第二章 并行主存与存储体系

目的：存放计算机系统中所需要处理的程序与数据 。

主存储器：用以存放正在运行的程序与数据 。

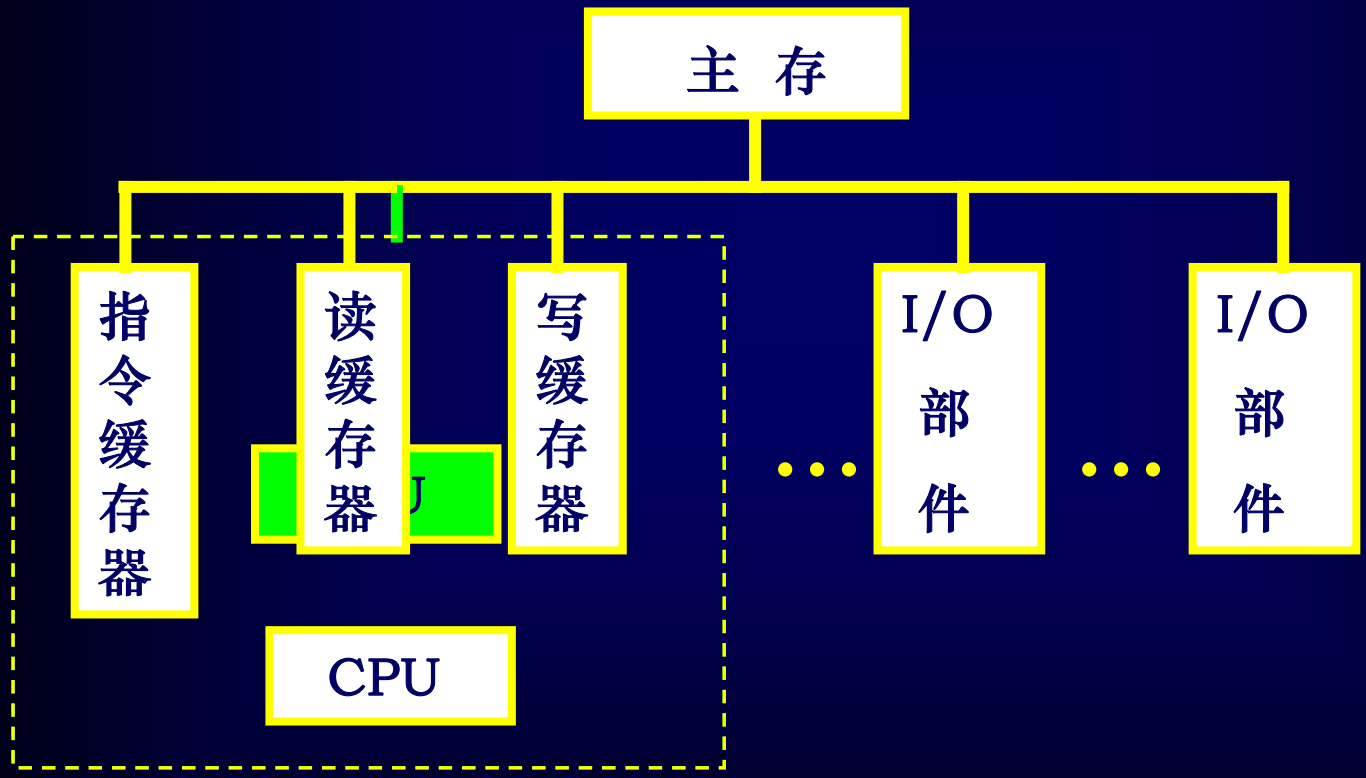
辅助存储器：存放等待运行的程序与数据。

通用寄存器组：是存放那些最经常用到的数据。

存储系统：两个或两个以上的速度、容量、价格不同的存储器采用硬件，软件或软、硬件结合的办法联接成一个系统。

一、存储系统概述

1、存储系统的形成



以存储器为中心的计算机结构

2、存储系统的层次结构

计算机存储系统三个基本参数：

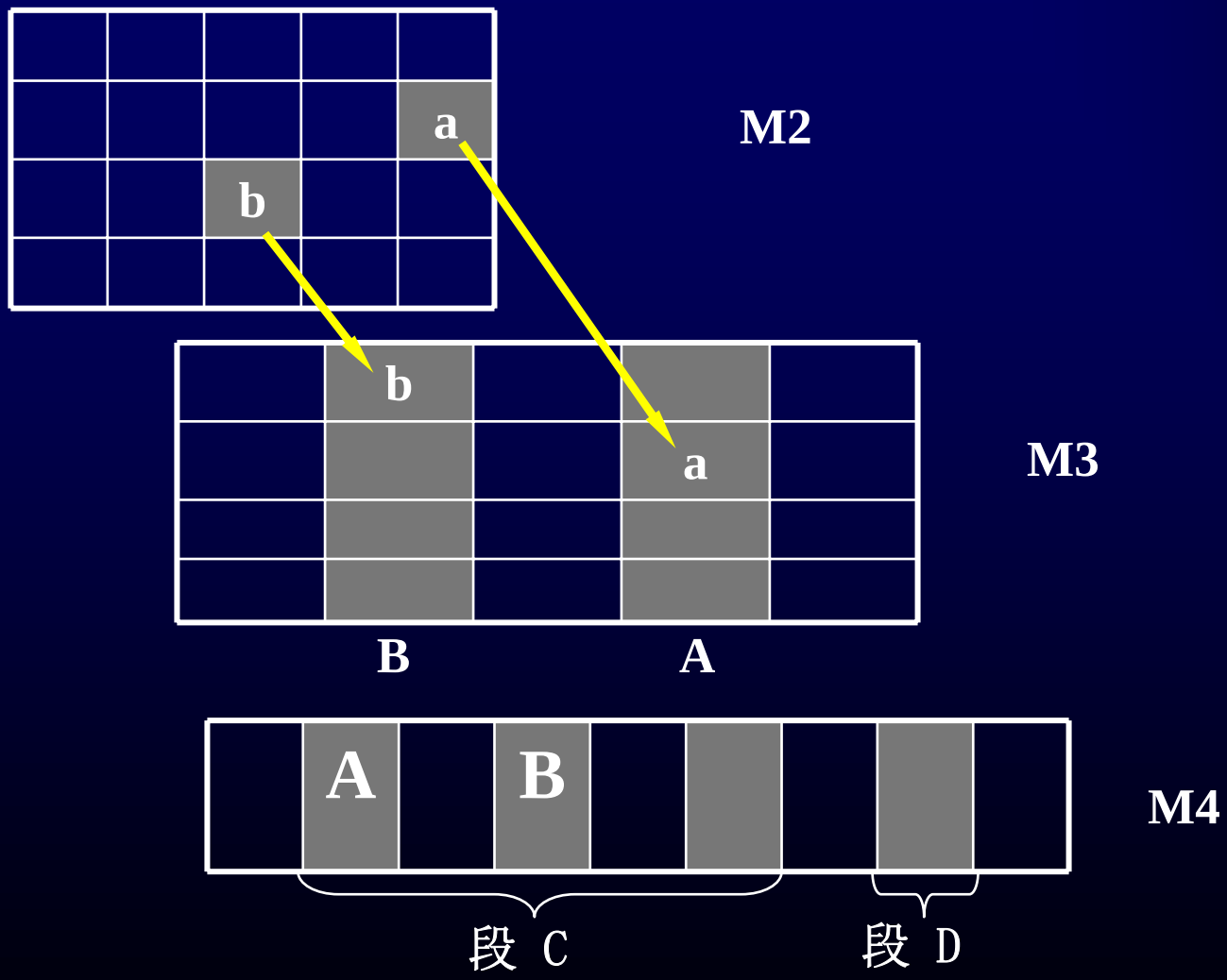
- 存储容量S　　以字节数表示，单位为B、KB、MB、GB、TB等。
- 存储器速度T　存储器访问周期，与命中率有关。
- 存储器价格C　表示单位容量的平均价值单位为 $C/bit或 $C/KB。

