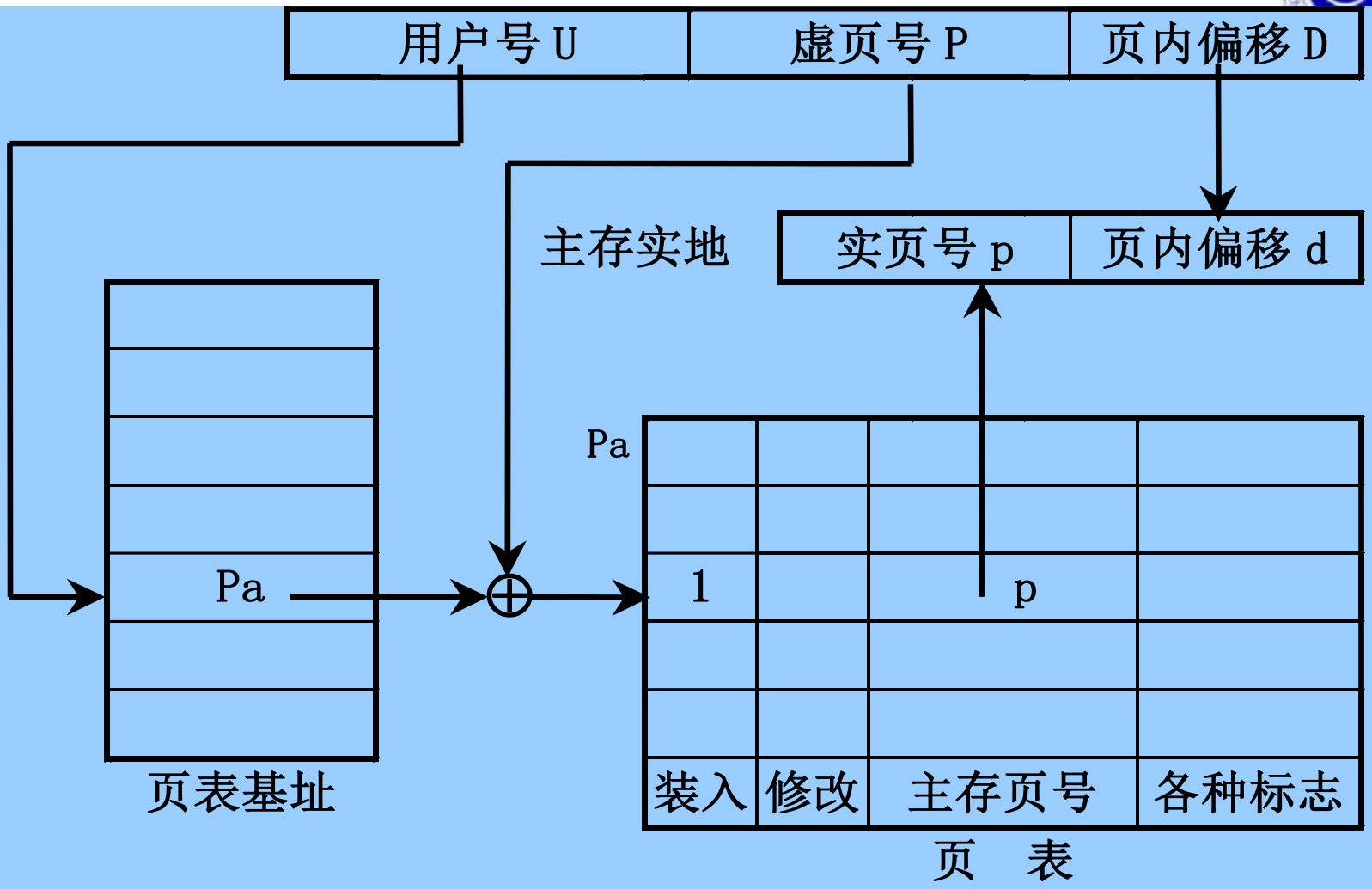
地址变换：

在程序运行时，把虚地址变换成主存实地址

地址映像方式 段式管理 页式管理 段页式管理

三种虚拟存储器： 页式虚拟存储器
段式虚拟存储器
段页式虚拟存储器

地址变换方法：页式管理



页式虚拟存储器的地址变换

主要优点：

(1)主存储器的利用率比较高

(2)页表相对比较简单

(3)地址变换的速度比较快

(4)对磁盘的管理比较容易

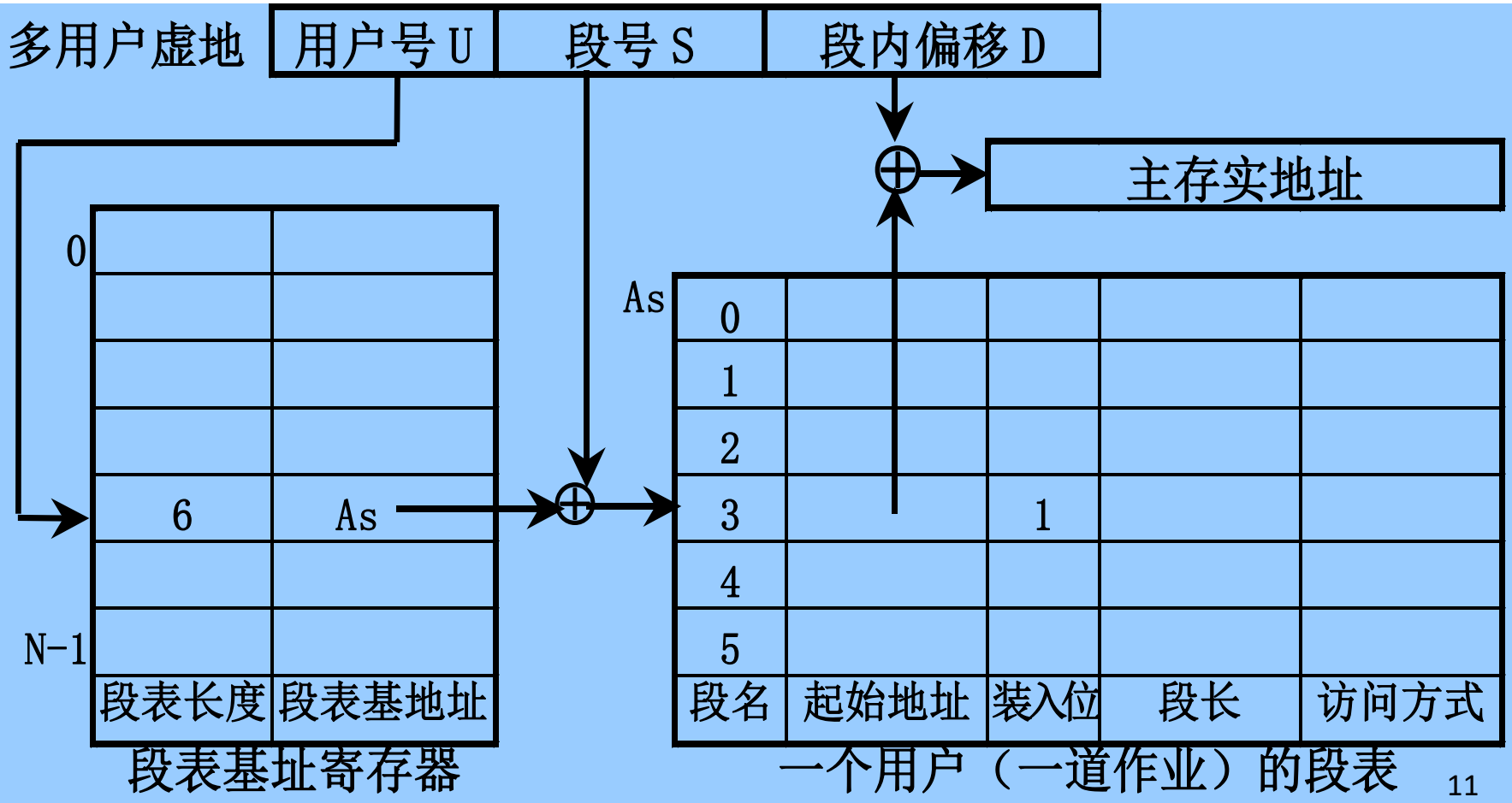
主要缺点：

(1)程序的模块化性能不好

(2)页表很长，需要占用很大的存储空间

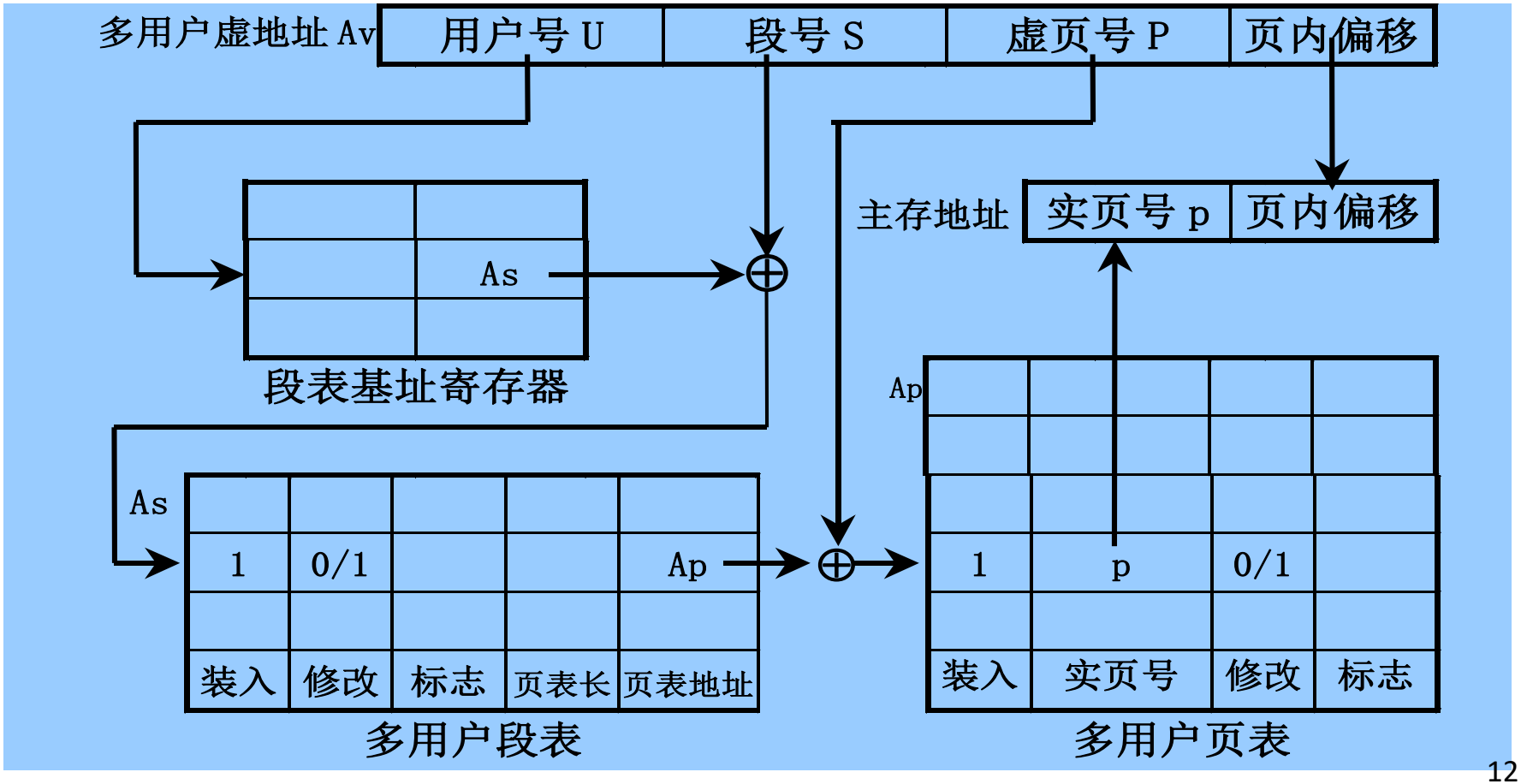
例如：虚拟存储空间4GB，页大小1KB，则页表的容量为4M字，16MB。

段式管理 地址变换方法：由用户号找到基址寄存器，读出段表起始地址，与虚地址中段号相加得到段表地址，把段表中的起始地址与段内偏移D相加就能得到主存实地址。



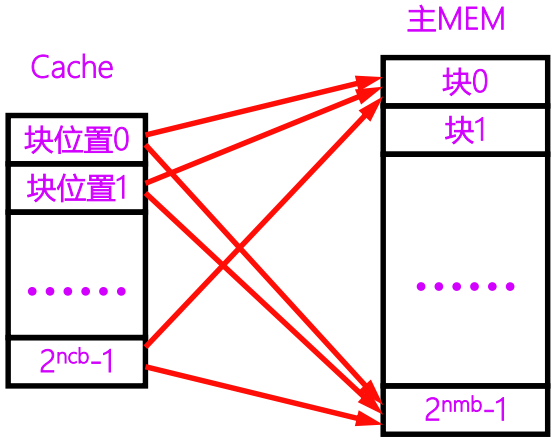
段页式管理 地址变换方法：

先查段表，得到页表起始地址和页表长度，
再查页表找到要访问的主存实页号，
把实页号p与页内偏移d拼接得到主存实地址。

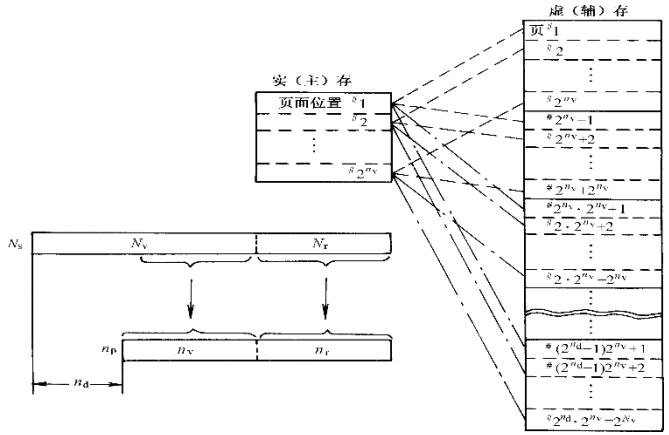


2 地址的映象与变换（续）

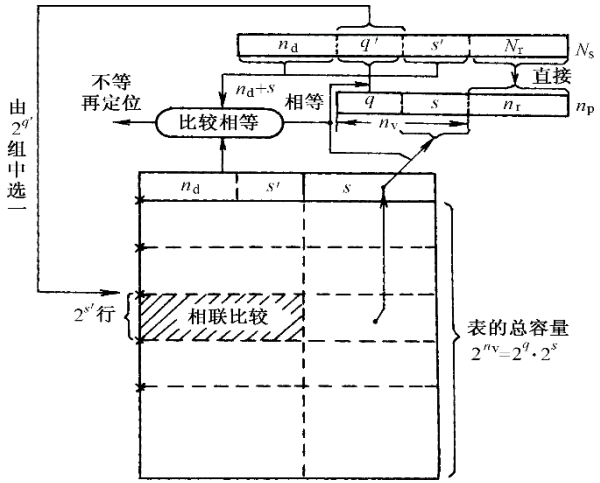
1. 全相联映像及其变换
2. 直接映像及其变换
3. 组相联映像及其变换
4. 段相联映像及其变换



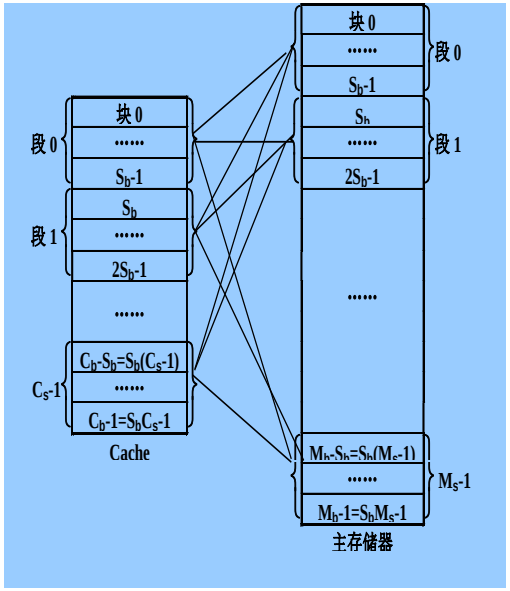
主存中任一块可映像到Cache内一块位置上



直接映像及其变换



组相联映像及其地址变换



段相联映像及其变换

3.2 页面替换算法及其实现

1.**替换算法**：主存（实存）所有页面已经全部被占用而又出现页面失效时，按照哪种算法（规则）来替换主存中某页。

需要注意的是考虑替换算法时首先要明确的是采用的相联映像

采用什么替换算法：

是否有利于提高存储体系的性能

是否提高命中率

是否易于实现

2．页面替换发生时间：

当发生页面失效时，要从磁盘中调入一页到主存

如果主存储器的所有页面都已经被占用，必须从主存储器中淘汰掉一个不常使用的页面，以便腾出主存空间来存放新调入的页面。

3．评价页面替换算法好坏的标准：

一是**命中率**要高，

二是算法要容易实现。

4.页面替换算法：

随机算法RAND 先进先出FIFO 近期最少使用LRU 优化替换OPT

替换算法分析（1）

时间 t	1	2	3	4	5	6	7	8	9	10	11	12	实际命中次数
页地址流	P1	P2	P3	P4	P1	P2	P5	P1	P2	P3	P4	P5	
主存页面数 N=3	1	1	1*	4	4	4*	5	5	5	5	5*	5*	
		2	2	2*	1	1	1*	1*	1*	3	3	3	
			3	3	3*	2	2	2	2	2*	4	4	
	调入	调入	调入	替换	替换	替换	替换	命中	命中	替换	替换	命中	3 次
主存页面数 N=4	1	1	1	1	1*	1*	5	5	5	5*	4	4	
		2	2	2	2	2*	2*	1	1	1	1*	5	
			3	3	3	3	3	3*	2	2	2	2*	
				4	4	4	4	4	4*	3	3	3	
	调入	调入	调入	调入	命中	命中	替换	替换	替换	替换	替换	替换	2 次

FIFO 算法在主存页面数增加时命中率反而下降

替换算法分析（2）

- 设有一个程序，共5页，页地址流如图3-16上部所示；分配给该程序的实存有3页；
- 现分别用FIFO，LRU，OPT替换算法推算这3页的使用情况，其中按所用算法选中为替换页的用星号（*）标记。
- FIFO命中3次，命中率 H=3/12=0.25
- LRU命中5次， 命中率H=5/12=0.417
- OPT命中6次， 命中率H=6/12=0.50
- 分析：
- FIFO命中率最低，
- LRU命中率接近OPT