每级存储器的性能参数可以表示为Ti，Si，Ci。存储系统的性能可表示为：Ti<Ti+1；Si<Si+1；Ci>Ci+1。

3、存储系统的包含性，一致性

包含性 在容量大的存储器中，一定能找到上层存储信息的副本。

一致性 副本修改，以保持同一信息的一致性。



二、并行存储器

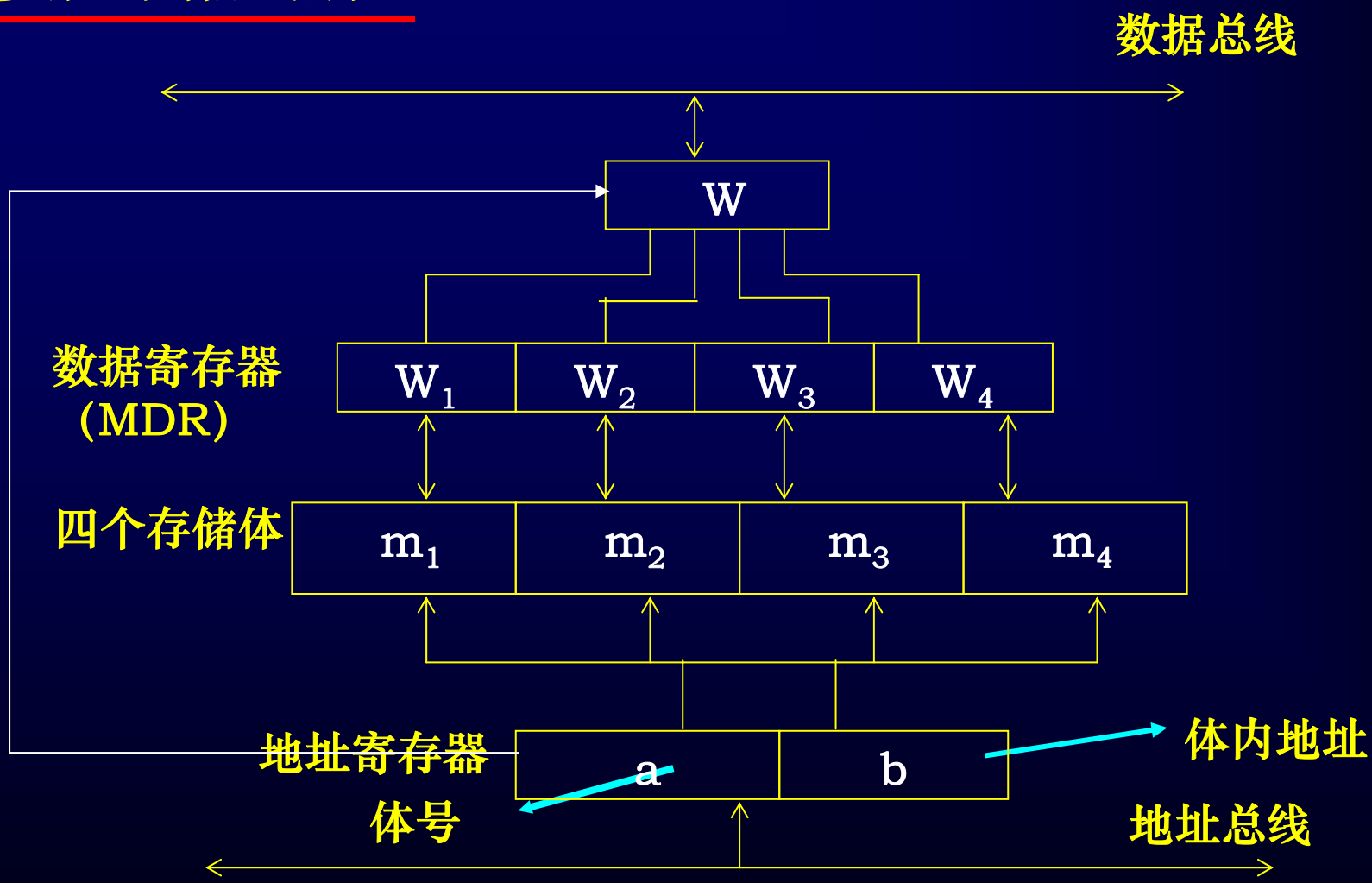
频带宽度：单位时间内所能访问的数据量。

解决频带平衡的三种方法：

- 多个存储器并行工作，并用并行访问和交叉访问等方法；
- 设置各种缓冲存储器；
- 采用Cache存储系统。

1、多体并行访问存储器

多体主存储器结构：



并行主存系统：一个主存周期内能读写多个字的主存系统。

优点：简单、容易。

缺点：访问的冲突大。

主要冲突：

- 取指令冲突（条件转移时）
- 读操作数冲突（需要的多个操作数不一定都存放在同一个存储字中）
- 写数据冲突（必须凑齐n个数才一起写入存储器）
- 读写冲突（要读出的一个字和要写入的一个字处在同一个存储字内时，无法在一个存储周期内完成）。

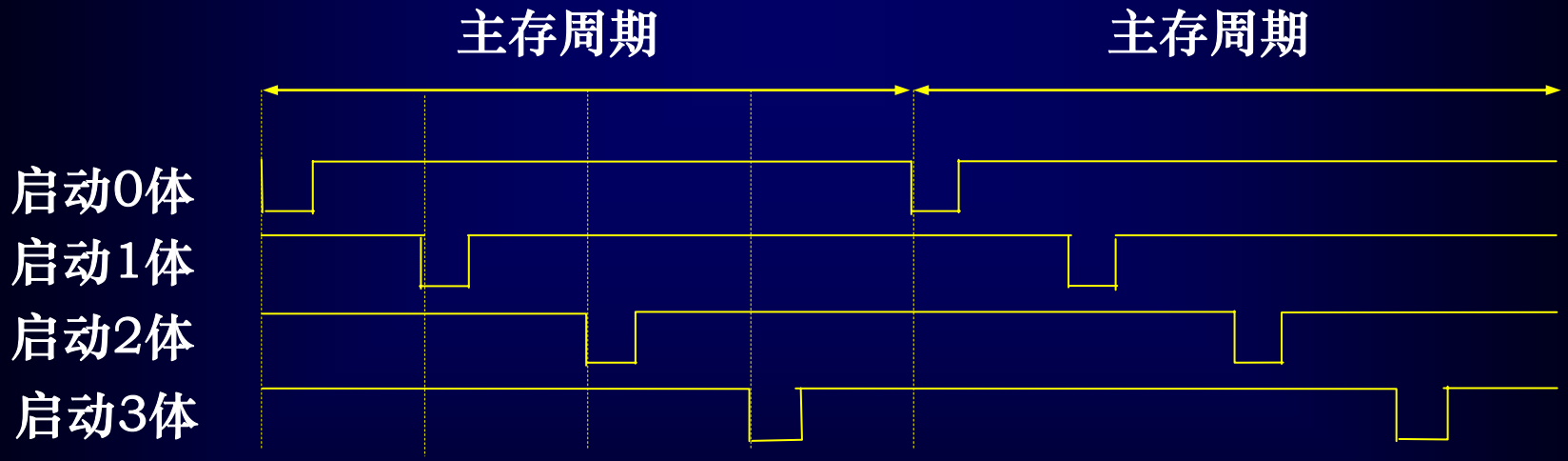
2、多体并行交叉访问存储器

模M主存储器：分为M个存储体的主存储器；

同时访问：采取同时启动，完全并行工作的方式；

交叉访问：多个访问源，同时向多个体中的任何一个发出访问请求；

分时访问：多个体采用分时启动的方式，互相错开一个存储体存储周期的1/M，交叉进行工作。



m=4 分时启动时间图

四个存储体交叉访问的时间关系

（1）高位交叉访问

m: 体数

n：每个体的容量

存储器总容量M应为M=m×n

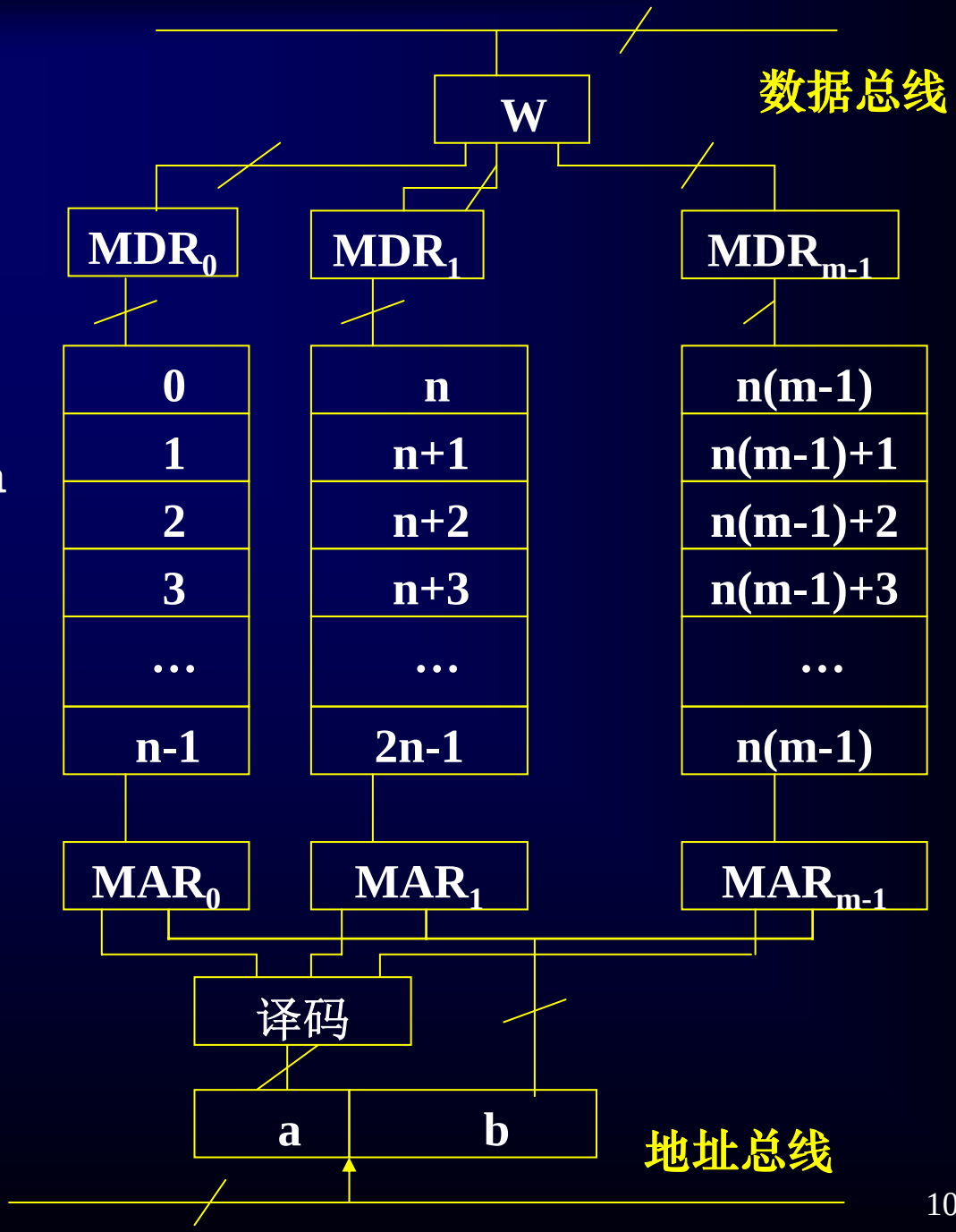
地址码长度：l=log2(m×n)

高位部分：存储体体号

$a=\log_2 m$

低位部分：体内地址

$b=\log_2 n$



缺点： 由于程序局部性的原理，近期所用到的指令和数据往往都集中在一个体内，就会出现并行访问冲突，只有一个存储模块在不停地忙碌，其他空闲。只有当指令跨越两个存储模块时，才并行工作。