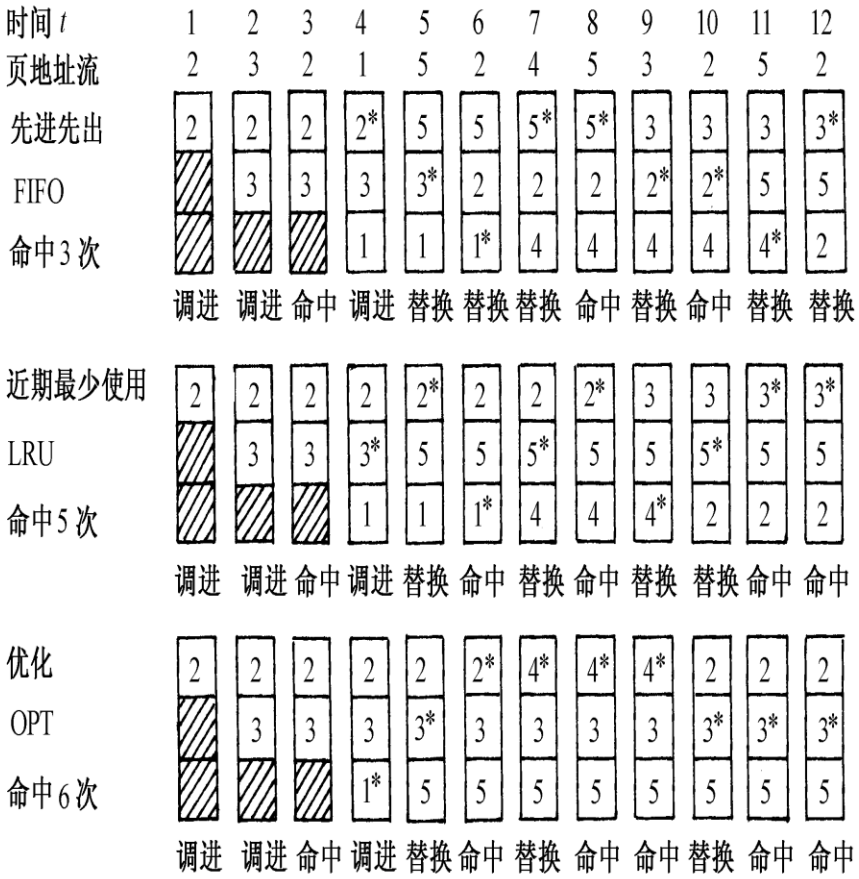


图3-16 替换算法过程

替换算法分析（3）

如图3-17所示：

若循环程序需4页，实存只有3页，

- 则LRU与FIFO命中率均为零，OPT命中3次。
- 在本例中，LRU和FIFO均连续、频繁地出现页面失效，几乎没有有效运算时间，

这种现象称为“颠簸”

- “颠簸”使系统效率显著下降，

是评价存储管理和存储体系性能的一个重要标志。

- 从图3-17可看出，如果分配给此程序的实存页数增加一页，则命中率可显著提高。

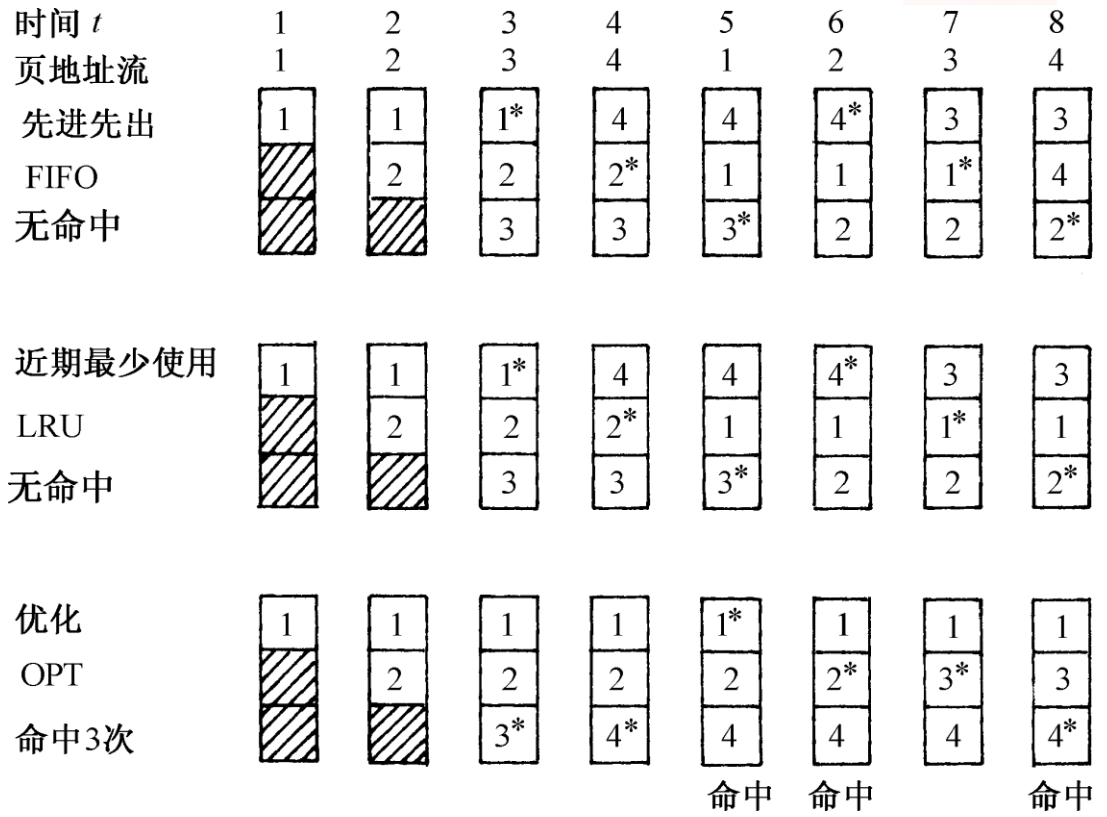


图3-17　循环程序所需页数大于实存页数的替换算法过程

提高主存命中率的方法

影响主存命中率的主要因素：

(1)程序在执行过程中的页地址流分布情况。

(2)所采用的页面替换算法。

(3)页面大小。

(4)主存储器的容量

(5)所采用的页面调度算法

FIFO 随机算法 LRU OPT

替换算法的分析

- **堆栈型替换算法**

定义：设A是长度为L的任何页地址流，t为已处理过t1个页面的时间点，n为分配给该地址流的实存页数，Bt(n)表示在t时间点n页实存的页面集，Lt表示到t时已遇到过的地址流中相异页的页数，如果替换算法具有下列包含性质：

$B_t(n) \in B_t(n+1)$，$n<L_t$

$B_t(n)=B_t(n+1)$，$n \geq L_t$

则此算法为堆栈型。

- 由于LRU算法在实存中保留的是n个最近使用过的页面，包含在n+1个最近使用过的页面里，
- 因此LRU是堆栈型算法，OPT也是堆栈型算法，而FIFO不是堆栈型算法。
- 堆栈型算法研究影响命中率的一个主要因素 — 分配给程序的主存页面数的增加与命中率的关系。
- 堆栈型算法的基本特点是： 随着分配给程序的主存页面数增加，主存的命中率也提高，至少不下降。

LRU替换算法的实现

1.如何对每次访问进行记录

2.如何根据记录判定（找出）近期最久未使用或

1、使用位法

基本思想：

- 给每个实页设一个使用位（访问位）。
- 开始时，使用位为全‘0’。
- 某页被访问（读/写）→该页使用位置‘1’。
- 页面失效时，找使用位=‘0’的页替换。

最早使用过的页替换**具体实现：**

*.在主存中建立页面表：

实页号	占用位	程序号	段页号	使用位	程序优先位	H3	其他
0							
1							
2							

*.每页按页号0、1、2、……、n 顺次占用页表(即替换表)中一行。

*.占用位: “1”→该页占用(即有信息在内)　“0”→该页空。

替换过程 页表使用位均为“0” →程序逐页调入；被占用的实有页：占用位=1

硬件定时 ⊿t→扫描页表使用位
- 使用位=0→H_S+1
- 使用位=1
  - H_S=0
  - 使用位=0

使用位：反映⊿t内使用状况

H_S：反映m个⊿t内使用状况

LRU替换算法的实现

2.堆栈法

基本思想:

*.栈顶为最近访问过页，

栈底为近期最久未访问过页号即为替换对象。

*.查找某页，经相联比较

→未找到 →该页压入栈顶 →栈底页出栈;

→找到 →该页压入栈顶 →该页以上各页下移一个位置。

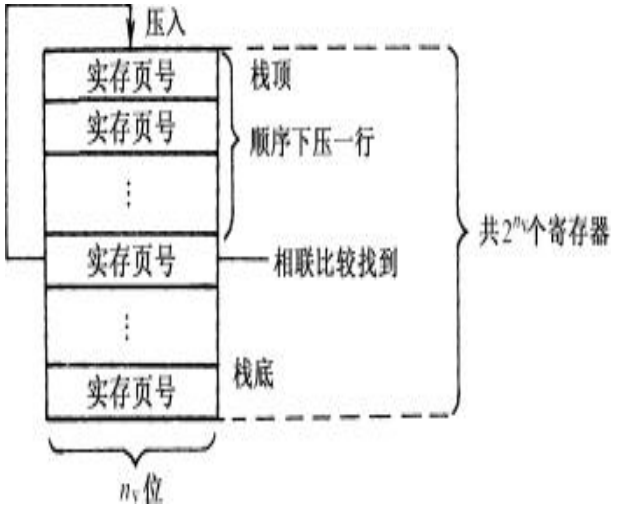


图3-19　用堆栈法实现LRU替换算法

3.3 并行主存系统

并行主存系统频宽分析

现代计算机以存储器为中心工作。

一般存储器访问源4个：

取指令、取操作数、写结果、I/O数据。

存储器访问速度跟不上系统要求，将影响系统性能。

例如：

速度为200MIPS计算机，各种访问源频带：取指：200MW/s（设每条指令长度为1字W。CPU取操作数和保存结果400MW/s各种输入输出设备：5MW/s

这样系统要求存储器频带宽度不低于605MW/s，则访问周期不大于16.5ns；

实际：主存DRAM的工作周期 200ns左右

解决存储器频带问题方法

1 多个存储器并行工作

2 设置各种缓冲存储器

3 采用存储系统，特别是Cache存储系统

并行主存系统频宽分析(续)