优点： 扩大存储容量非常方便。如果在多任务或多用户的应用状态下，可以将不同的任务分别存放在不同的体内，减少了访问冲突，发挥了并行访问的优点。

（2）低位交叉访问

m: 体数

n：每个体的容量

高位部分：体内地址

$a=\log_2 n$

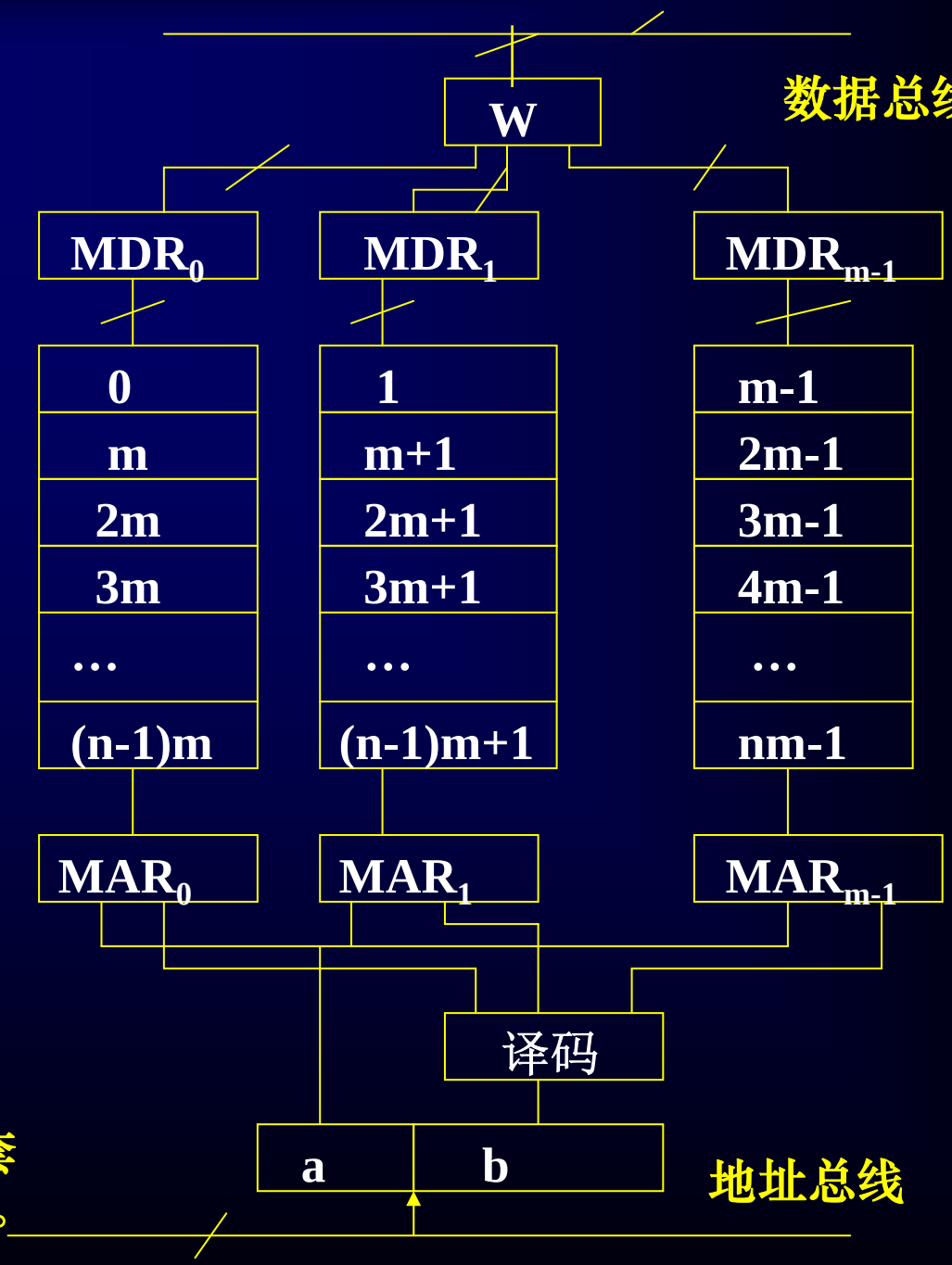
低位部分：存储体体号

$b=\log_2 m$

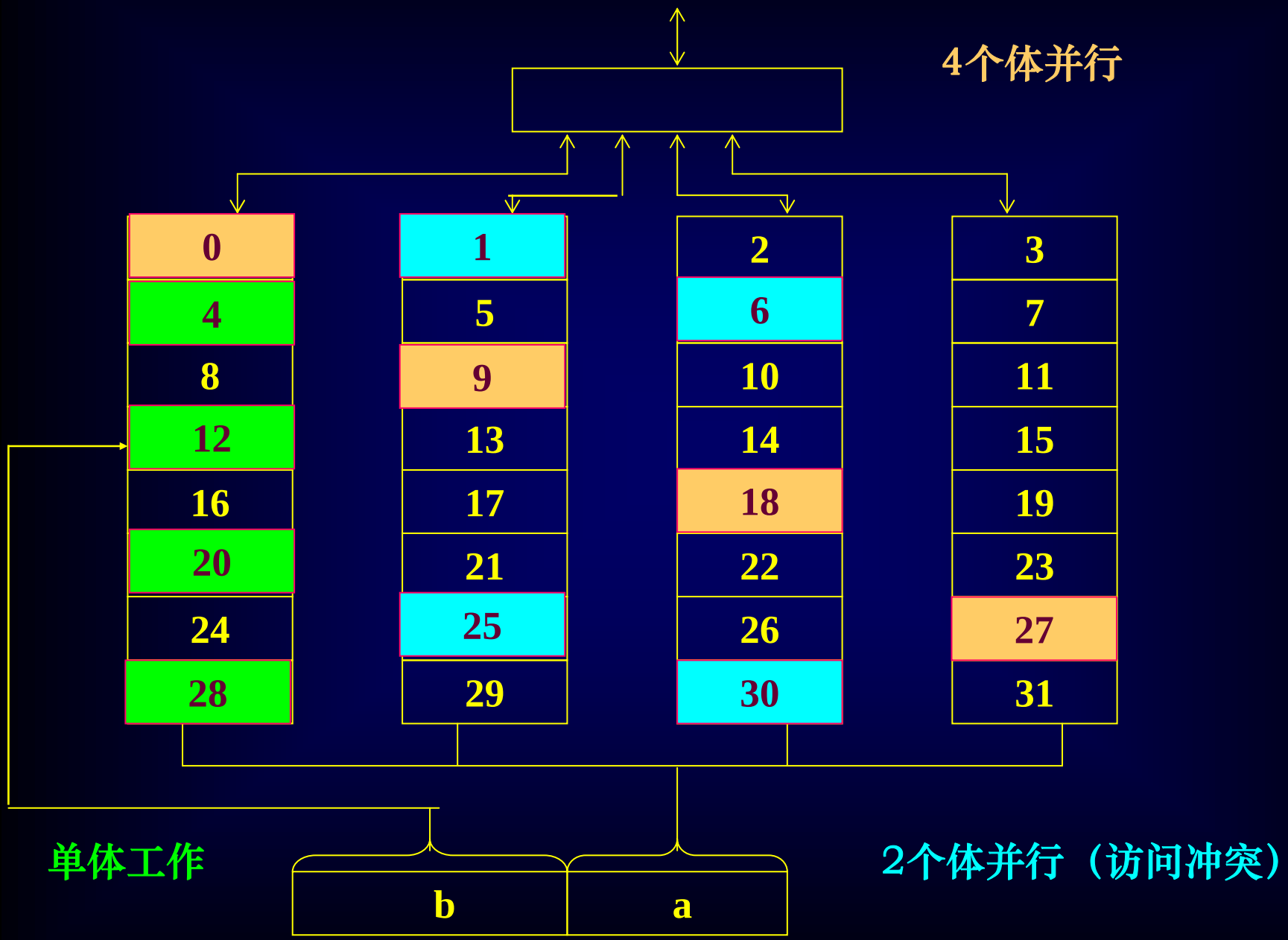
分时访问

低位交叉存储器结构

在结构上，4个体可以共用一套寻址译码驱动电路及控制部件。



例如，n=8 m=4多体并行低位交叉编址



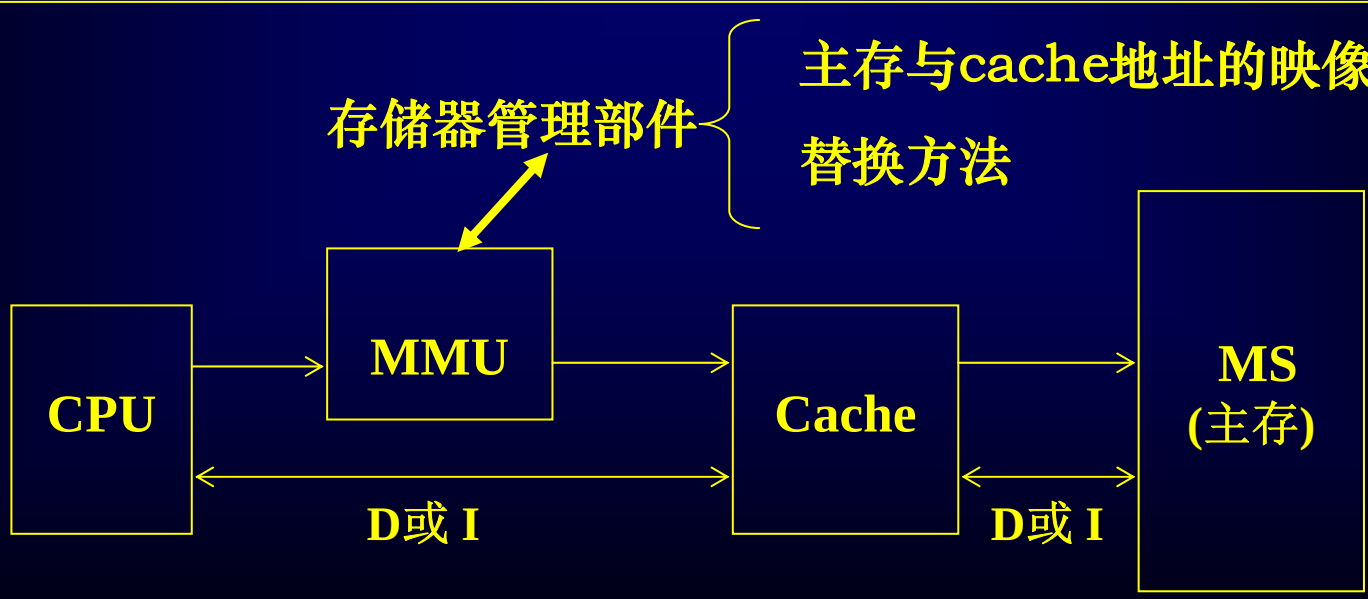
三、高速缓冲存储器CACHE

1、高速缓冲存储器的功能、结构与工作原理

高速缓冲存储器： 存在于主存与CPU之间的一级存储器，由静态存储芯片(SRAM)组成，容量比较小但速度比主存高很多，接近于CPU的速度。