方法：把m字w位的存储器改变成为m/n字n×w位的存储器

逻辑实现：把地址码分成两个部分，一部分作为存储器的地址另一部分负责选择数据

主要缺点：访问冲突大

(1)取指令冲突

(2)读操作数冲突

(3)写数据冲突

(4)读写冲突

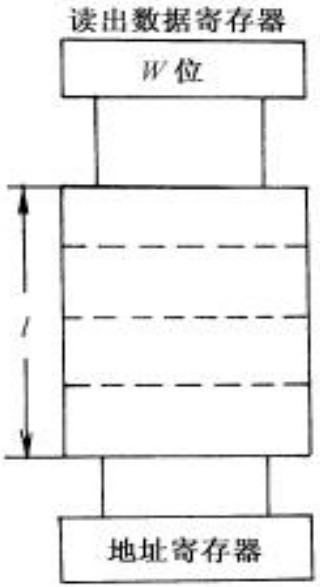


图3-20　单体单字存储器

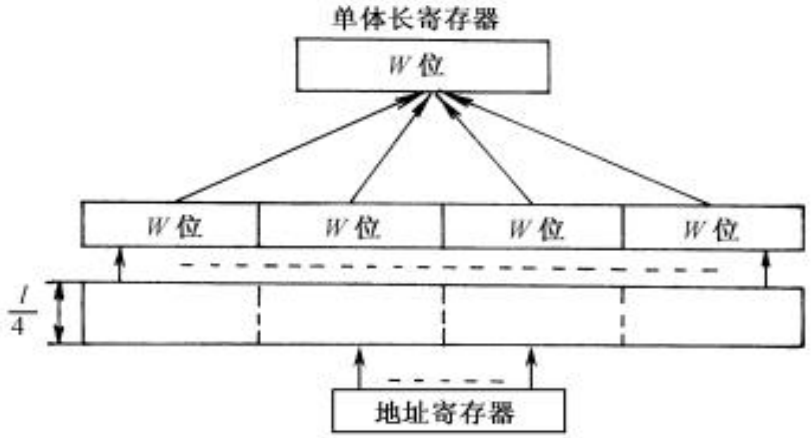


图3-21 单体多字（m=4）存储器

理论上：速率是单体单字存储器的4倍

单体多字存储器

两种组成方式：

单体多字方式和多体并行方式。

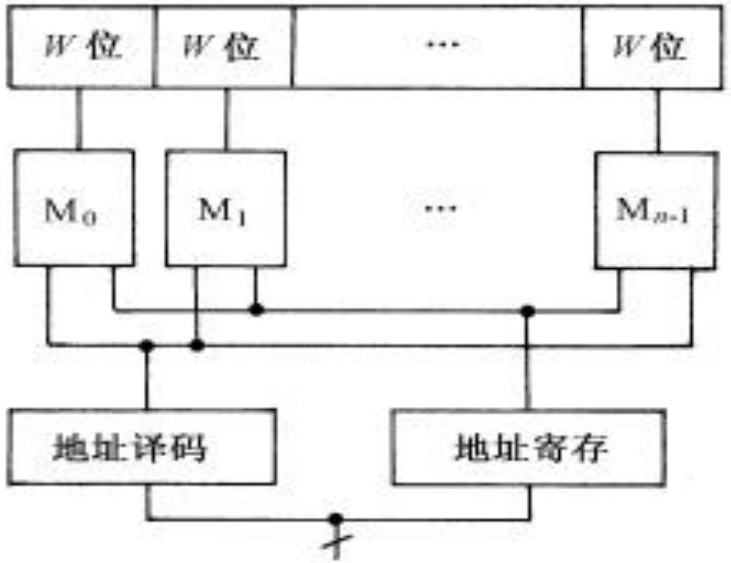


图3-22　单体方式

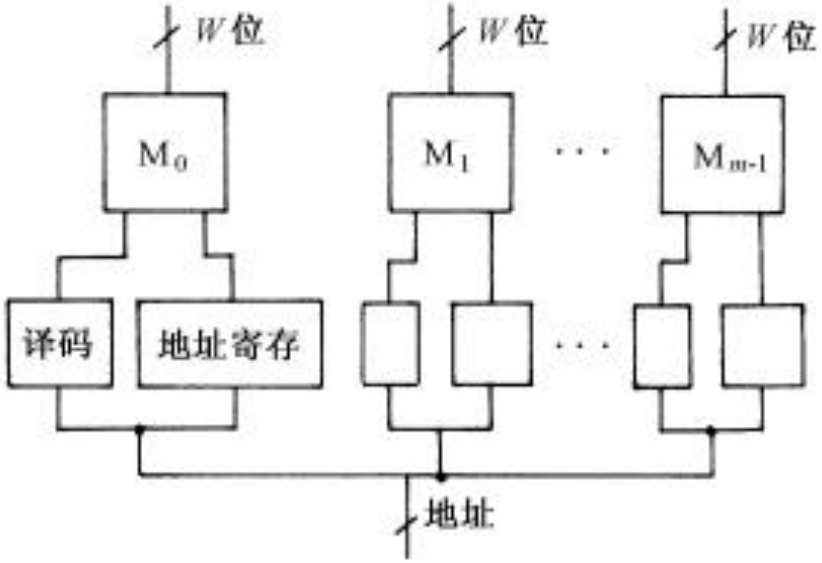


图3-23 多体方式

多体交叉存储器

1．多体交叉编址

同体相邻差4

多体并行主存系统和
单体并行主存系统最
大频宽可以相同，区别？
程序常有转移，速度？
同一存储体访问冲突？

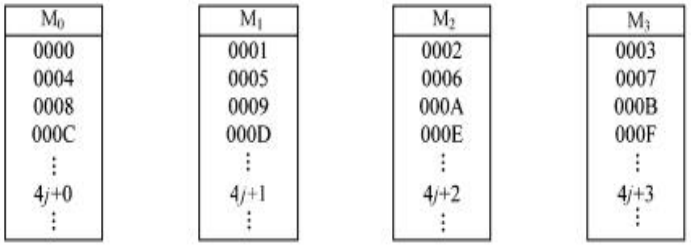


图3-24　4个分体的编址序列

地址线（最后2位 A1 A0 片选）：　A1A0：　00　01　10　11

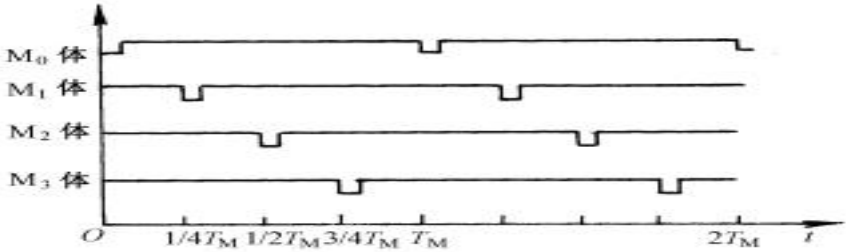


图3-25　4体并行分时工作

2．多体交叉存储器分时工作原理

分体分时启动

即每隔1/4存储周期启动一个分体

3．多体交叉存储器组成

大容量半导体MEM

由许多相同容量的MEM芯片组成，
所以每个单元编址多模m交叉编址。

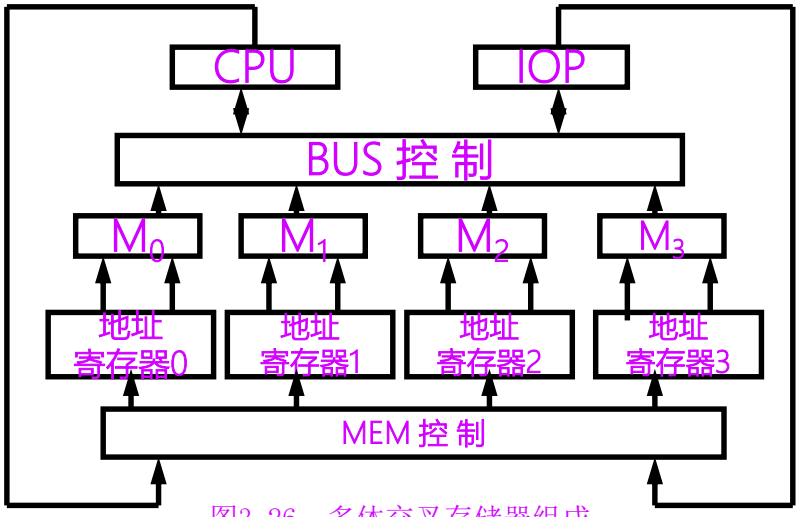
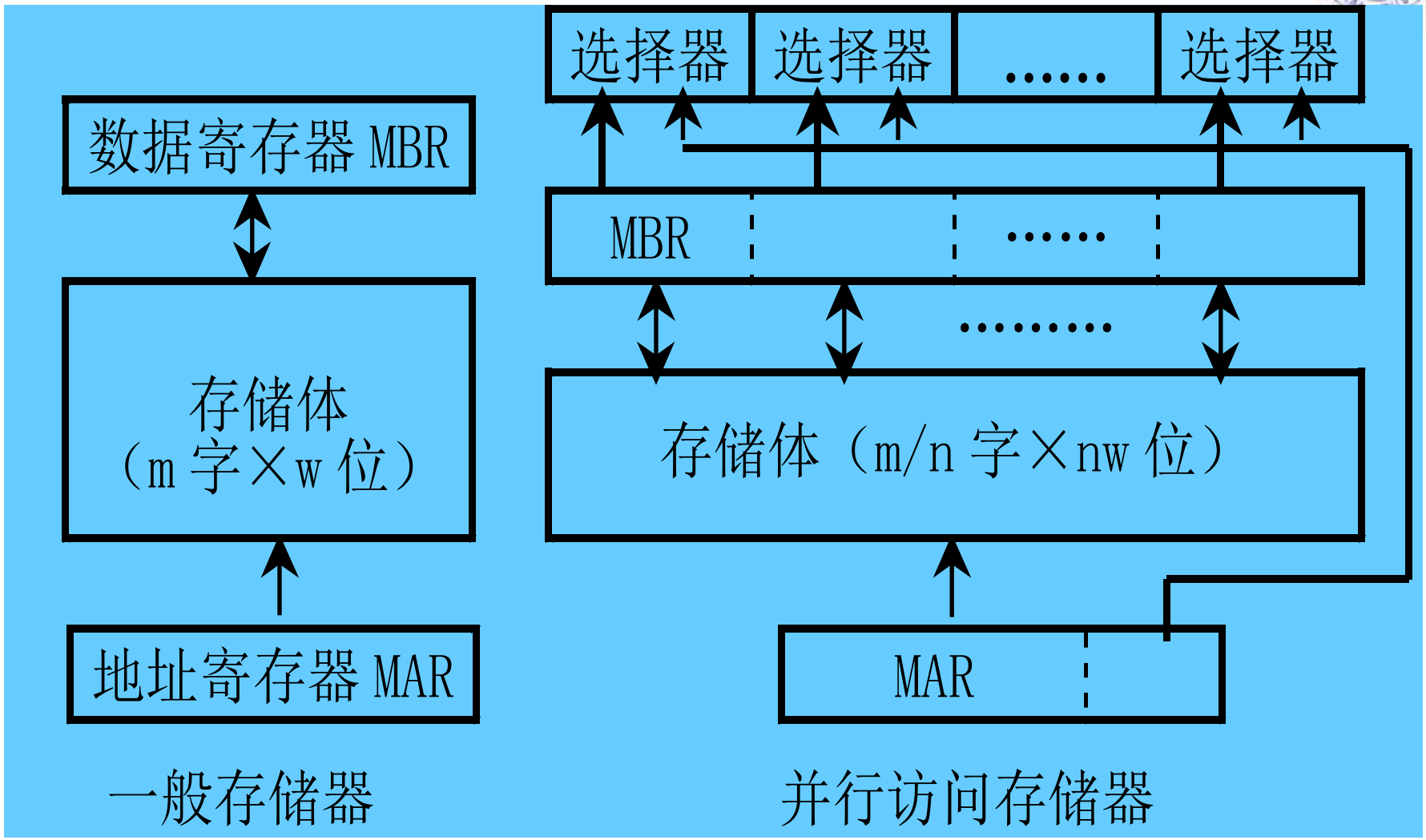


图3-26　多体交叉存储器组成

- **并行访问存储器结构框图**



交叉访问存储器

1. 高位交叉访问存储器

主要目的：扩大存储器容量

实现方法：用地址码的高位部分区分存储体号

参数计算方法:

m：每个存储体的容量，

n：总共的存储体个数，

j：存储体的体内地址，j＝0，1，2，…，m-1

k：存储体的体号，k＝0，1，2，…，n-1

存储器的地址：$A＝m×k＋j$

存储器的体内地址：$A_j＝A \bmod m$。

存储器的体号：$A_k＝\left\lfloor \frac{A}{m} \right\rfloor$

地址空间的划分和访问周期的控制

1．地址空间的划分

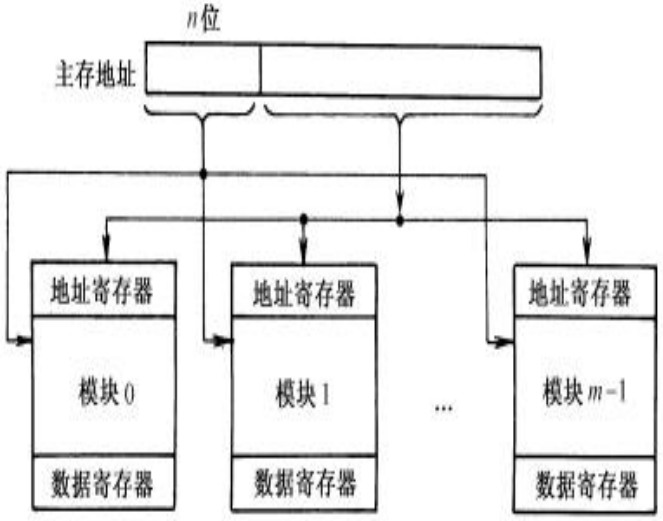


图3-27　并行存储器按高位地址划分模块
适合：程序和数据放在不同模块
缺点：不能明显提高主存的速率

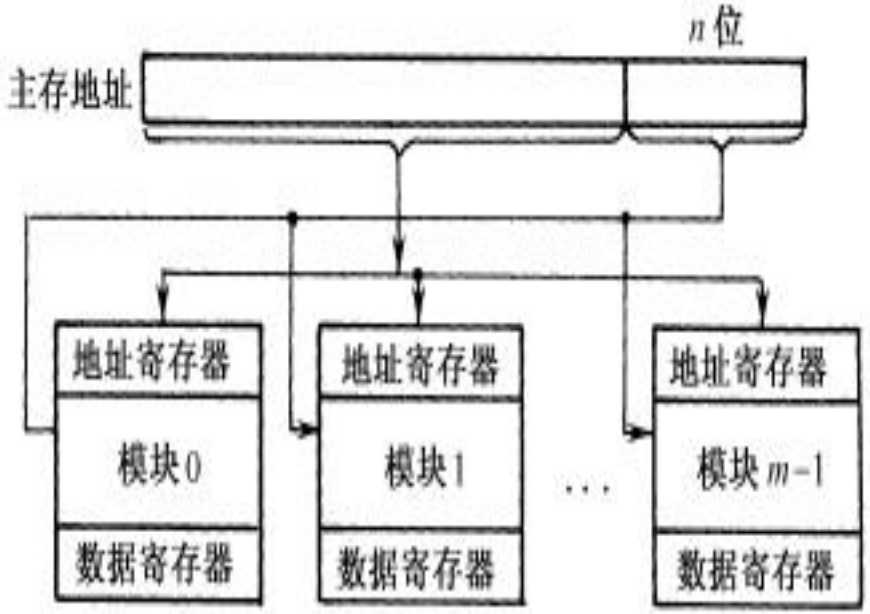


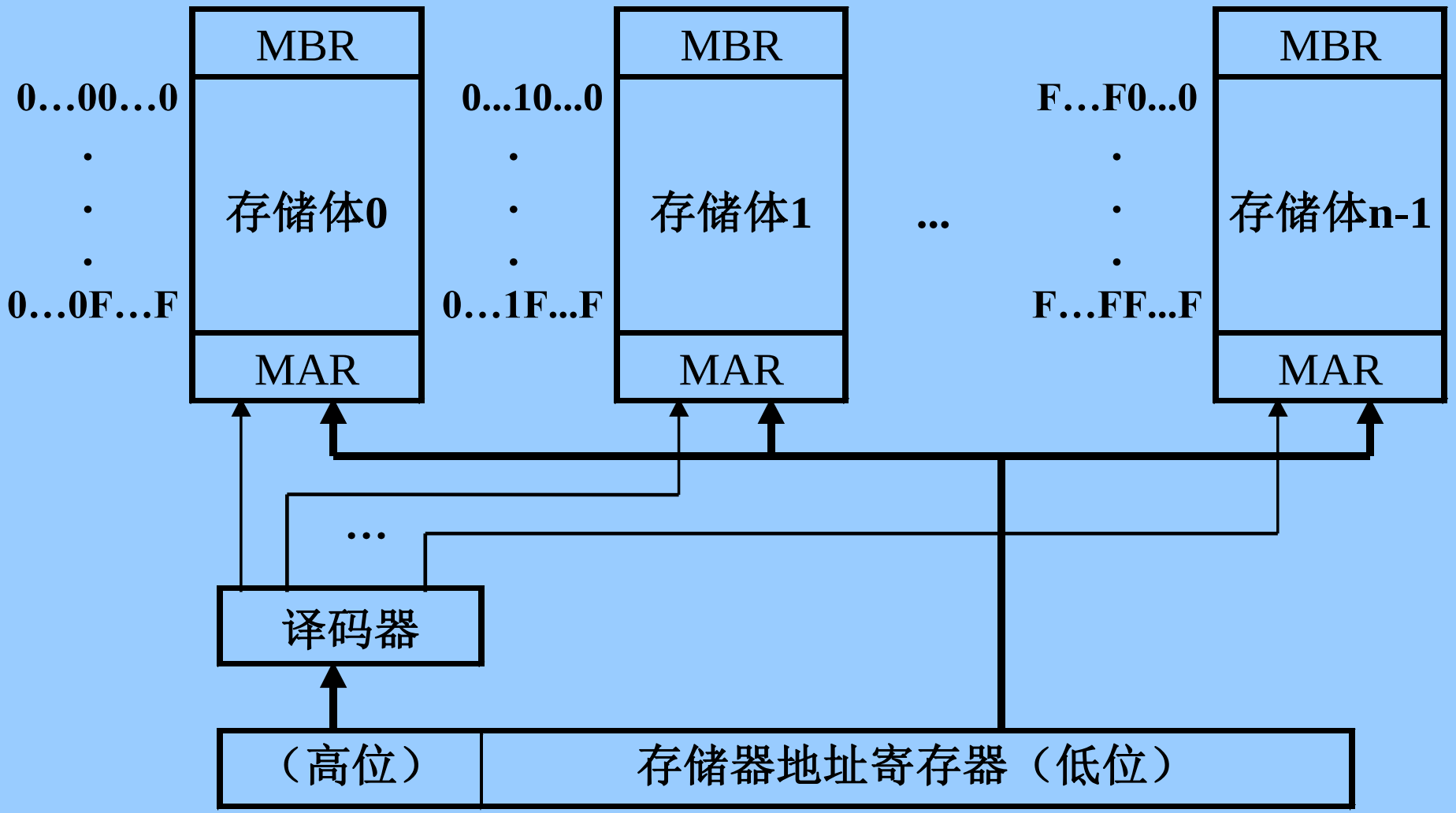
图3-28　并行存储器按低位地址划分模块
优点：连续地址访问时，速度快
缺点：容错性差；
　　地址不均匀分布，速度降低