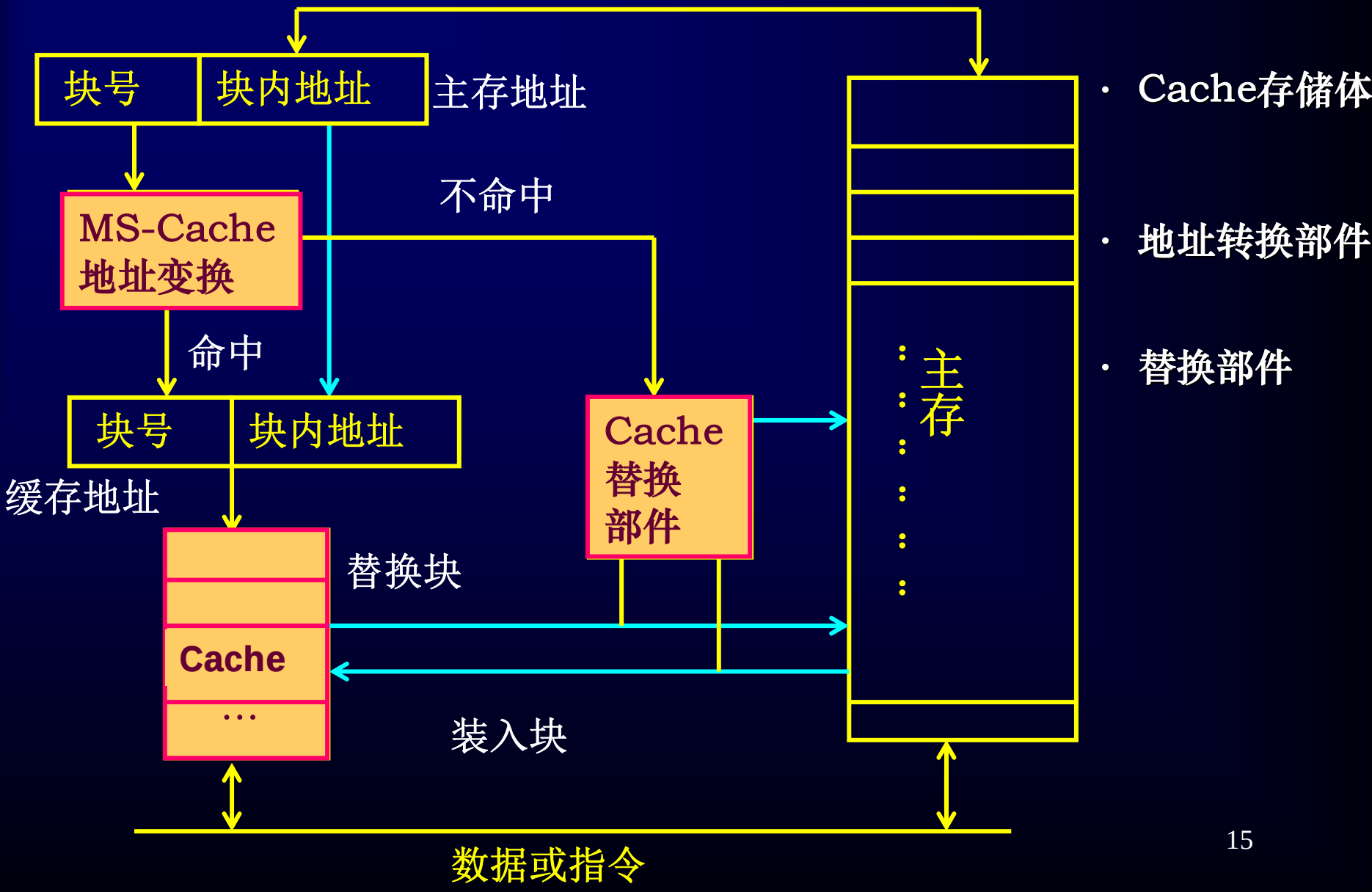
Cache 的 功 能： 存放那些近期需要运行的指令与数据。

目　　的： 提高CPU对存储器的访问速度。



CPU与Cache、主存的关系

Cache的结构


块号
块内地址
主存地址
不命中
MS-Cache
地址变换
命中
块号
块内地址
缓存地址
Cache
替换
部件
主存
替换块
Cache
…
装入块
数据或指令
· Cache存储体
· 地址转换部件
· 替换部件

2、地址映象与转换

- 地址映象是指某一数据在主存中的地址与在缓存中的地址两者之间的关系。

三种地址映象：

- 全相联方式
- 直接相联
- 组相联

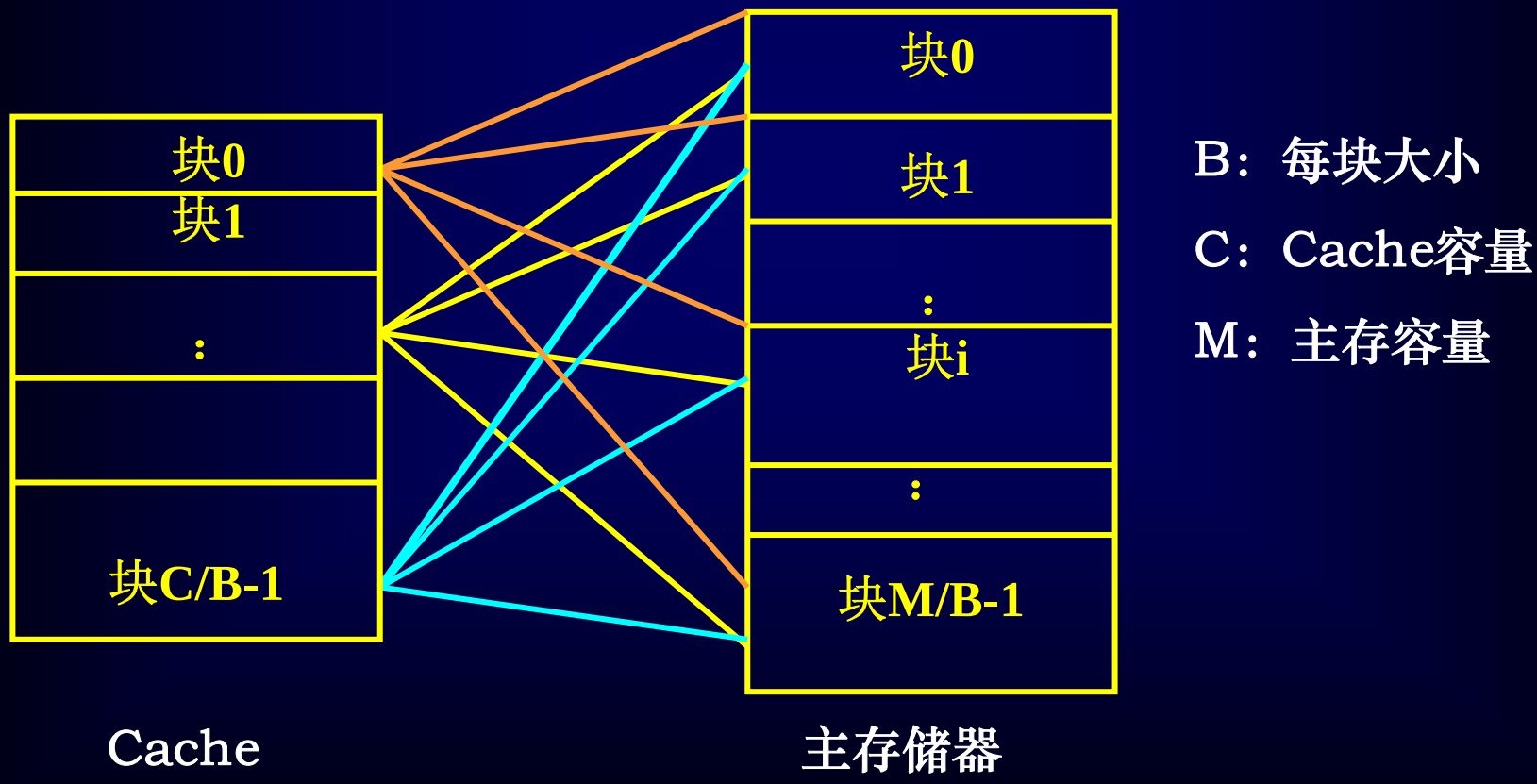
地址映象与转换是在两个不同的时间进行的，在数据由主存调入缓存时，按照地址映象规则建立地址映象表，即目录表在CPU访问该数据时，根据地址目录表的记录，进行地址转换。