2.访问周期的控制

同时访问　交叉开关互联网络
交叉访问　分时总线互连方式

容错：将高位与低位交叉存取加以组合，产生许多不同的交叉存取存储器组织，就可以较好地实现“容错”和兼顾提高速度

• 高位交叉访问存储器结构框图



2. 低位交叉访问存储器

主要目的：提高存储器访问速度

实现方法：用地址码的低位部分区分存储体号

参数计算：

m：每个存储体的容量，

n：总共的存储体个数，

j：存储体的体内地址，j＝0，1，2，…，m-1

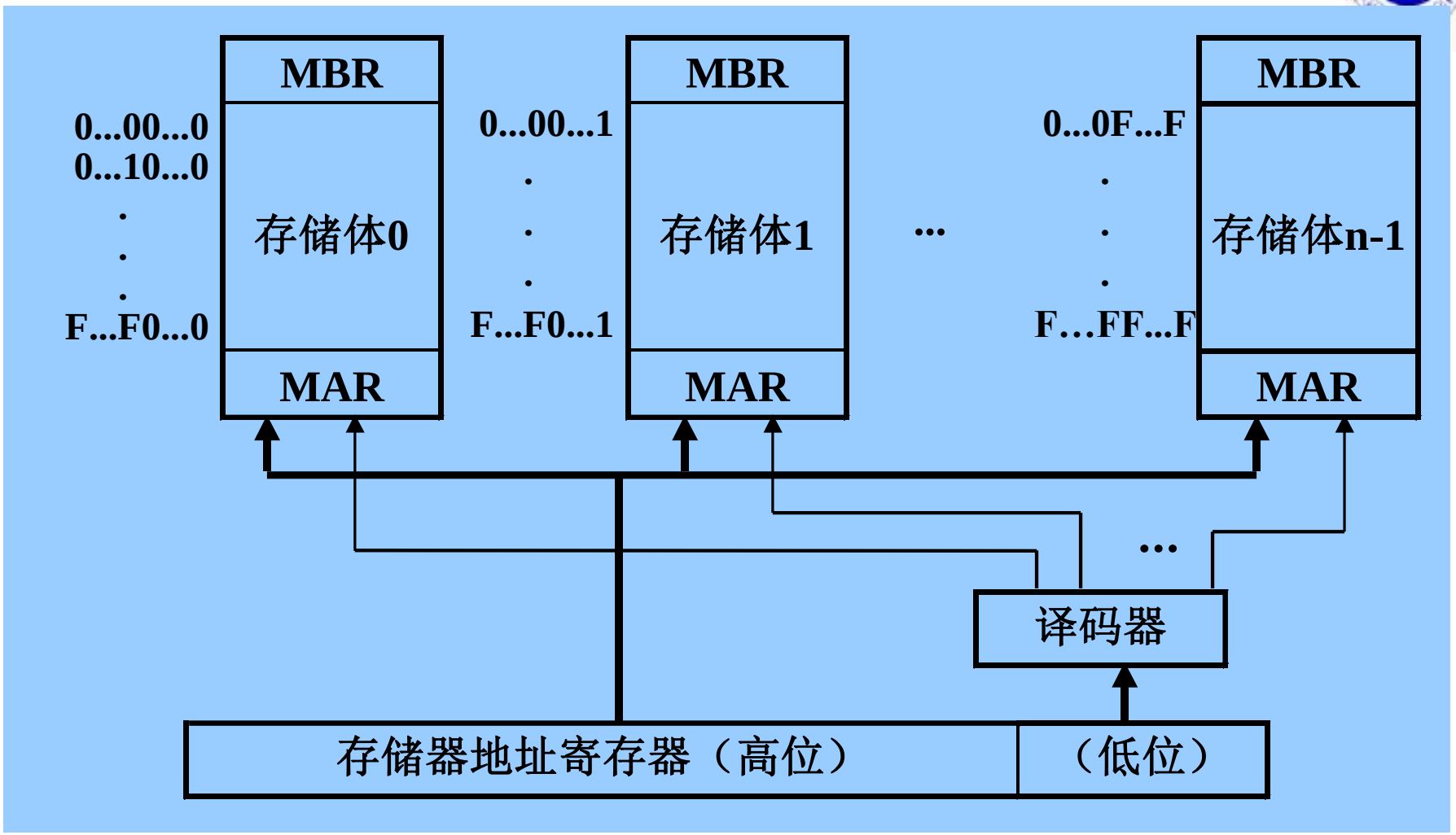
k：存储体的体号，k＝0，1，2，…，n-1

存储器地址A的计算公式为：$A = n \times j + k$

存储器的体内地址：$A_j = \left\lfloor \frac{A}{n} \right\rfloor$

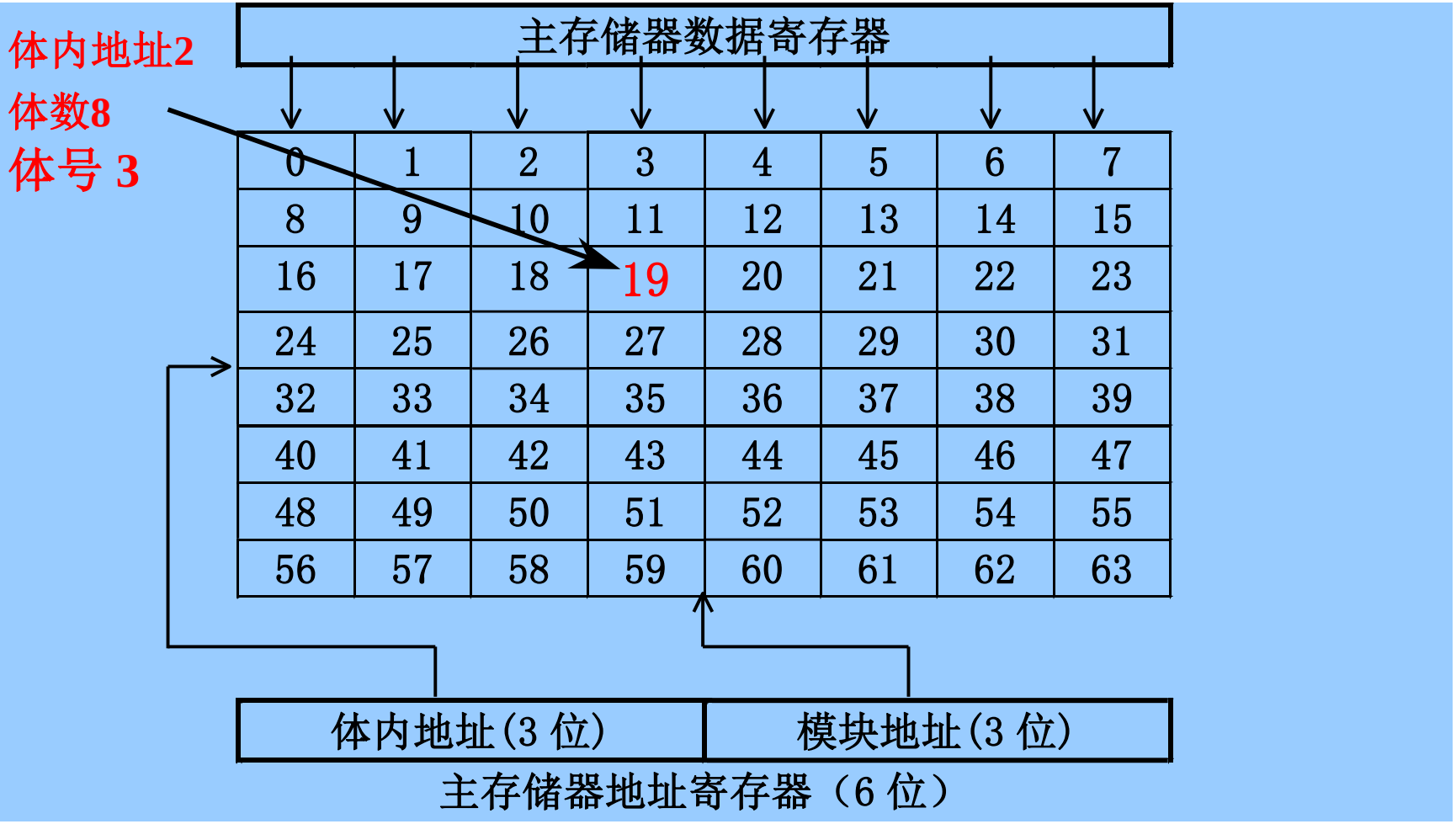
存储器的体号：$A_k = A \bmod n$

• 低位交叉访问存储器结构框图



地址编码方法：

由8个存储体构成的低位交叉编址方式



几台巨型、大型计算机的主存储器结构

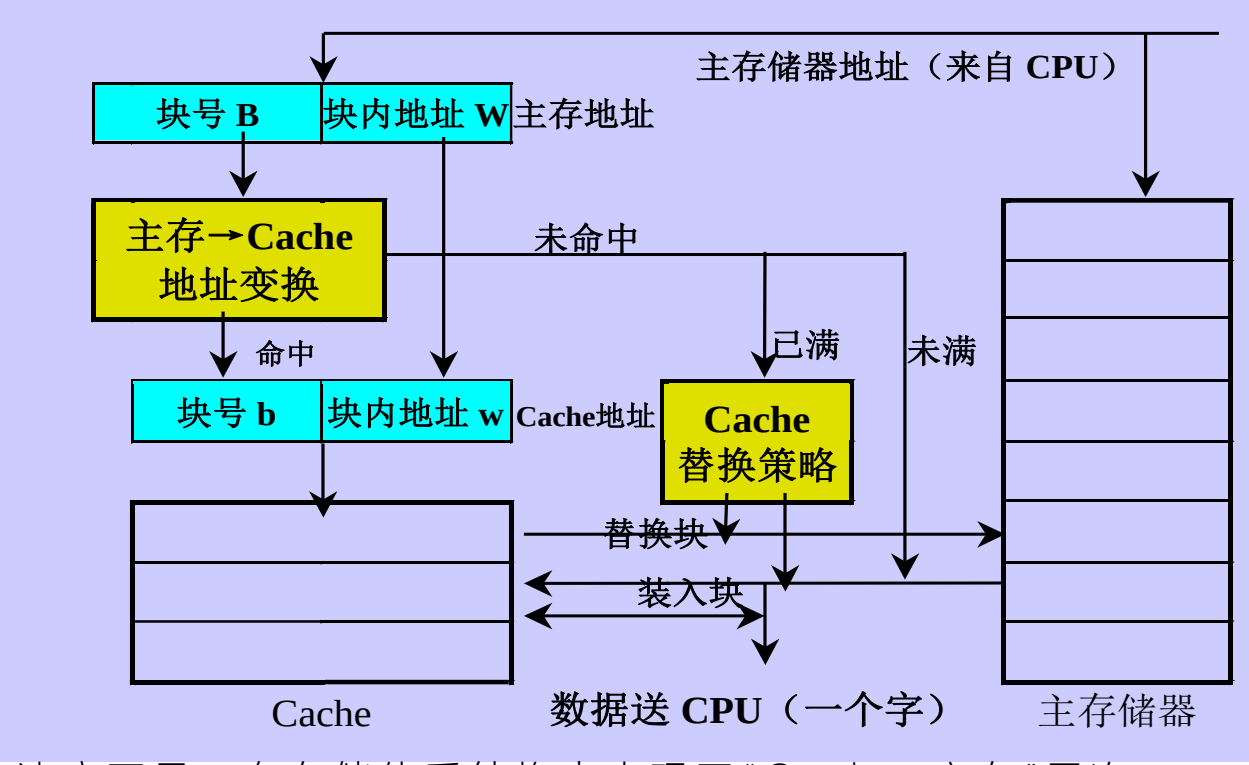
机器型号	存储体个数	存储体字长	存储周期 Tm	频带宽度 Bm
IBM370/165	4	32	2,000ns	8MB/s
CDC6600	32	32	1,000ns	128MB/s
CDC7600	32	32	275ns	465MB/s
CRAY-1	16	64	50ns	2,560MB/s
ASC	8	256	160ns	1,600MB/s
STAR-100	32	512	1,280ns	1,600MB/s

3.4 高速缓冲存储器（Cache）

3.4.1 Cache基本结构和工作原理
3.4.2 Cache的替换算法分析
3.4.3 Cache的透明性 Cache的一致性
3.4.4 任务切换对失效率的影响
3.4.5 多处理机系统的Cache结构
3.4.6 Cache-主存层次性能分析
3.4.7 Cache性能计算

3.4.1 Cache基本结构和工作原理

1. 基本工作原理



为了弥补主存速度不足，在存储体系结构中出现了"Cache-主存"层次，Cache的速度和主存的容量。Cache在当代计算机系统中已普遍使用，成为提高系统性能的不可缺少的功能部件。容量不断增大、管理实现全硬化、部件高度集成，已成为当代Cache的特征。

3.4.1 Cache基本结构和工作原理

1．访问局部化

2．Cache基本结构

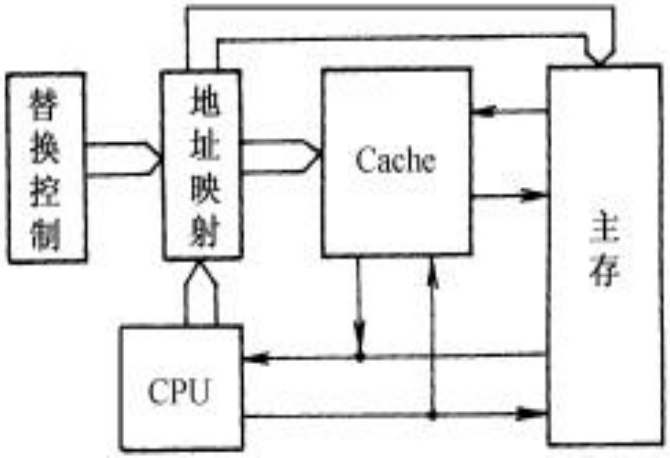


图3-32　Cache存储系统的原理框图

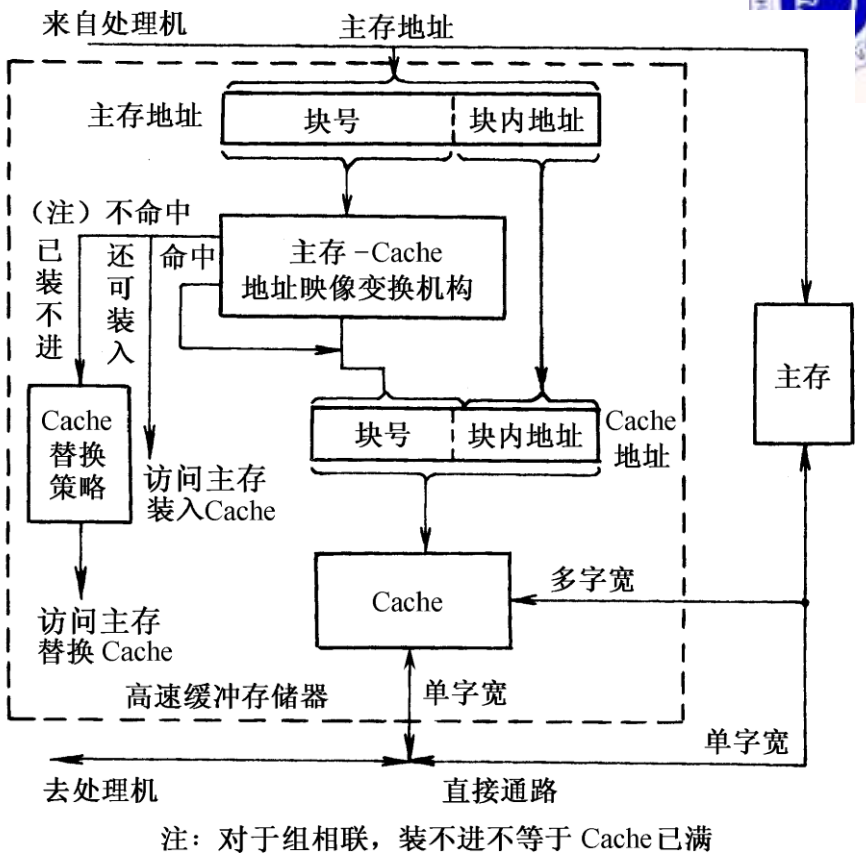


图3-33　Cache基本结构

Cache的设计要求是：在价格允许的前提下，提高命中率和缩短访问时间，尽可能减少因不命中造成的时间延迟以及为修改主存所花的时间开销。

直接通路实现读直达 CPU 直接写入主存写直达

主存与Cache之间采用主存多体交叉多字结构

3.4.2 Cache替换算法及其分析

使用的场合：

直接映象方式实际上不需要替换算法

全相联映象方式的替换算法最复杂

主要用于组相联、段相联等映象方式中

要解决的问题：

记录每次访问Cache的块号

在访问过程中，对记录的块号进行管理

根据记录和管理结果，找出替换的块号

主要特点：全部用硬件实现

- Cache地址变换一般采用组相联映像
- 绝大多数Cache采用LRU替换算法。
- 预取法的效果不能只用提高命中率来衡量，还要考虑访存开销等因素。

Cache的透明性

- 由于地址变换和块替换算法由硬件实现，对系统程序员和用户都是透明的

1 地址映像与变换方法

地址映像：

把主存中的程序按照某种规则装入到Cache中，建立主存地址与Cache地址之间的对应关系。

地址变换：

当程序已经装入到Cache之后，在程序运行过程中，把主存地址变换成Cache地址。

在选取地址映像方法要考虑的主要因素：

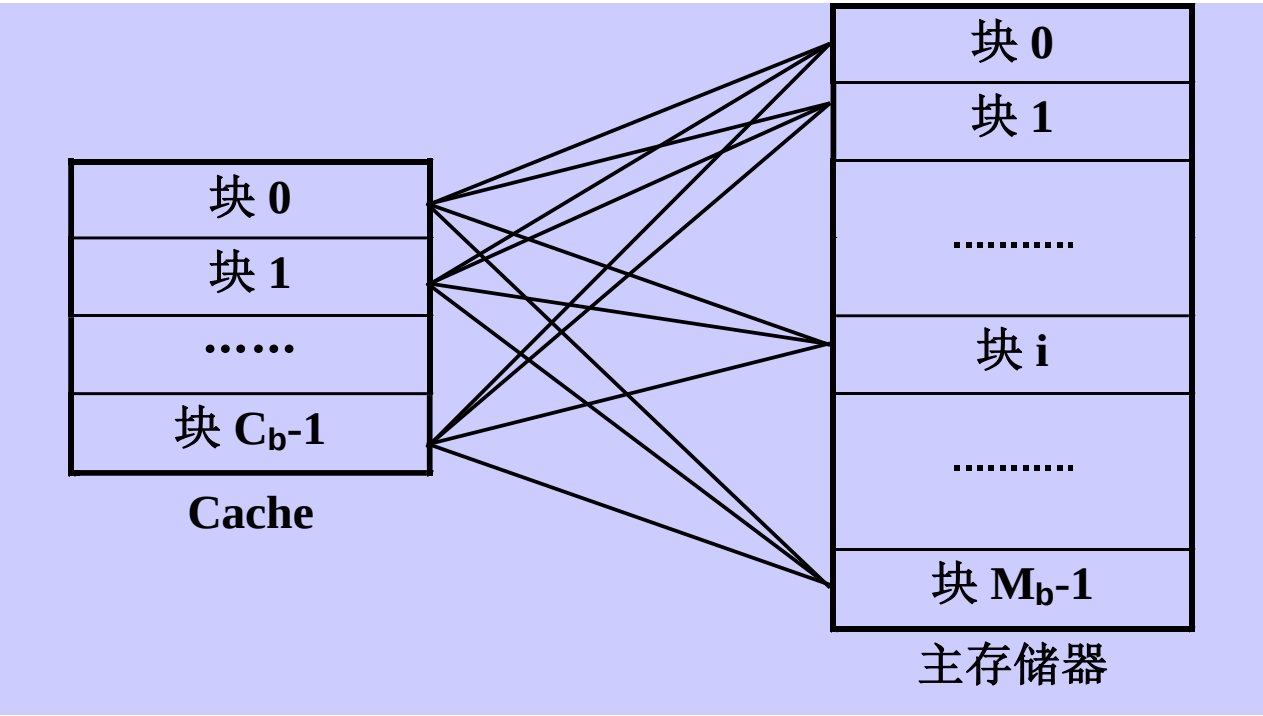
地址变换的硬件实现容易、速度要快，

主存空间利用率要高，

发生块冲突的概率要小

1)．全相联映像及其变换

映像规则：主存的任意一块可以映象到Cache中的任意一块。（映象关系有$C_b \times M_b$种）



2). 直接映像及其变换

映像规则：

主存储器中一块只能映像到Cache的一个特定的块中。

Cache地址的计算公式：

$b = B \bmod C_b$

其中：b为Cache块号，

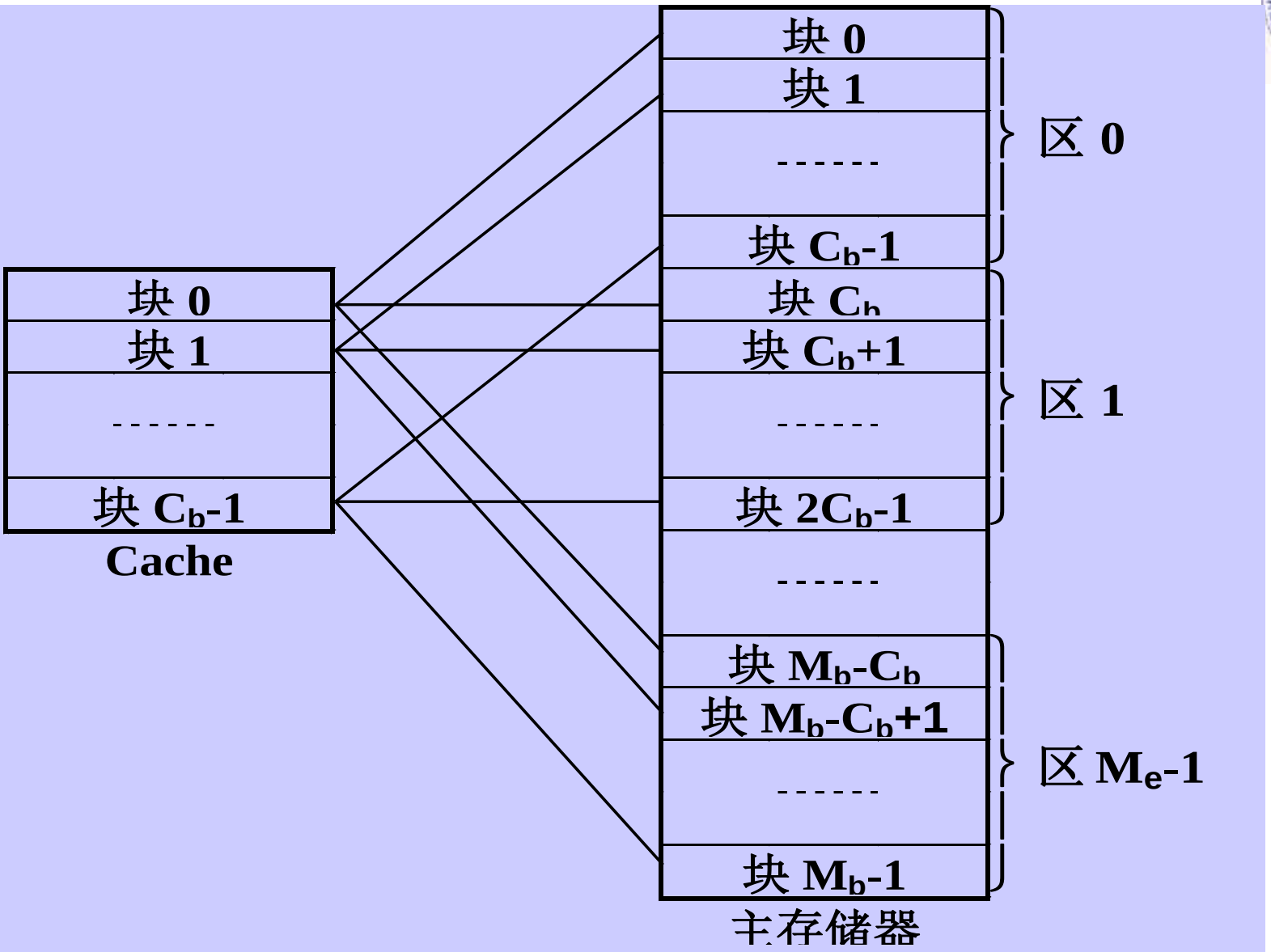
B是主存块号，

C_b是Cache块数。

实际上，

Cache地址与主存储器地址的低位部分完全相同。

直接映像方式的地址映像规则



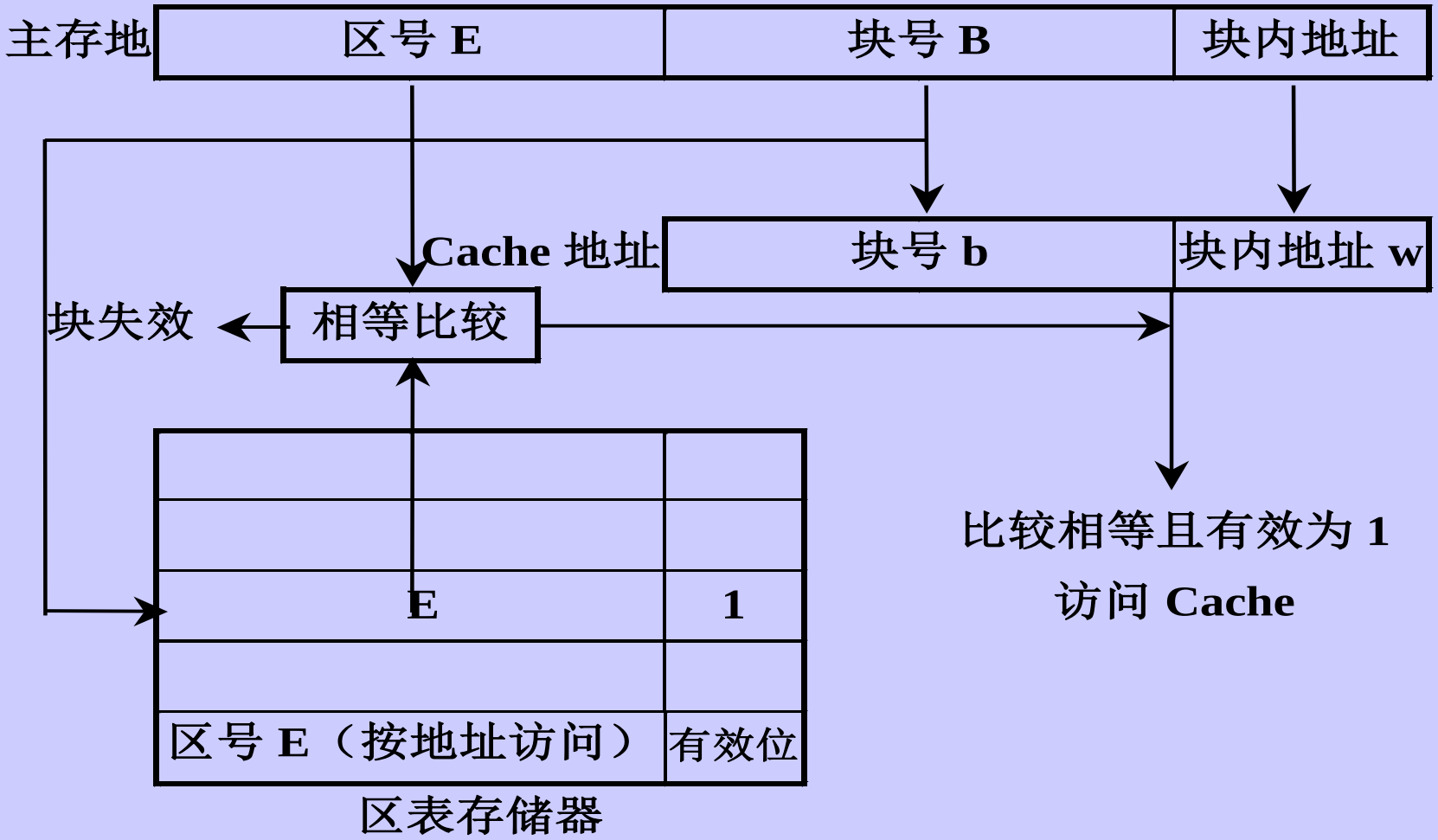
直接映像方式的地址变换过程：

用主存地址中的块号B去访问区号存储器，把读出来的区号与主存地址中的区号E进行比较：

比较结果相等，有效位为1，则Cache命中，否则该块已经作废。

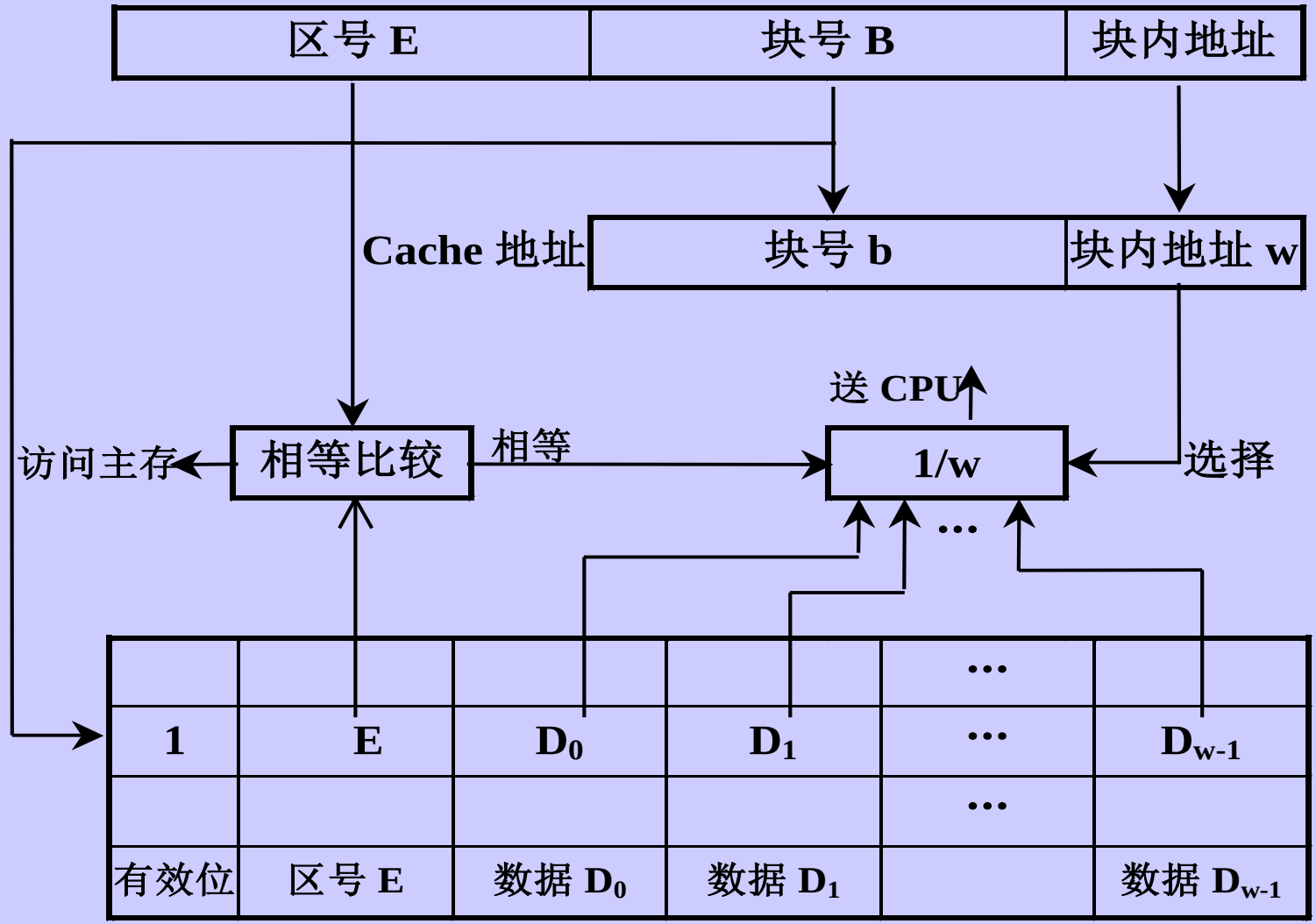
比较结果不相等，有效位为1，Cache中的该块是有用的，否则该块是空的。

直接映象方式的地址变换规则



提高Cache速度的一种方法：

把区号存储器与Cache合并成一个存储器



按地址访问的 Cache

2). 直接映像及其变换的优缺点

- 主要优点：

  硬件实现很简单，不需要相联访问存储器

  访问速度比较快，实际上不需要进行地址变换

- 主要缺点：

  块的冲突率比较高。

3). 组相联映像及其变换

映像规则：

主存和Cache按同样大小划分成块和组

主存和Cache的组之间采用直接映像方式

在两个对应的组内部采用全相联映像方式

组相联映像方式的优点：

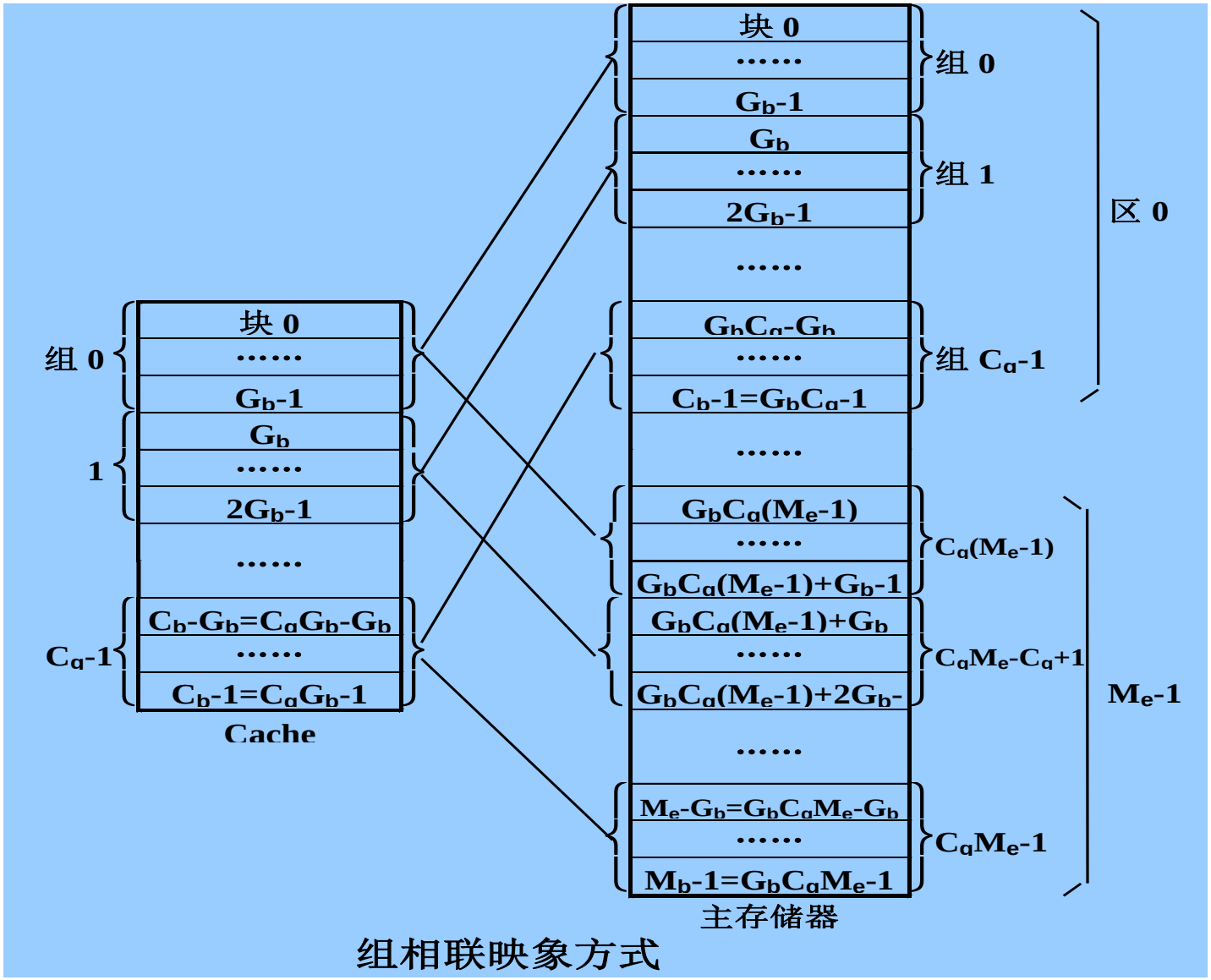
块的冲突概率比较低

块的利用率大幅度提高

块失效率明显降低

组相联映像方式的缺点：

实现难度和造价要比直接映像方式高。


块 0
……
G_b-1
G_b
……
$2G_b$-1
……
G_bC_a-G_b
……
C_b-1=G_bC_a-1
……
$G_bC_a(M_e-1)$
……
$G_bC_a(M_e-1)+G_b-1$
$G_bC_a(M_e-1)+G_b$
……
$G_bC_a(M_e-1)+2G_b$-
……
M_e-G_b=$G_bC_aM_e$-G_b
……
M_b-1=$G_bC_aM_e$-1
主存储器
组 0
组 1
组 C_a-1
区 0
$C_a(M_e-1)$
C_aM_e-C_a+1
C_aM_e-1
M_e-1
组 0
1
C_a-1
块 0
……
G_b-1
G_b
……
$2G_b$-1
……
C_b-G_b=C_aG_b-G_b
……
C_b-1=C_aG_b-1
Cache
组相联映象方式

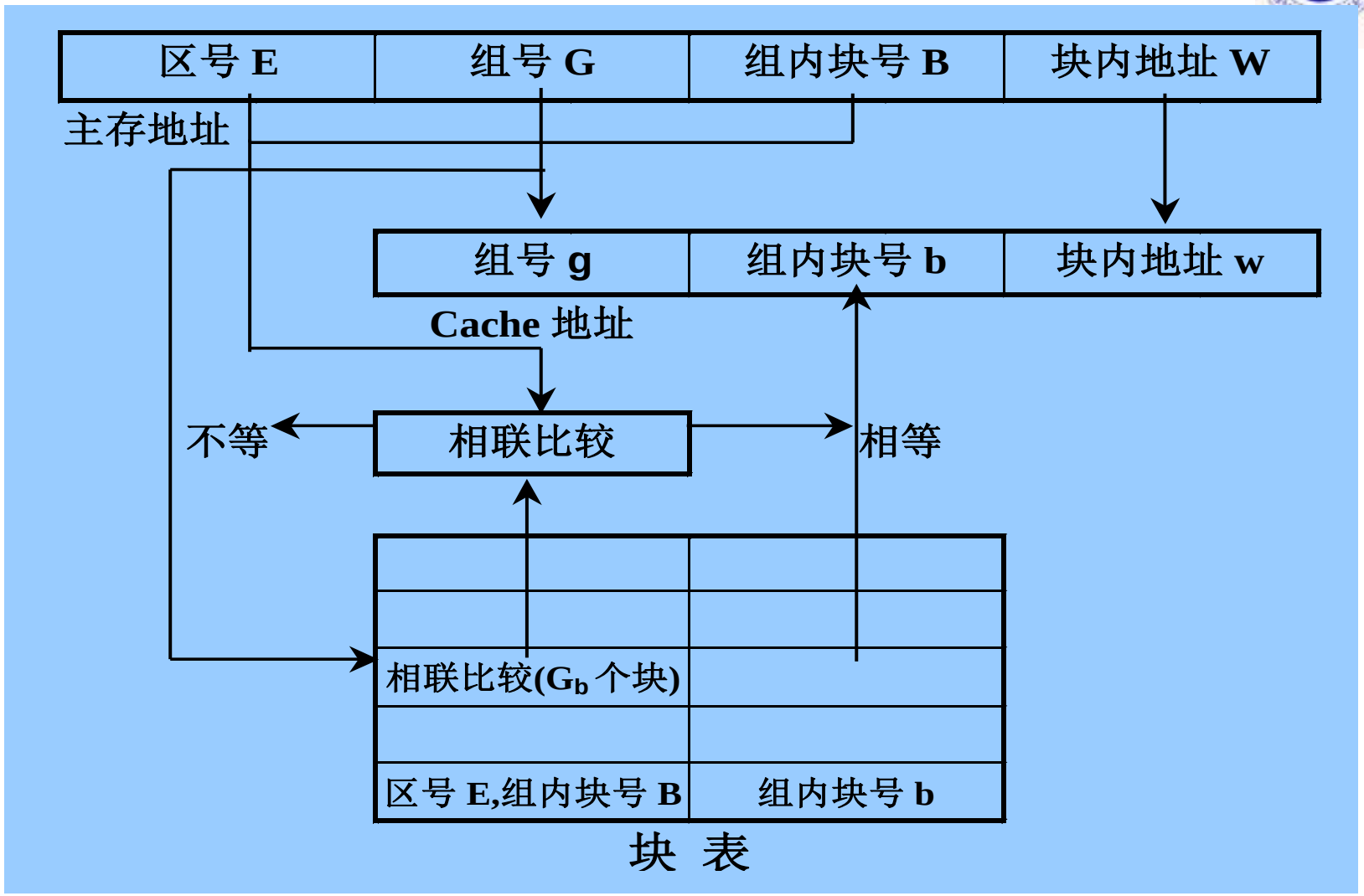
组相联映像的地址变换过程：

用主存地址中的**组号G**按地址访问块表存储器

把读出来的一组区号和块号与主存地址中的区号和块号进行**相联比较**。

- 如果有相等的，表示Cache命中；
- 如果全部不相等，表示Cache没有命中。

• 组相联映像的地址变换


区号 E
组号 G
组内块号 B
块内地址 W
主存地址
组号 g
组内块号 b
块内地址 w
Cache 地址
不等
相联比较
相等
相联比较(G_b 个块)
区号 E,组内块号 B
组内块号 b
块 表

组相联映象方式典型机器的 Cache 分组情况

机器型号	Cache 块数 C_b	每组的块数 G_b	Cache 组数 C_g
DEC VAX-11/780	1024	2	512
Amdahl 470/V6	512	2	256
Intel i860 D-Cache	256	2	128
Honeywell 66/60	512	4	128
Amdahl 470/V7	2048	4	512
IBM 370/168	1024	8	128
IBM3033	1024	16	64
Motolola 88110	256	2	128