全相联的地址映象规则：

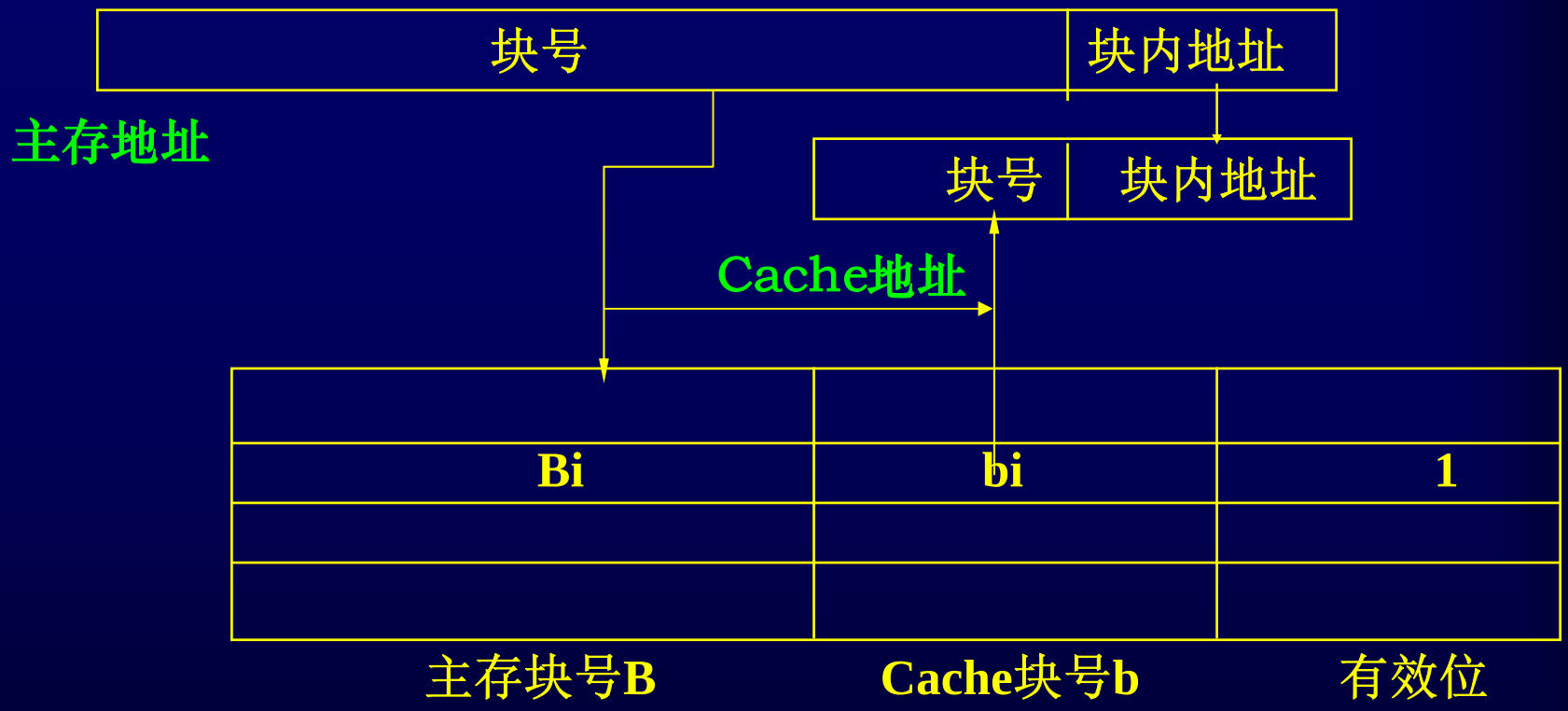
1) 主存与缓存分成相同大小的数据块。

2) 主存的某一数据块可以装入缓存的任意一块的空间中。

全相联映象方式 ：



全相联地址转换



优点：命中率较高，Cache的存储空间利用率高；

缺点：线路复杂，成本高，速度低。

例：假设在某个计算机系统中Cache容量为64K字节，数据块大小是16个字节，主存容量是4M，地址映象为全相联方式。

优点：命中率比较高，Cache存储空间，利用率高。

缺点：相关存储器访问时， 每次都要与全部内容比较，如上例中目录表的容量为4096，如此大容量的相关存储器难以实现，而且速度低，成本也比较大，因此在缓存结构中应用的比较少。

解：画出主、

主存地

缓存地

30 13 12 1

目录表格式及容量

主存地址	缓存地址	有效位

共31位=18主存块地址+12位缓存块地址+1位有效位

容量：应与缓存块数量相同即2^{12}=4096。

直接相联方式

- 直接相联的地址映象规则

1. 主存与缓存分成同样大小的块；
2. 主存容量应是缓存容量的整数倍，将主存空间按缓存的容量分成区，主存中每一区的块数与缓存的总块数相等；
3. 主存中某区的一块存入缓存时只能存入缓存中块号相同的位置。