4). 位选择组相联映象及其变换

地址映象规则：

主存和Cache都按同样大小分块，

Cache在分块的基础上再分组，

主存按照Cache的组容量分区。

主存的块与Cache的组之间采用直接映象方式，

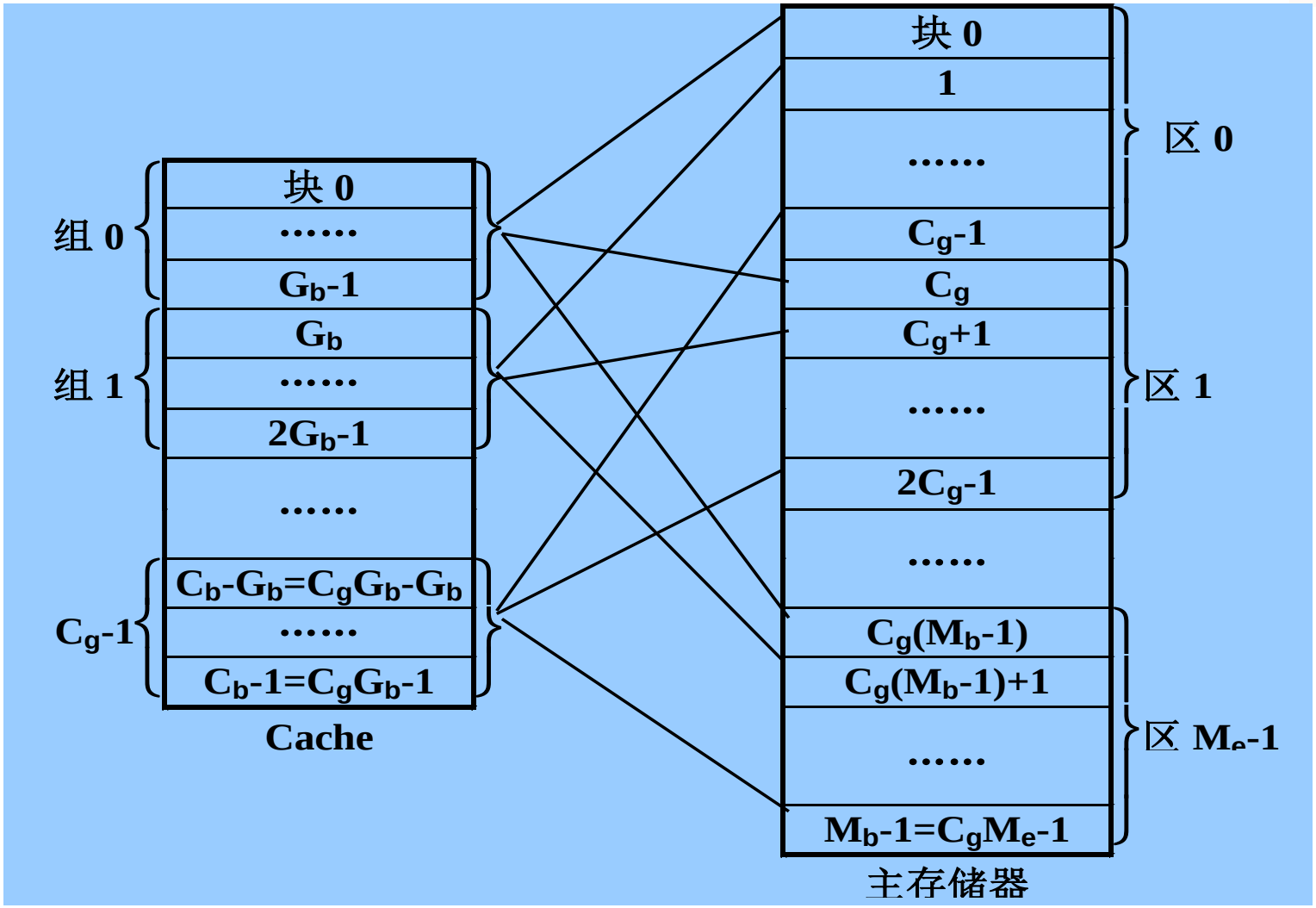
主存中的块与Cache中组内部的各个块之间采用全相联映象方式。

与组相联映象方式比较：

映象关系明显简单，实现起来容易。

在块表中存放和参与相联比较的只有区号E

位选择组相联的地址映象规则



5). 段相联映像及其变换

映像规则：

主存和Cache都按同样大小分块和段

段之间采用全相联映象方式

段内部的块之间采用直接映象方式

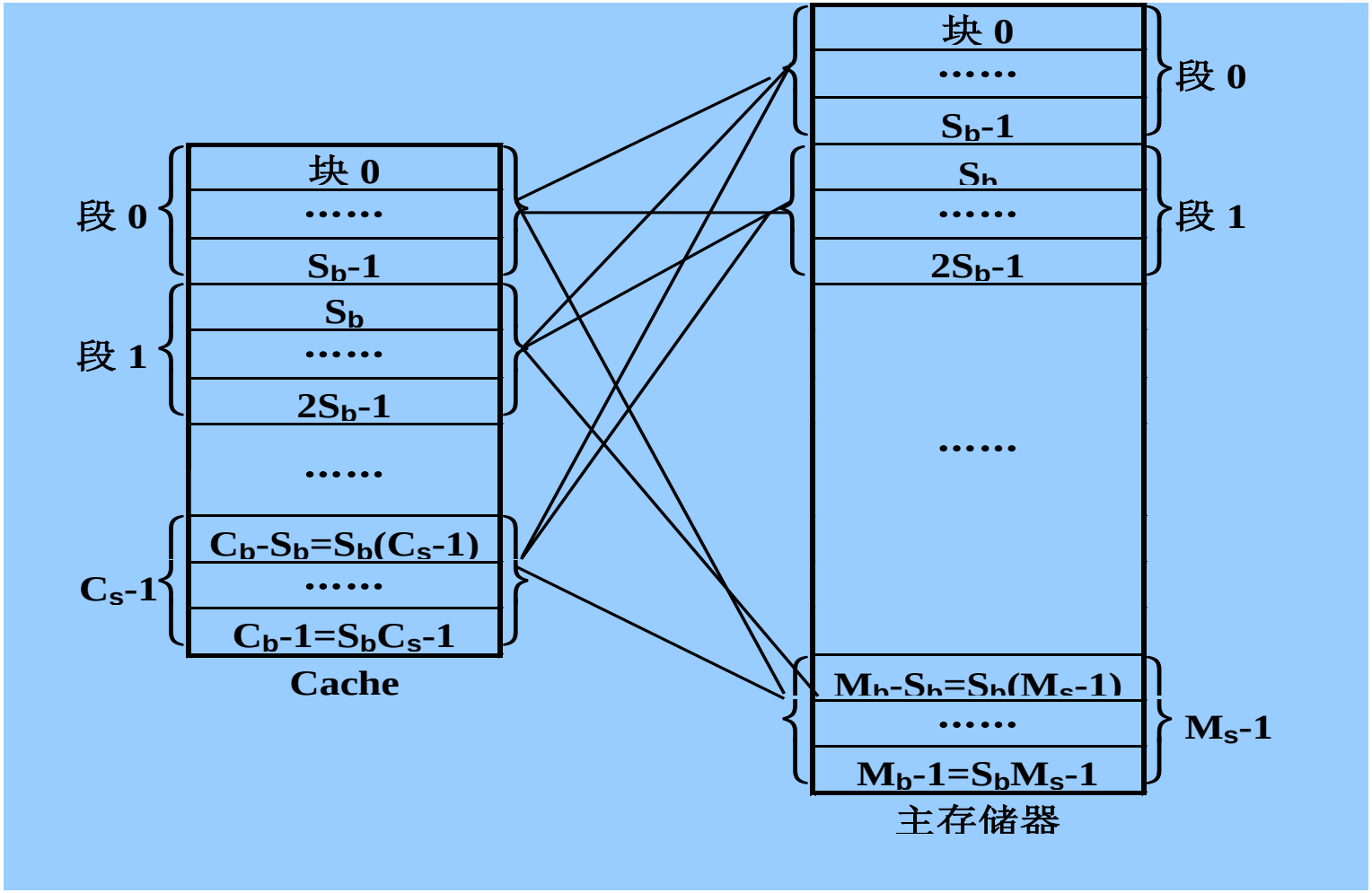
地址变换过程：

用主存地址中的段号与段表中的主存段号进行相联比较

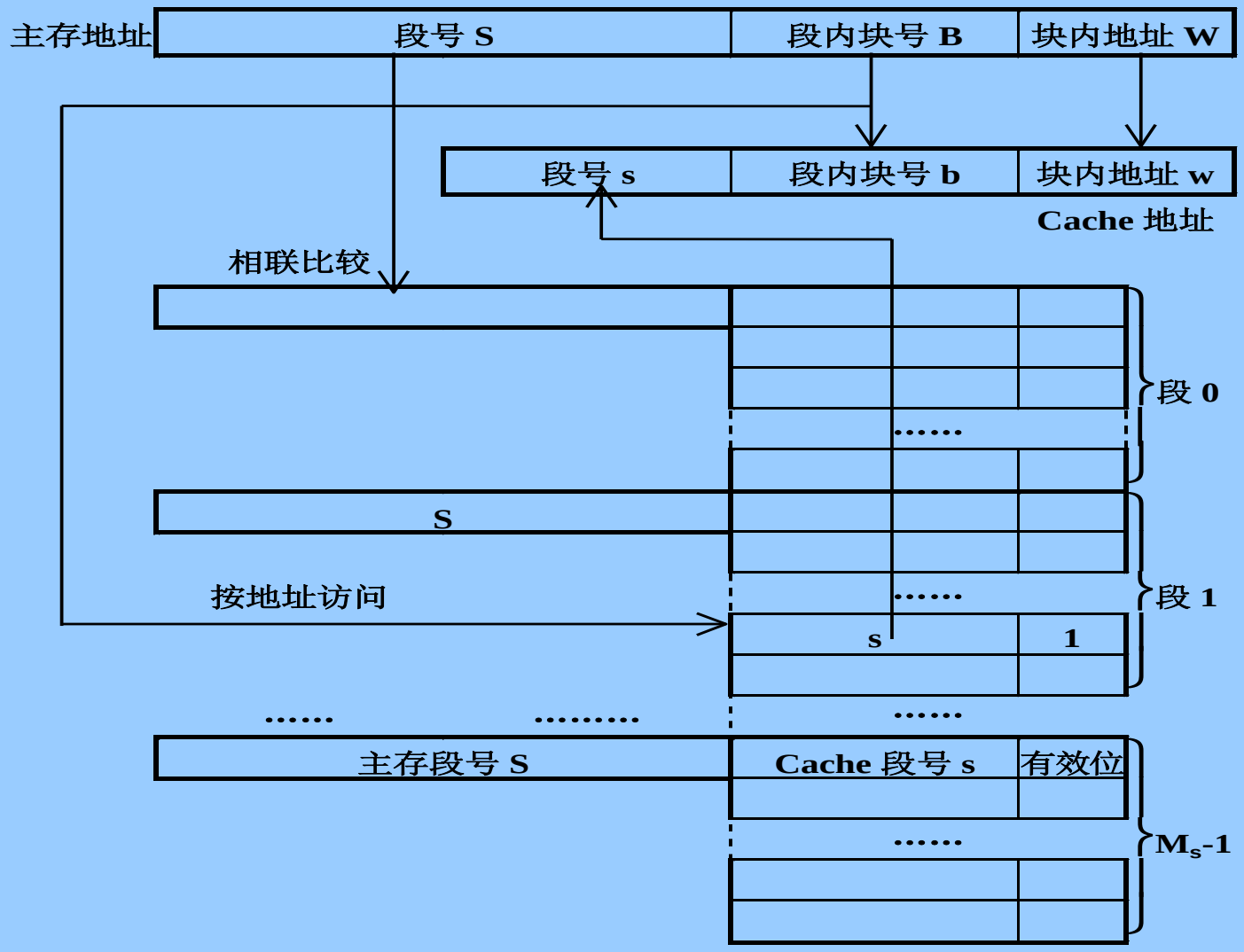
如果有相等的，用主存地址的段内块号按地址访问Cache的段号部分。

把读出的段号s与主存地址的段内块号b及块内地址w拼接起来得到Cache地址；

段相联映像地址映像规则



段相联映象地址变换过程



段表（按地址和相联两种访问方式）

段相联映像方式的优缺点

主要优点：

段表比较简单，实现的成本低。

例如：一个容量为256KB的Cache，分成8个段，每段2048块，每块16B。

在段表存储器中只需要存8个主存地址的段号，而在块表中要存储8×2048＝16384个区号，两者相差2000多倍。

主要缺点：

当发生段失效时，要把本段内已经建立起来的所有映像关系全部撤消。

2. 实现方式

1). 轮换法及其实现

用于组相联映像方式中，有两种实现方法。

方法一：每块一个计数器

在块表内增加一个替换计数器字段，

计数器的长度与Cache地址中的组内块号字段的长度相同。

替换方法及计数器的管理规则：

新装入或替换的块，它的计数器清0，

同组其它块的计数器都加“1”。

在同组中选择计数器的值最大块作为被替换的块。

方法二：每组一个计数器

替换规则和计数器的管理：

本组有替换时，计数器加“1”，

计数器的值就是要被替换出去的块号。

例 NOVA3机的Cache采用组相联映象方式，Cache每组的块数为8，每组设置一个3位计数器。在需要替换时，计数器的值加“1”，用计数器的值直接作为被替换块的块号。

轮换法的优点：实现比较简单，能够利用历史上的块地址流情况

轮换法的缺点：没有利用程序的局部性特点

2). LRU算法及其实现

为每一块设置一个计数器

计数器的长度与块号字段的长度相同

计数器的使用及管理规则：

新装入或替换的块，计数器清0，同组中其它块的计数器加1。

命中块的计数器清0，同组的其它计数器中，凡计数器的值小于命中块计数器原来值的加1，其余计数器不变。

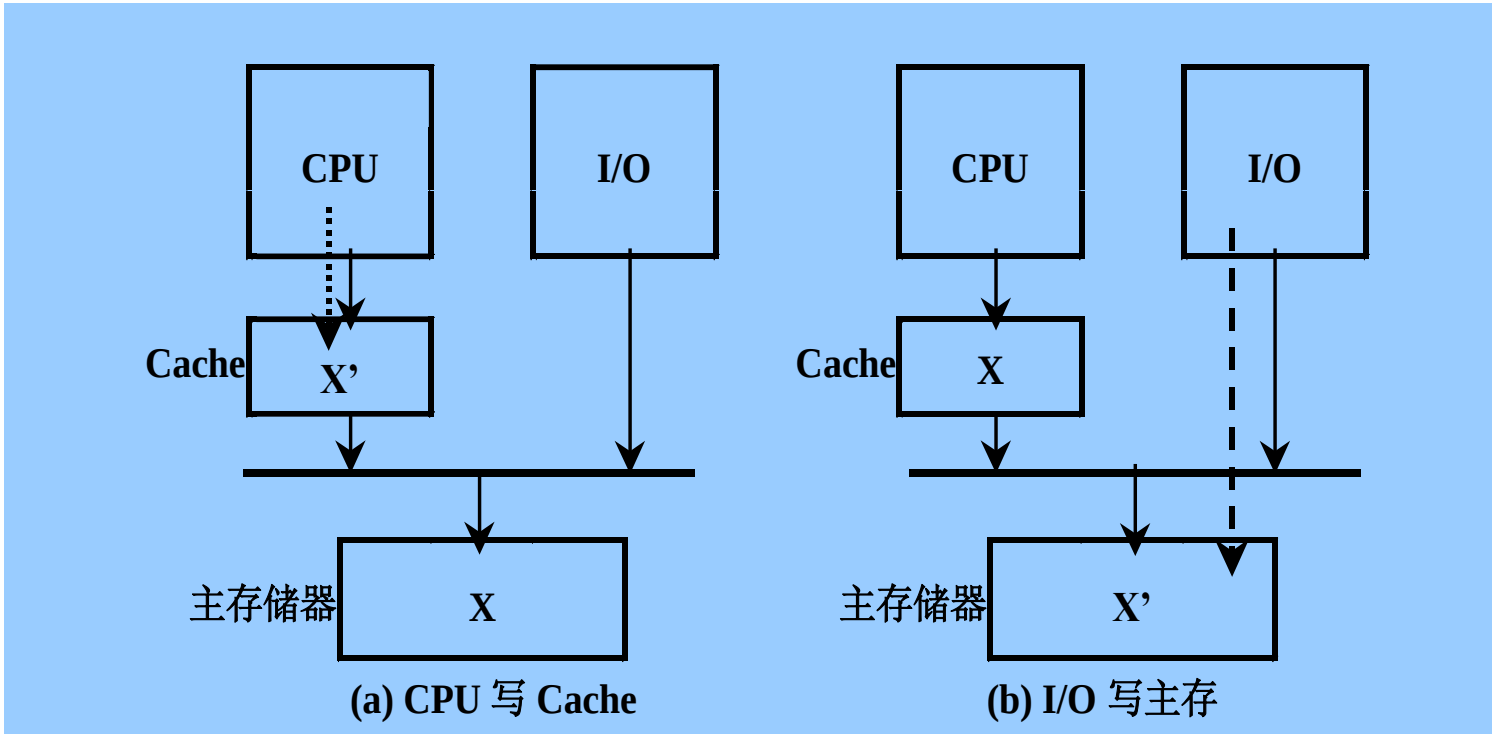
需要替换时，在同组的所有计数器中选择计数值最大的计数器，它所对应的块被替换。

3.4.3 Cache的一致性 Cache透明性

造成Cache与主存的不一致的原因：

(1) 由于CPU写Cache，没有立即写主存

(2) 由于IO处理机或IO设备写主存



Cache的读/写操作

(1)标志交换方式（写回法）WB(Write-Back)

CPU的数据只向Cache写入，并用标志注明，直至该块被替换，才将该块写回主存

写入Cache，不写入主存，仅当替换时，才把修改过的Cache块写回主存

(2)写直达法，写通过法，WT(Write-through)

CPU的数据写入Cache时，同时也写入主存，使原本和副本同时修改

写回法与写直达法的优缺点比较：

写回法对总线带宽占用少，写直达法实现简单

可靠性，写直达法优于写回法。

写直达法能够始终保证Cache是主存的副本。

如果Cache发生错误，可以从主存得到纠正。

写Cache的两种方法：

(1)不按写分配法：

在写Cache不命中时，只把所要写的字写入主存。

(2)按写分配法：

在写Cache不命中时，还把一个块从主存读入Cache。

目前，**在写回法中采用按写分配法，**

在写直达法中采用不按写分配法。

Cache的透明性

由于地址变换和块替换算法由硬件实现，对系统程序员和用户都是透明的。

3.4.4任务切换对失效率的影响

- Qsw—任务切换的平均时间间隔
- Qsw小，说明切换频繁；
- Cache的冷启动失效率——指Cache为空到装满这一段时间的失效率
- Cache的热启动失效率——指Cache装满后测出的失效率

1.Qsw和工作负荷对失效率的影响

- 工作负荷大，Cache满，Qsw小（说明切换频繁），失效率高。

2.Qsw和Cache容量对失效率的影响

- Qsw一定时， Cache容量小，热启动失效率高，冷启动失效率低；
- Cache容量增大，热启动失效率降低，冷启动失效率低增大。
- Cache容量＝工作区，热启动失效率下降平缓。
- Cache容量增大，热启动失效率迅速降低，增大Qsw使失效率明显减少。