直接映象公式

$$b = A \bmod C/B$$

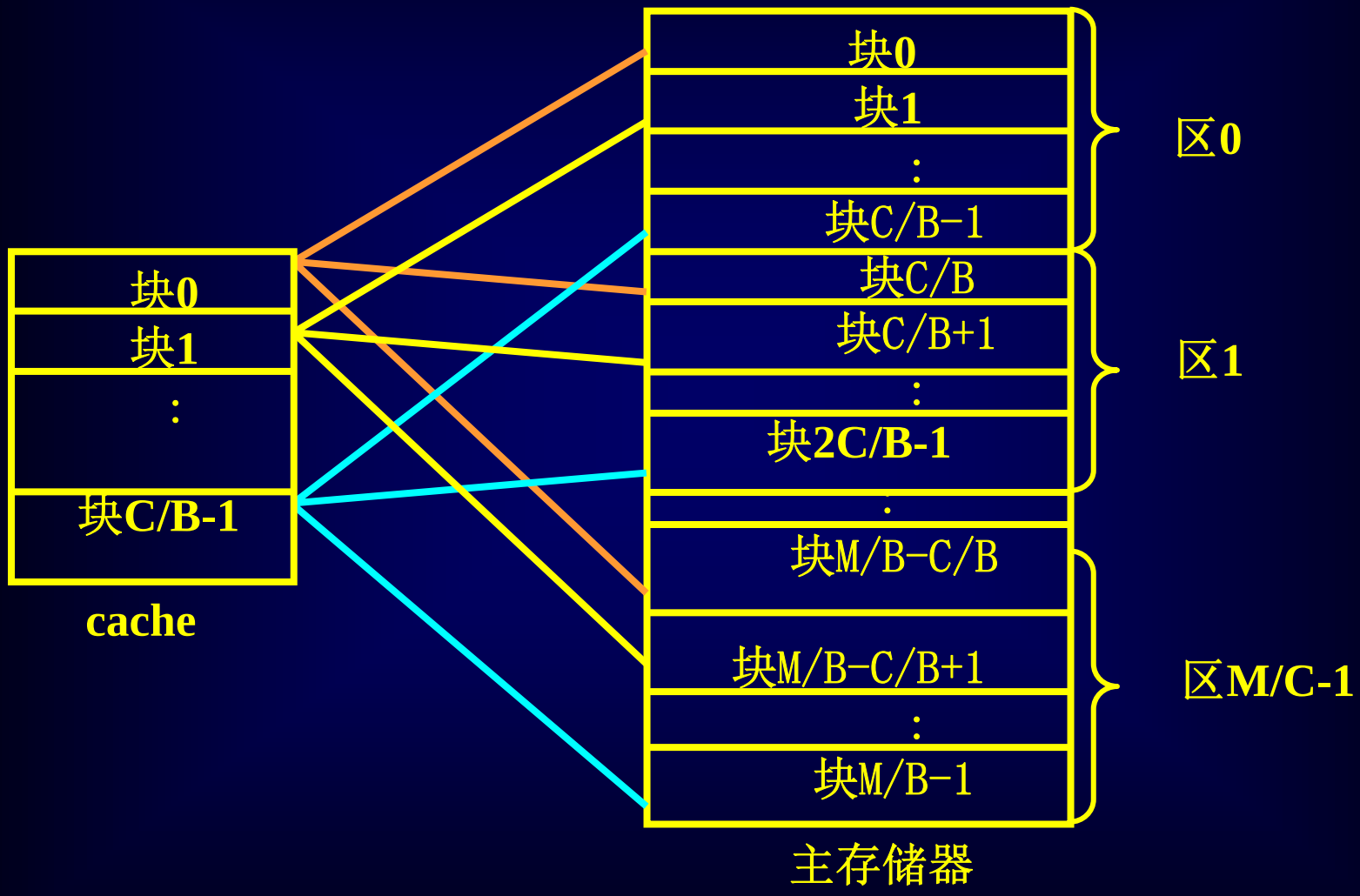
其中

b： Cache的块号

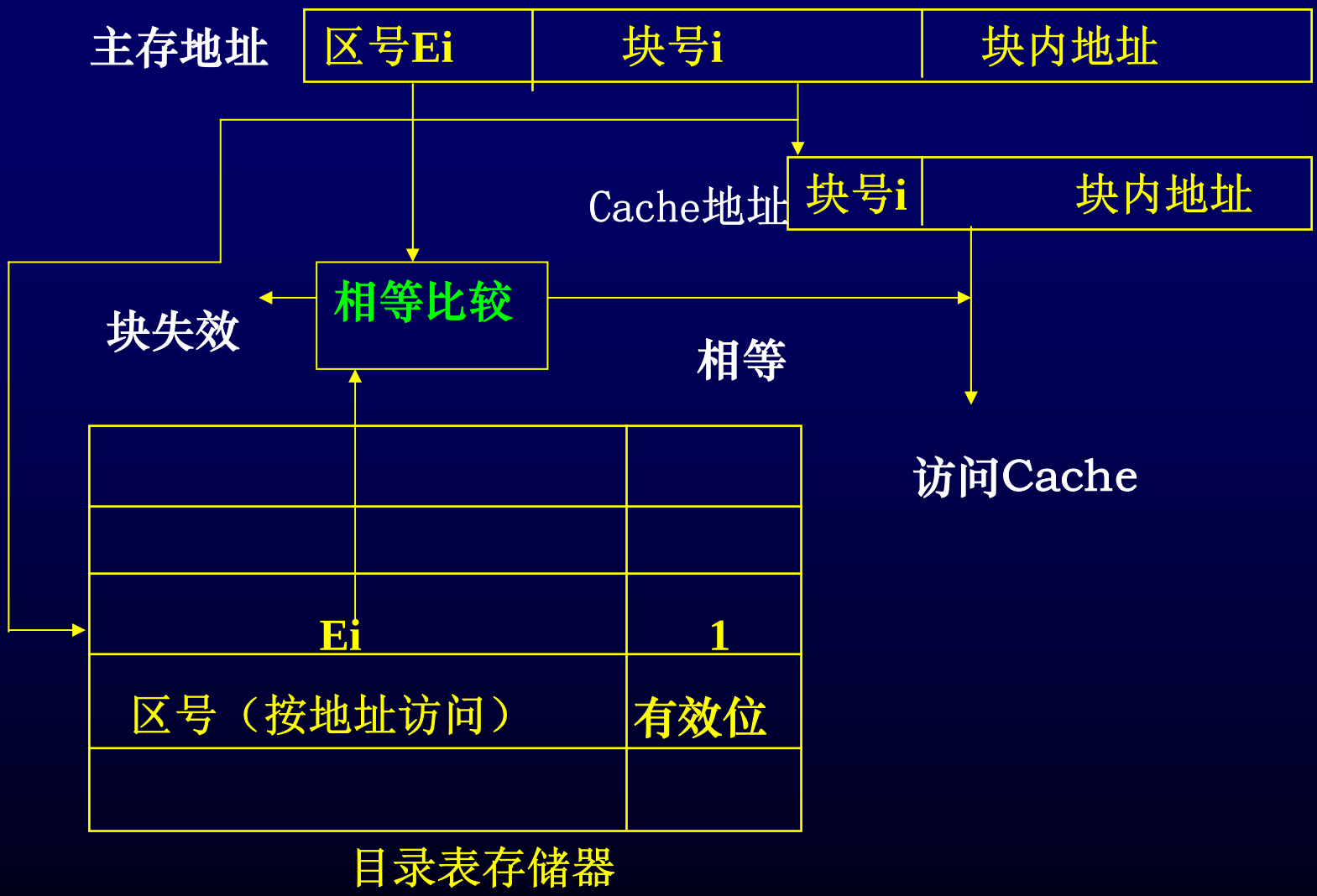
A：主存的块号

C/B：Cache的块数

直接相联地址映象


块0
块1
:
块C/B-1
cache
块0
块1
:
块C/B-1
块C/B
块C/B+1
:
块2C/B-1
:
块M/B-C/B
块M/B-C/B+1
:
块M/B-1
区0
区1
区M/C-1
主存储器

直接相联地址转换



优点：简单；缺点：命中率低。

例：假设在某个计算机系统中Cache容量为64K字节，数据块大小是16个字节，主存容量是4M，地址映象为直接相联方式。

（1）主存地址多少位？如何分配？

（2）Cache地址多少位？如何分配？

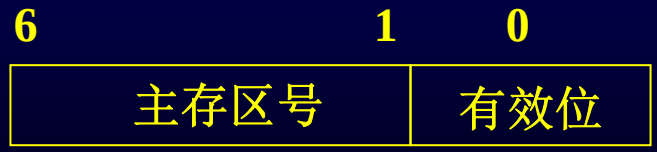
（3）目录表的格式和容量？

解：

主存地址格式：

缓存地址格式：

目录表的格式：



容量：应与缓存块数量相同即2^{12}=4096 个存储单元

组相联映象方式

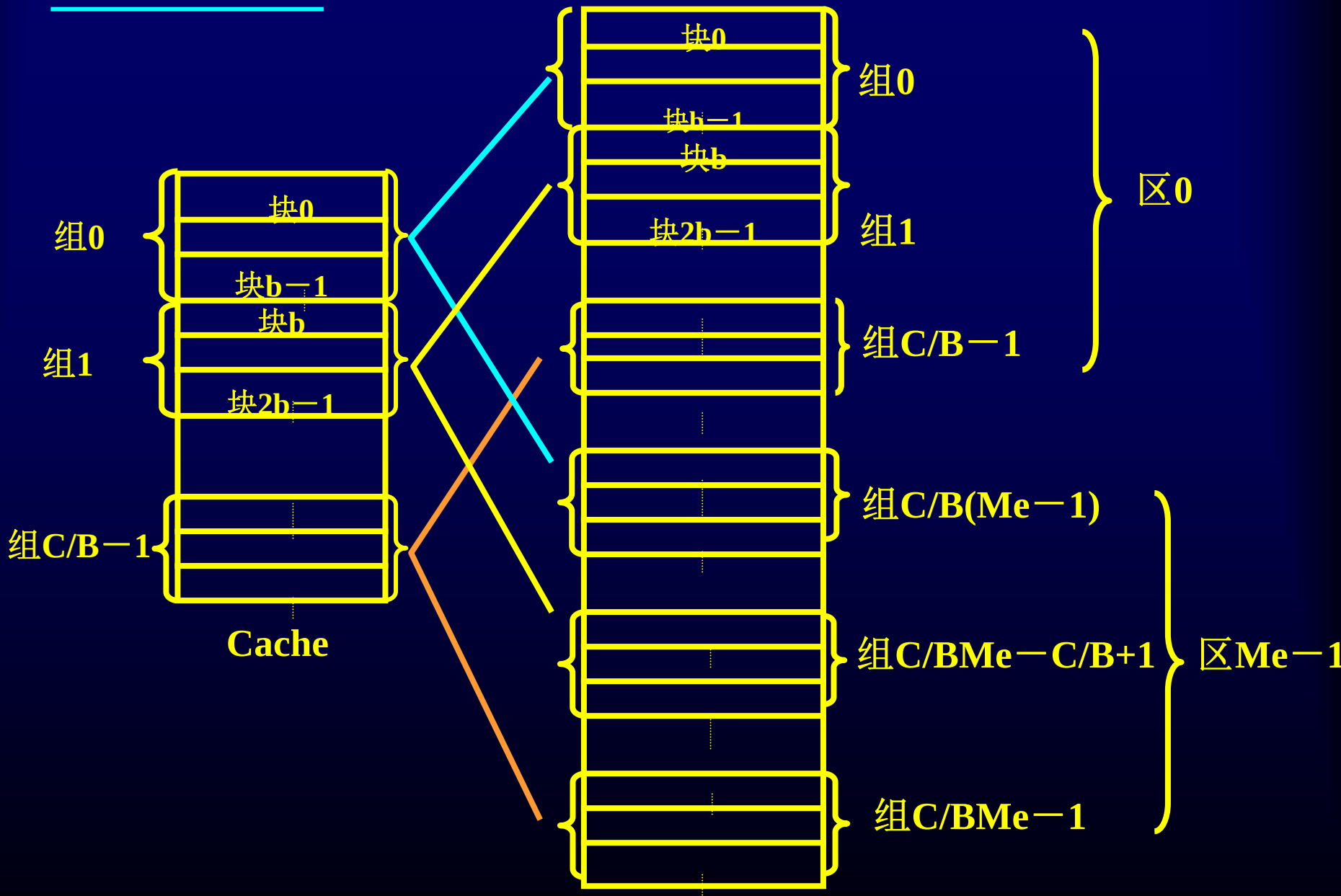
组相联的映象规则

1. 主存与缓存分成相同大小的块;
2. 主存与缓存分成相同大小的组;
3. 主存容量是缓存容量的整数倍，将主存空间按缓存的大小分成区，主存中每一区的组数与缓存的组数相同。
4. 组间直接相联；组内全相联。