任务切换导致Cache失效率解决方法：

- 增大Cache容量
- 改善调度算法
- 设置多个Cache用于不同的目的。

3.4.5多处理机系统的Cache结构

多处理机类型：

1. 一个CPU和多个I/O处理机（共享主存系统）
2. 多个CPU
3. 多个CPU和多个I/O处理机

共享一个Cache

优点：保证信息的一致性

缺点：一个Cache带宽很难满足两个以上CPU的访问要求

每个处理机都有自己的Cache

如何保证信息的一致性

（1）播写法　　广播，各Cache内有就写入

（2）控制某些共享信息进入Cache

（3）目录表法

Cache不命中 查表判定是否在别的Cache内，是否被修改，再决定如何读写此单元。

3.4.6 Cache-主存层次性能分析

- 块的大小、组的大小及Cache容量增大都会提高命中率
- Cache在调块时，CPU空等，这时希望块的大小较小
- 等效存储周期 Ta=Hc*tc+(1-Hc)tm
- 平均存储器访问时间=命中时间+不命中率*不命中代价
- **Cache 性能计算 Cache优化策略 Cache优化技术总结**
- **[例3-1]**16KB 指令Cache加一个16KB数据Cache与一个32KB一体Cache相比较，哪一个具有更低的不命中率？
- **[例3-2]**设Cache不命中代价为100个时钟周期，所有指令都用1个时钟周期完成，平均不命中率为2%，平均每条指令访问存储器1.5次，每1000条指令平均Cache不命中次数为30，分别用不命中率和每条指令不命中次数计算Cache对计算机性能（即CPU时间)的影响。
- **[例3-3]**设Cache为理想状态时，CPI为2.0，时钟周期为1.0ns，平均每条指令访存1.5次，两个Cache容量都是64KB，块容量为64B。分别计算采用直接映像和组相联映像，分别计算平均存储器访问时间以及CPU性能。
- **[例3-4]** 沿用例[3-3]，设通过加长处理器时钟周期到1.25倍支持乱序执行，分析平均存储器访问时间和CPU性能（CPU时间）；
- **[例3-5]**设在1000次访存中，一级Cache的L1有40次不命中，L2有20次不命中，分析不命中率及存储器平均访问时间。
- **[例3-6]**设L2直接映像时命中时间为10个时钟周期，采用2路组相联时命中时间为10.1个时钟周期， L2直接映像的局部不命中率为25%，采用2路组相联时局部不命中率为20%，L2的不命中代价为100个时钟周期，分析L1的不命中代价。

3.5 虚拟存储器

3.5.1 虚拟存储器基本结构和工作原理
3.5.2 虚地址和辅存实地址的变换
3.5.3 多用户虚拟存储器
3.5.4 加快地址变换的方法
3.5.5 虚拟存储器性能分析

3.5.1 虚拟存储器基本结构和工作原理

虚拟存储系统、虚拟存储体系等

概念由英国曼彻斯特大学Kilbrn等人于1961年提出

到70年代广泛应用于大中型计算机系统

目前，许多微型机也使用虚拟存储器

把主存储器、磁盘存储器和虚拟存储器都划分成固定大小的页

主存储器的页称为实页

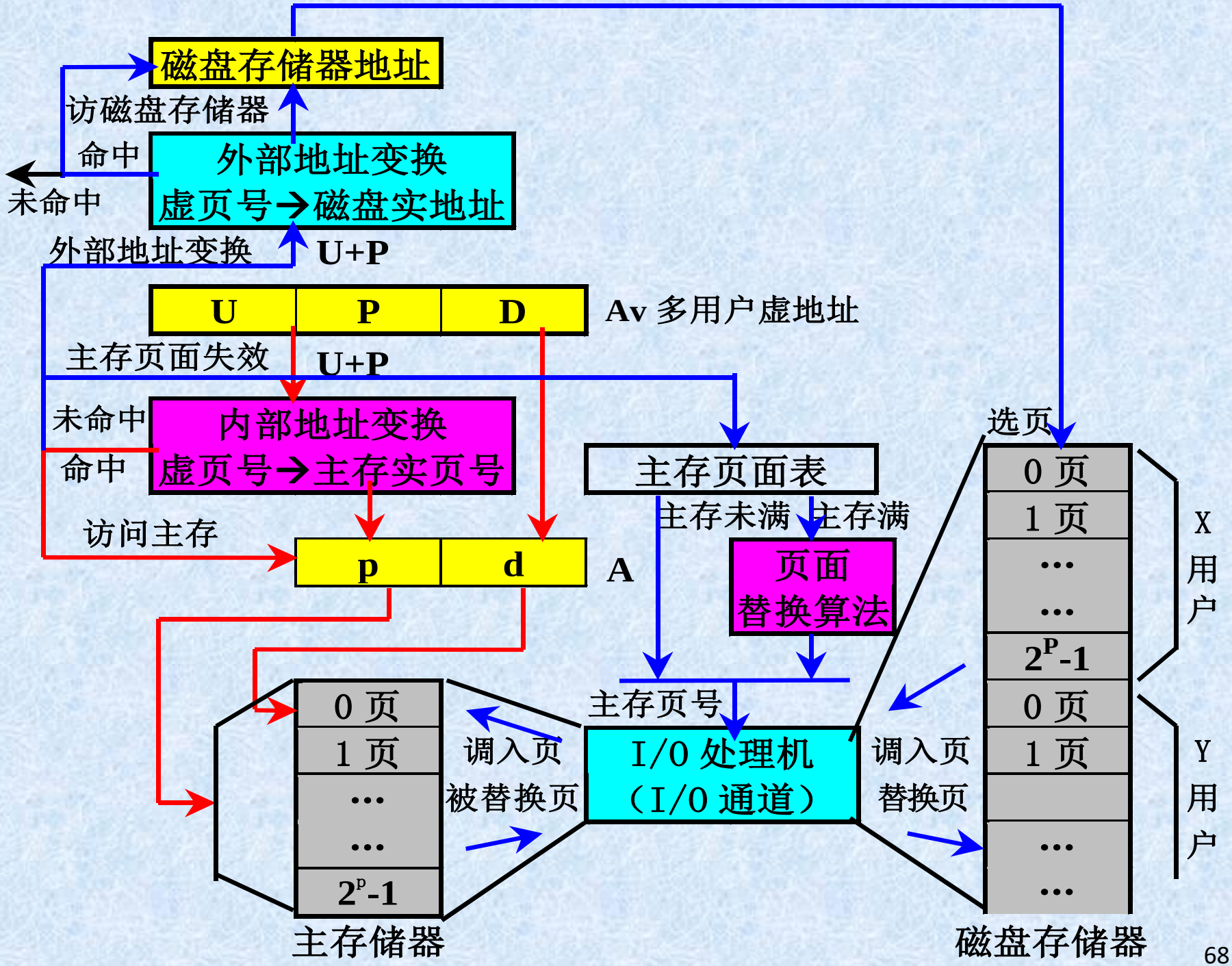
虚拟存储器中的页称为虚页

虚拟存储器是指"主存-辅存"层次，具有辅存容量、接近主存的等效速度和辅存的每位成本，使程序员可以按比主存大得多的虚拟存储空间编写程序（即按虚存空间编址）

Cache的主要作用是弥补主存和CPU之间的速度差距，用硬件实现对程序员透明。

虚拟存储器的主要作用是弥补主存和辅存之间的容量差距，

基本软件实现，适当结合硬件实现


磁盘存储器地址
访磁盘存储器
命中
外部地址变换
虚页号→磁盘实地址
未命中
外部地址变换
U+P
U
P
D
Av 多用户虚地址
主存页面失效
U+P
未命中
内部地址变换
虚页号→主存实页号
命中
访问主存
p
d
A
主存页面表
主存未满
主存满
页面
替换算法
主存页号
I/O 处理机
（I/O 通道）
调入页
被替换页
调入页
替换页
选页
0 页
1 页
…
…
2^p-1
0 页
1 页
…
…
2^P-1
0 页
1 页
…
…
X
用
户
Y
用
户
主存储器
磁盘存储器

3.5.2 虚地址和辅存实地址的变换

1.虚地址—辅MEM实地址格式

用户程序的机器指令

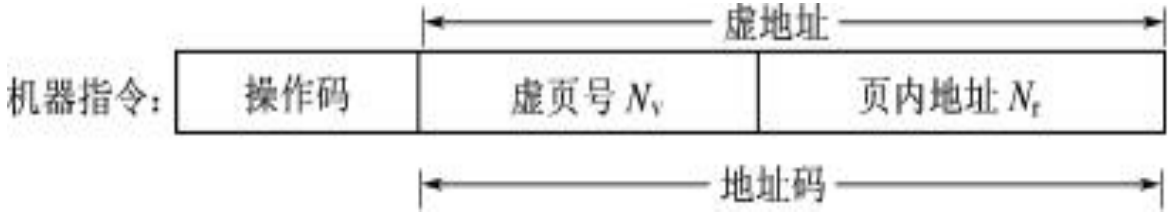


图3-37 虚地址组成

NV：NV位二进制编码

辅存—磁盘实地址格式

图3-38 实地址示例

3.5.2 虚地址和辅存实地址的变换

- 装入位为1，表示外页表内的为有效辅存（磁盘）的实地址。
- 虚地址到辅存实地址的映像采用全相联方式，其变换用软件实现。

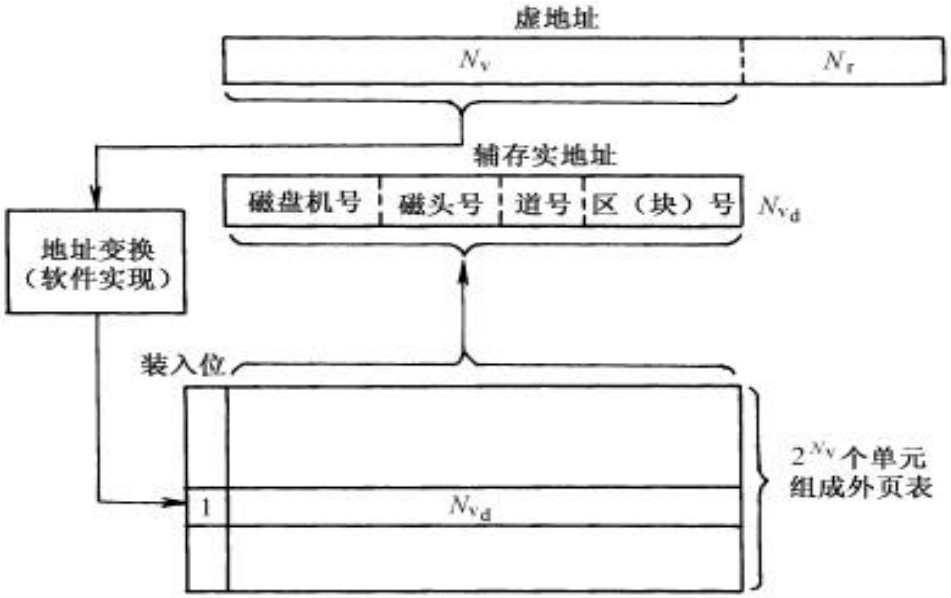


图3-39 虚地址到辅存实地址的变换过程

3.5.3 多用户虚拟存储器

1.多用户虚拟MEM分配

- 每个用户分配2^N个单元
- 另用U个bit→对应2^U个用户

} 整虚存 $2^U \cdot 2^{N_V} \cdot 2^{N_r}$

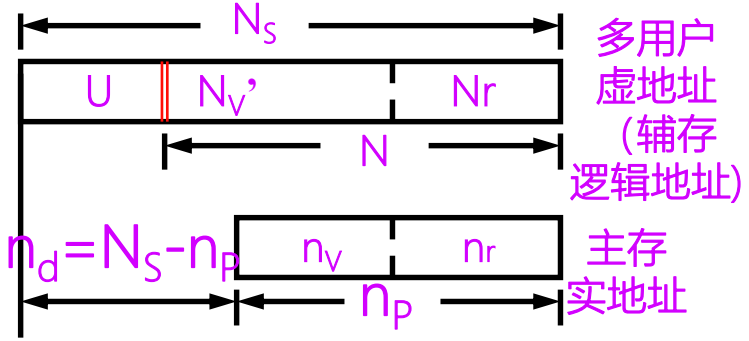
2.多用户虚地址NS=U+指令中虚地址N

- 格式：
- 主存地址格式：

nr：页内实地址。

nv：页实地址。

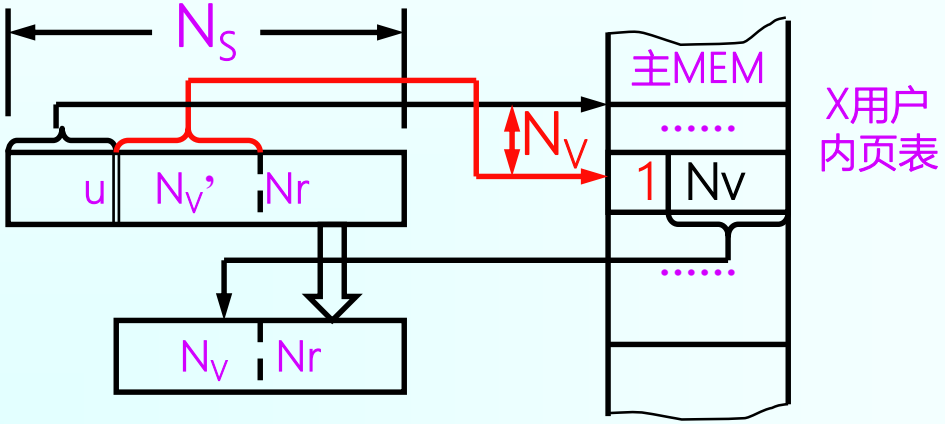
- U为用户编号→放于未用寄存器。



3.5.3 多用户虚拟存储器

3.“全相联页式管理”查表法

* 在主存中，对应每个用户建立内页表。



用户编号ID Reg→u（X用户）

指令中虚地址 N转换成N_V+N_r 实地址

u→指向X用户内页表首址。

u+N_V→指向页表内X用户的某单元。

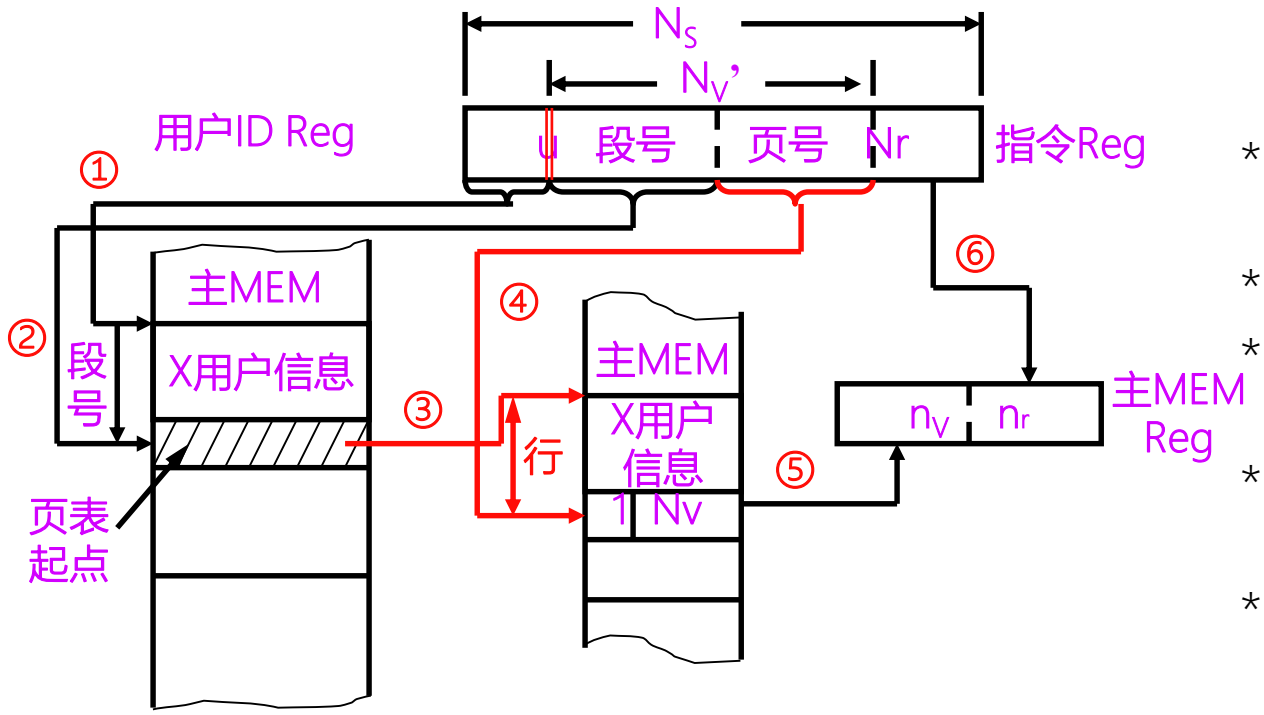
* 取出该单元内容→判装入位=1→是，则后面N_V为有效。

* 取出NV与指令虚地址中Nr，即Nr=nr，

nv+nr(Nr) →指向实存，即nv+nr=实存实地址。

3.5.3 多用户虚拟存储器

4.“全相联段页式管理”查表法



* 指令码中虚地址的NV分成段号、页号

NV=段号+页号

* 在主MEM对每一用户开辟两个表：段表、页表。
* u→X用户段表起始。
* 段号→段表内某单元→取出→X用户页表起始。
* 页号→页表内某单元→取出→主MEM Reg中N_V。
* 实地址=n_V+n_r（Nr）。

3.5.3 多用户虚拟存储器

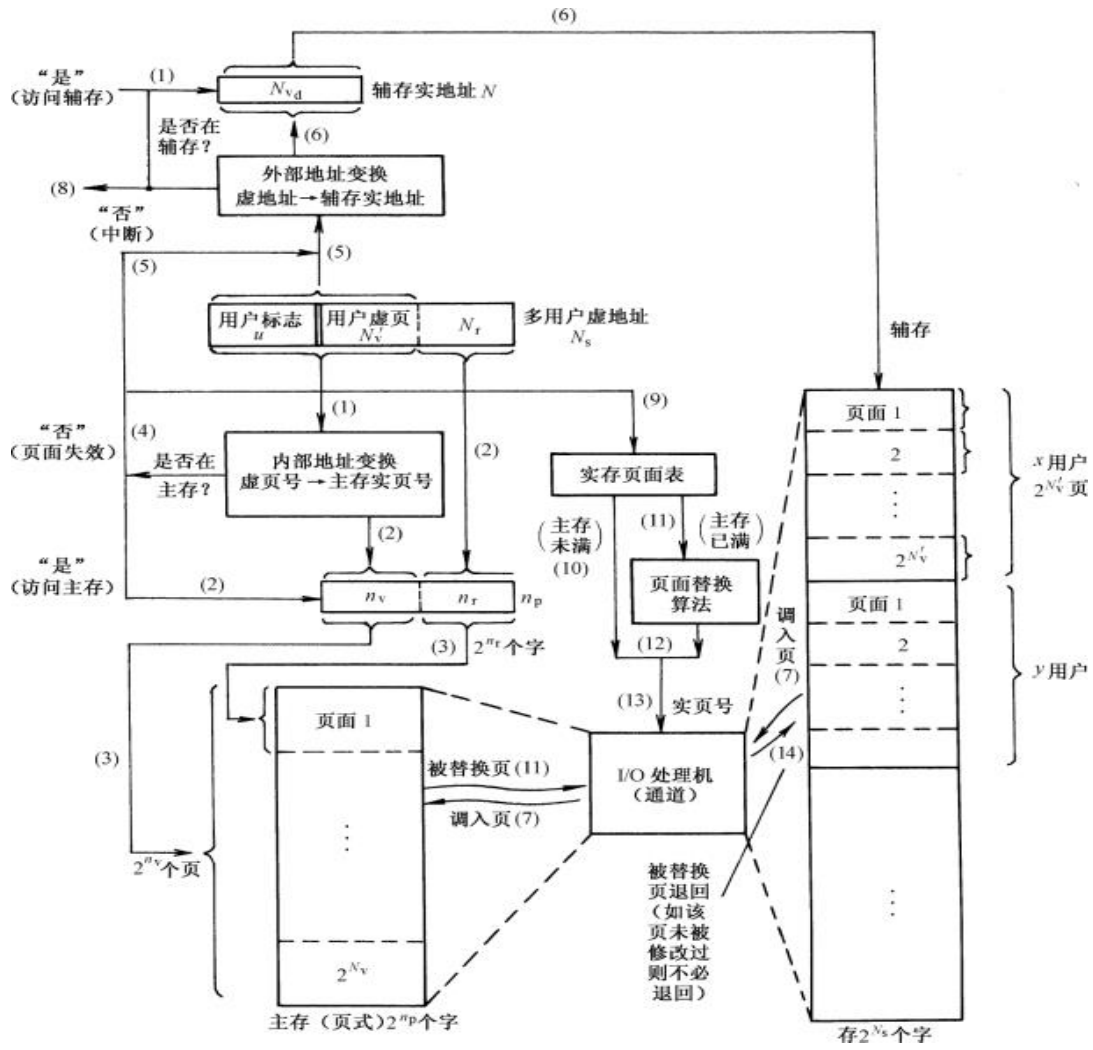


图3-43 多用户虚拟存储器工作全过程

1.取值→用户虚地址Ns(用户标志u+用户虚页号N_V' →内部地址变换。

2.查内页表u+NV' →取出内页表对应单元内容→判装入位=1→是，则3；否，则4。

3.内页表装入位为1，单元内容n_V+nr（N_r）→拼接成主存实地址→完成MEM RD/WR。

4.内页装入位=0→页面失效中断 →转5，启动外部地址转换；→同时转入9，启动查实存页表

5.u+N_V' →外部地址转换形成辅存实地址。

6.判断该页是否在辅存

→在，则将辅存地址送辅存

→否，引发中断，转8。

7.在I/O处理机（通道）参与下，将该页从辅存中调入主存。该操作与CPU的运行并行进行。

3.5.3 多用户虚拟存储器

8.该页不在辅存，则转入中断，从海量存储器调入辅存

9.与5、6并行，页面失效时，进入查实存页表（LRU页表）（主MEM已满→11）。

10.查实存页表主MEM未满，则将内部地址变换后得到的实页号送实存页表。

11.查实存页表主MEM已满，则执行LRU算法

12.找到被替换的页，得到该页的实页号。

13.将实页号送I/O处理机。 I/O处理机根据辅存实地址到辅存读出一页信息，然后根据主存实页号将该页信息写入主存。

14.页面替换时，如果被替换的页调入主存后一直未经修改则不需回送辅存；如果已修改，则需先将其送回辅存原来位置，而后再把调入页装入主存。

3.页面失效中断处理

- 用缓冲Reg将页面失效时，中断现场，保护，新页面调入后，继续执行（原处连续下去或重头开始—被打断的指令）。
- 预判技术：预判字符串长度，在跨页时，将后页调入后执行读字符串指令。
- 替换技术不能出现"颠簸"现象。在调入后页时，不能把前一页调出。

3.5.4 加快地址变换的方法

1. 虚拟MEM地址变换的速度

* 虚地址→主MEM实地址，访问实存一次变换一次→速度要求快。
* 虚地址→辅存实地址，页面失效时进行→速度可较低。页面替换→只有页面失效且主存已满时出现→速度要求可低一些。
* 虚拟MEM速度取决于虚地址→主存实地址速度。

2. 虚地址—主存实地址时间

* 页表法：查页表时间+访存时间→访存两次。
* 段页法：查段表+查页表+访存时间→访存三次。
* 相联目录表法：对2NV行表全相联查表+访主存。

3. 快表和慢表速度

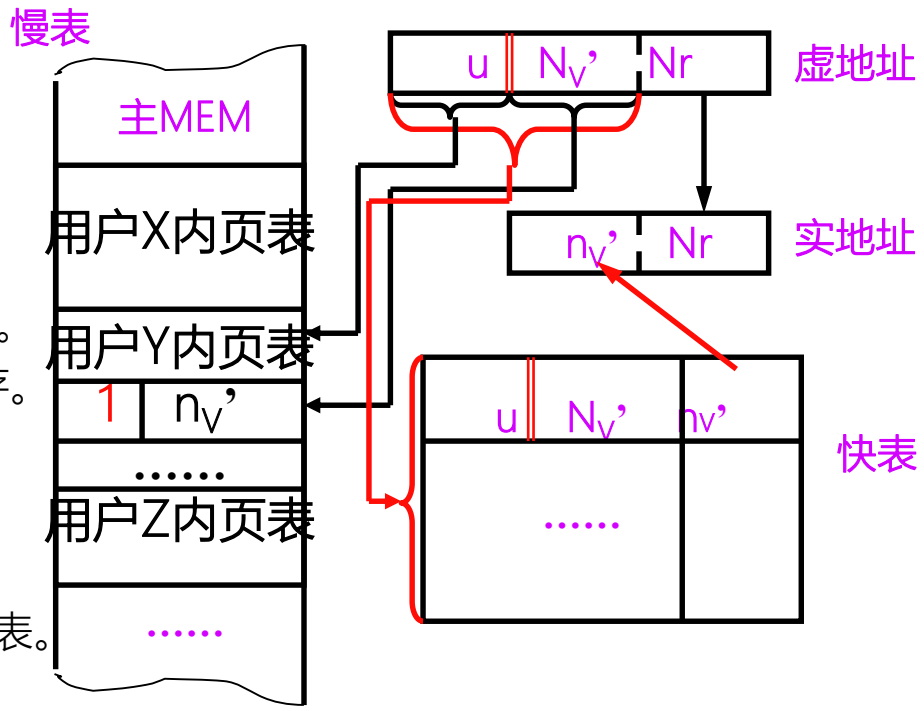
∵在一段时间内，查表只用到少数n行，

∴用快速硬件构成8-16行的部分目录表→快表。

同时，在主MEM中建立一个完整的内存页表→慢表。

* 快表只是慢表中一个小的子集。

4.快表与慢表工作过程



3.5.4 加快地址变换的方法

* 虚地址→

→硬件查快表（8-16行）→（查到n_V→实地址Reg）/（查不到→某慢表结果）。

→访问MEM，查慢表→

→查到结果n_V→（快表已有，则作废）/（快表没有，则送实Reg，将此虚地址和n_V送快表）

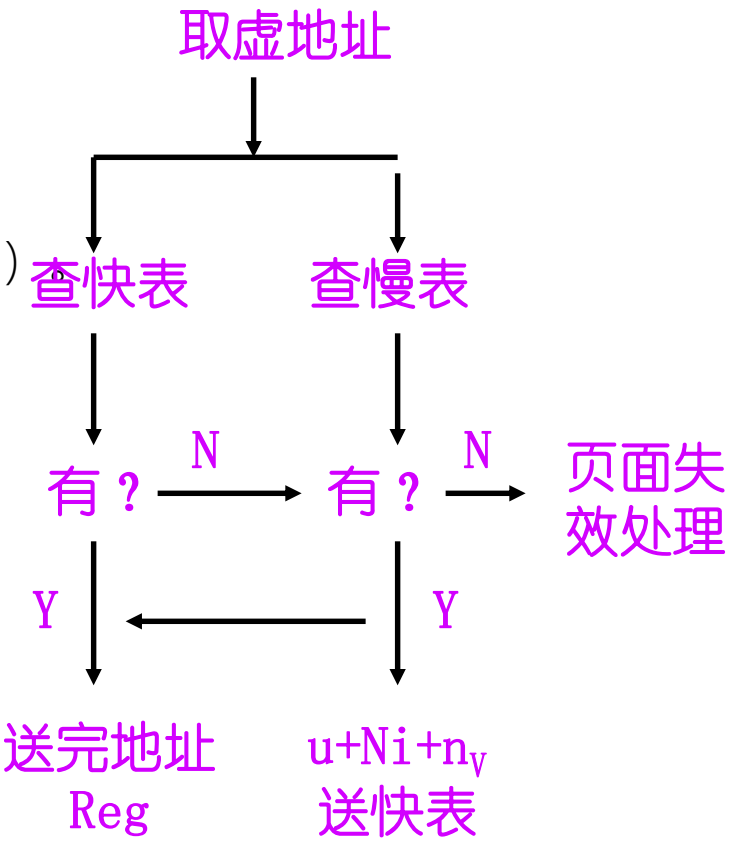
→查不到→页面失效处理。

* 快表中内容由慢表提供，且用LRU方法替换。

5.快表的查找

顺序查找

散列函数



3.5.4 加快地址变换的方法

6.快表实现—IBM370快表实现

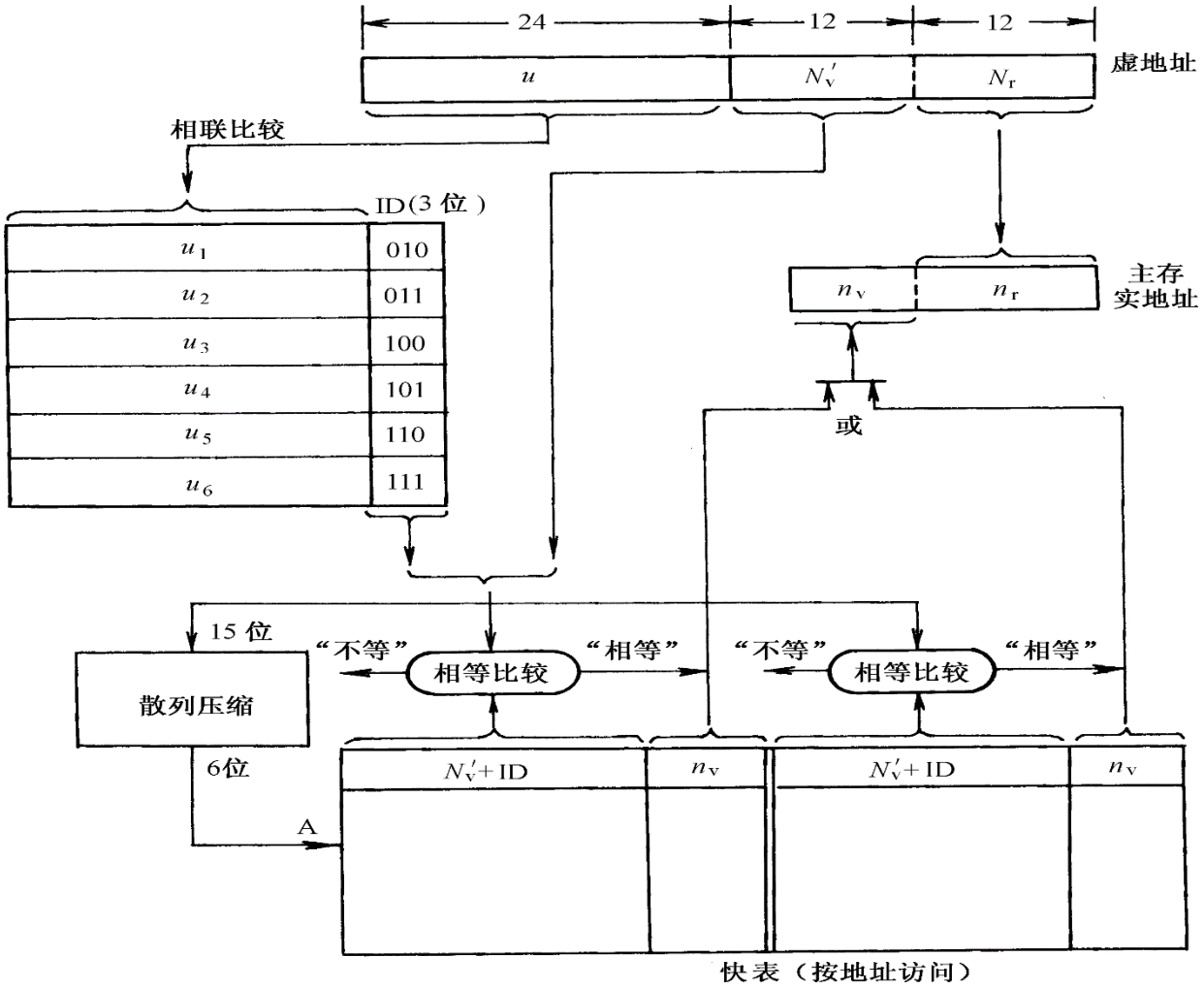


图3-46 IBM 370/168的虚拟存储器快表结构

3.5.5 虚拟存储器性能分析

1. 主存空间利用率N

* 定义：用户程序占据的主存空间与分配给该用户的主存容量之比。

* 用户程序平均长度S_S，页面平均长度S_P

当S_S》S_P时，页内零头平均值为$S_P/2$，

在主存内的页表（内、外页表）：

S_S/S_P，

∴非程序空间：

$$S_U=S_P/2+S_S/S_P=\text{零头}+\text{页表数}$$

主存空间利用率：

$$N=S_S/(S_S+S_U)=2S_PS_S/[S_P^2+2S_S(1+S_P)]$$

∴N最大值，$N_{max}\rightarrow Su_{min}\rightarrow S_P^{OPT}$

∴S_U对S_P求导$dS_S/dS_P=1/2-S_S/S_P^2=0$，

$S_P^2=2S_S$

∴代入N式，化简。

见P105图3.47。

* 实用S_P大小确定，可以从以下角度考虑：

(1)从用户程序占用S_S出发，确定S_P^{OPT}，

N^{OPT}。

(2)∵对磁盘查找时间》传送一页内容时间，

∴不论页面大小查找时间一样。

(3)从指令、操作数、字符串跨页存贮概率来调整S_P

(4)从OS为页面替换时间→ S_P↓。

(5)从指令命中率→对S_P↑。

3.5.5 虚拟存储器性能分析

2.主存命中率H

- 对用户程序分配主存S_I一定时
- S_P↑→H↑，当S_P达到某值后，总页数减少→S_P↑→H↓。
- 原因：若任务地址流A相邻逻辑地址之间距离dr。

(1)在S_P较小时，若dr＜S_P，则随S_P↑→H↑。

(2)若dr＞S_P，在将S_P↑→对于堆栈型LRU，在分配主存容量一定时→总页数=总容量S_I/S_P↓→页面失效↑→H↓。

∴在S_P较小时，原因(1)起作用；

当S_P ＞某值，则原因(2)起作用。

3.5.5 虚拟存储器性能分析

对用户程序分配主存S_1一定时。

- S_P↑→H↑，当S_P达到某值后，总页数减少→S_P↑→H↓。
- 原因：若任务地址流A相邻逻辑地址之间距离dr。

  (1)在S_P较小时，若dr < S_P，则随S_P↑→H↑。

  (2)若dr > S_P，在将S_P↑→对于堆栈型LRU，

  在分配主存容量一定时→总页数= 总容量S_I/S_P↓→页面失效↑→H↓。

∴在S_P较小时， 原因(1)起作用；

当S_P > 某值，则原因(2)起作用。

- S_I与H关系：S_I↑→H↑，但S_I太大→H达最大值，但同一时期只有一部分主存在起作用→使主存空间利用率N ↓↓。
- S_I、S_P采用折衷平衡S_P、H、N三者关系：

  S_P应以H为主选定，但兼顾N，不使N太低。

  (1) H (2) N
- 对H影响因素：

(1)替换算法 (2) S_P太小 (3) S_I太小

虚拟存储器性能分析

- 在虚拟MEM结构中CPU效率

(1) 分时系统：

分配给每个用户的CPU时间片大小影响虚拟MEM使用

CPU时间片↓→SP较大→减少页面交换次数。

(2) 多任务系统：

多任务时 在调页时

CPU与I/O处理机并行工作→

当作业数J < JOPT CPU效率↑，。

当J > JOPT时，主MEM分配的SI太小→H↓↓→CPU效率↓。

3.5.5 虚拟存储器性能分析

3.Cache-主存 - 辅存层次

CPU以虚地址访问，

若在Cache中，则把虚地址变换成Cache地址，访问Cache

- 若虚地址在主存，不在Cache，
- 则把虚地址变换成主存地址（经快表）和Cache地址，
- 访问主 存，并调入Cache
- 若不在主存和Cache内，
- 把虚地址变换成辅存地址、主存地址（经慢表）和Cache地址，
- 从辅存调入主存，再调入Cache。

3.6 主存保护与控制

3.6.1 主存保护

目的：多用户系统、多处理机系统的主存内同时存由多个用户程序和系统软件。为了防止一个用户程序出错破坏主存内系统用户软件和其他用户程序，及防止用户程序超出自己的主存区域非法访问。
包括存储区域保护和访问方式的保护

（一）区域保护

可以限定上下界，缺点：需要占用连续主存区域。而虚拟存储系统各页离散分布于内存。无法使用。

1. 页表保护

每个程序有自己的页表，虚地址出错，只能影响到表内相应的实页
但对于形成地址的硬、软件出错产生非法主存实地址时，页表无法保护

- **访问方式保护**

读--R
写--W
执行—E
区域保护＋访问方式保护

2. 键式保护

主存每页一个存储键，每个用户的存储键值不同。同时操作系统为用户确定访问键。
程序访主存必须存储键和访问键相符

3. 区域保护

把系统软件和用户程序按重要性及对系统能否正常工作的影响程度分层.
环号越大等级越低，级别低不能访问级别高的。
特点：保证用户程序出错不会侵犯系统软件和保护低级不会破坏高级部分

3.6.2 主存控制部件

- 主存系统除存储体外，还要有复杂的主存控制部件（存控）来完成地址变换、存储保护的检验、错误检测及错误纠正、协调多个CPU和多个I/O处理机（通道）对主存的访问等功能。
- 不同的计算机对存控的安排不同，有的把上述某些功能（如主存地址形成和读出信息的合并或分离）交由处理机实现。但是，不论放在哪里，上述这些控制功能是必须具备的。
- 改善系统吞吐能力、提高系统速度和可靠性的不可缺少的部件，也是存储体系能有效工作的重要保障。

3.6.3 磁盘冗余阵列

- 磁盘阵列（Disk Array）可以有效提高存储系统的可靠性和性能。

- 它将数据分布到多个磁盘上（即数据带状分布），
- 可以同时访问几个磁盘，提升了吞吐率；将数据在不同的磁盘上进行备份提高可靠性。

- RAID（Redundant Array of Inexpensive Disk，廉价磁盘冗余阵列）

本章小结：

存储系统的定义及层次结构

地址映像和变换方法

替换算法及其实现

并行主存系统

高速缓冲存储器（Cache）

虚拟存储系统

存储器性能评价

主存保护和控制