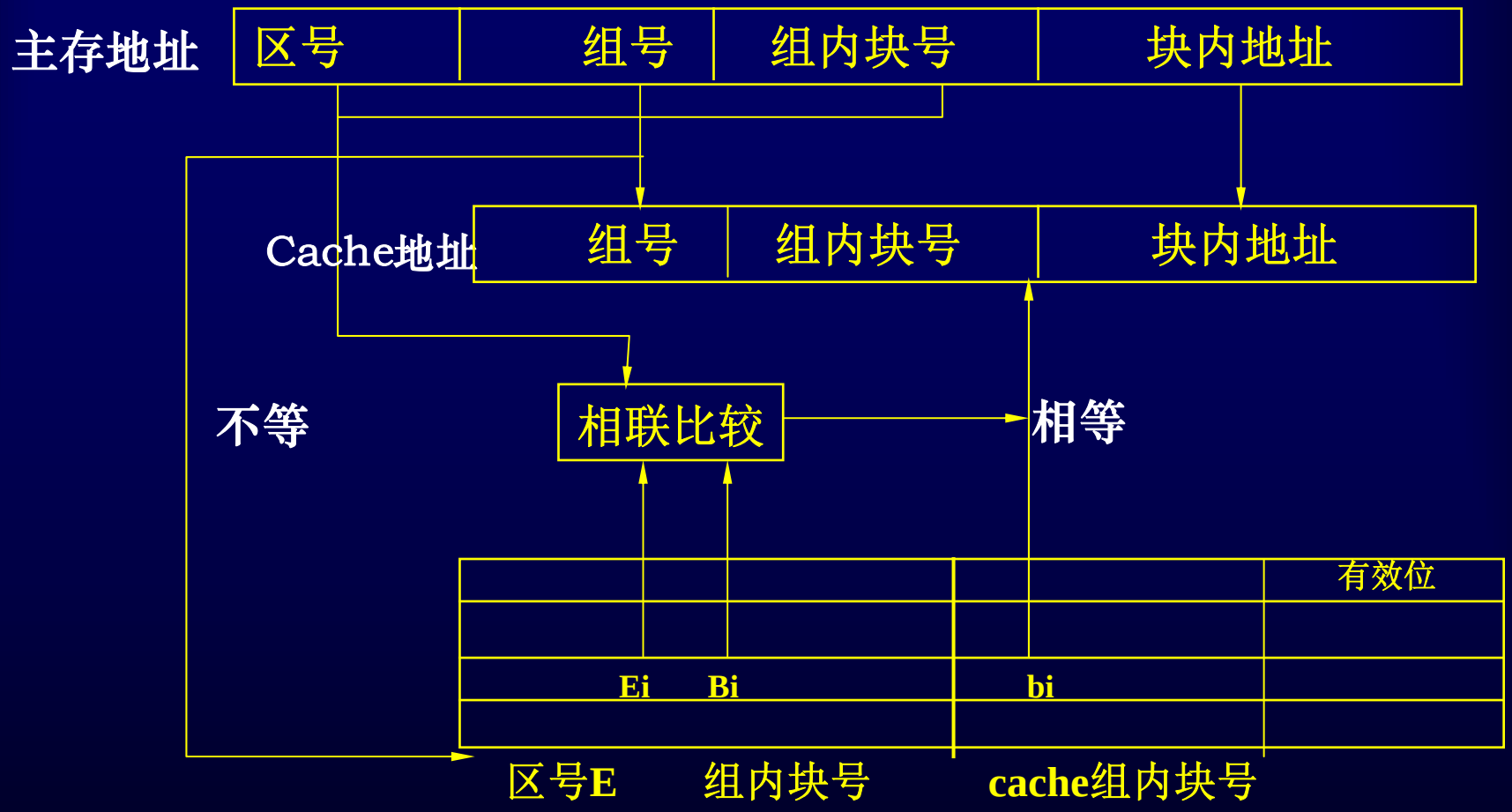
思考题：

当组相联映象的组内块数大到等于Cache的块数时，就变成什么映象？而当块数小到只有一块时就变成了什么映象？

组相联地址映象


块0
块b－1
块b
块2b－1
组0
组1
组C/B－1
Cache
块0
块b－1
块b
块2b－1
组0
组1
组C/B－1
组C/B(Me－1)
组C/BMe－C/B+1
组C/BMe－1
区0
区Me－1

组相联映象的地址转换



优点：速度快，命中率高；

例：主存容量为1MB，缓存容量为32KB，每块为64个字节，缓存共分128组。

请写出：

（1）主存与Cache的格式；

（2）相关存储器的格式与容量

解：

主存地址：

缓存地址：

相关存储器的格式：

相关存储器的容量，应与缓存的块数相同，即：组数×组内块数=2^9=128×4=512个存储单元。

3、替换策略

- 随机法：(Random,RAND法)
- 先进先出法(First-In First-Out,FIFO法)
- 近期最少使用法(Least Recently Used,LRU法)
- 最久没有使用法（Least Frequently Used,LFU）

(一) 随机法：(RAND法)

随机法是随机的确定替换的存储块。设置一个随机数产生器，依据所产生的随机数，确定替换的块。

优点：简单、易于实现；

缺点：没有依据程序局部性原理，命中率比较低。

(二) 先进先出法(FIFO法)

先进先出法是选择那个最先调入的块进行替换。

优点：考虑到了程序运行的历史状况，先进先出方法易于实现；

缺点：没有根据局部性规律。其命中率比随机法好些，但还不满足要求。

(三)最近最少使用法(LRU法)

LRU法是依据各块使用的情况，总是选择那个最近最少使用的块被替换。这种方法比较好的反映了程序局部性规律。因为最近最少使用的块，很可能在将来的近期也很少使用，所以LRU法的命中率比较高。

缺点：实现起来比较困难，它不但要记录每块使用次数的多少，而且要反映出近期使用的次数。

（四）最久没有使用法(LFU)

技术思想与LRU法是一样的只不过是实现方法上有所不同。LRU法是记录近期使用次数的多少，而LFU法是记录最久有没有使用过，所以后者实现起来比较容易。

设有一道程序，有1至5共五页，执行时的页地址流（即执行时依次用到的程序页页号）为：

2，3，2，1，5，2，4，5，3，2，5，2

若分配给该道程序的主存有3页，分别采用FIFO和LRU替换算法表示这3页的使用和替换过程。


时间t
1 2 3 4 5 6 7 8 9 10 11 12
页地址流
2 3 2 1 5 2 4 5 3 2 5 2
先进先出
FIFO
命中3次
2 2 2 2* 5 5 5* 5* 3 3 3 3*
3 3 3 3* 2 2 2 2* 2* 5 5
1 1 1* 4 4 4 4 4* 2
调进 调进 命中 调进 替换 替换 替换 命中 替换 命中 替换 替换
近期最少
使用LRU
命中5次
2 2 2 2 2* 2 2 2* 3 3 3* 3*
3 3 3* 5 5 5* 5 5 5* 5 5 5
1 1 1* 4 4 4 4* 2 2 2
调进 调进 命中 调进 替换 命中 替换 命中 替换 替换 命中 命中

注意：

- LRU也不是理想的方法，它仅仅是根据过去访存的频率估计未来的访存情况，因而只是推测的方法；
- 块命中率与地址流、块的大小和块的数量有关，应具体问题具体分析，选择适当的算法；
- 存储单元替换时应注意防止出现颠簸现象，即调入一块时将另一块调出，紧接着又需要访问刚刚调出块的数据，而将该块调入时又将上一块调出，即颠簸。

练习1：

对于一个容量为3个块的全相联Cache，假定访问的地址块号序列为 1，2，3，4，1，2，3，4，分别用FIFO算法和LRU算法，写出其队列变化情况，并得出结论。 **结论：产生颠簸现象，说明命中率与页地址流有关。**

练习2

对于一个全相联Cache，假定访问的地址块号序列为 1，2，3，4，1，2，5，1，2，3，4，5，在先进先出替换方式下，分别写出分配给程序的主存页面是3页和4页的情况下，其队列的变化情况，并得出结论。

结论：FIFO算法不是堆栈型算法，主存页数增加，命中率反而下降。

时间t	1	2	3	4	5	6	7	8
页地址流	1	2	3	4	1	2	3	4

先进先出 FIFO

无命中

1	1	1*	4	4	4*	3	3
	2	2	2*	1	1	1*	4
		3	3	3*	2	2	2*
调进	调进	调进	替换	替换	替换	替换	替换

近期最少使用LRU

无命中

1	1	1*	4	4	4*	3	3
	2	2	2*	1	1	1*	4
		3	3	3*	2	2	2*
调进	调进	调进	替换	替换	替换	替换	替换

LRU和LFU替换算法的实现

计数器方法：缓存的每一块都设置一个计数器，计数器的操作规则是：

1．被调入或者被替换的块，其计数器清“0”，而其他的计数器则加“1”；

2．当访问命中时所有块的计数值与命中块的计数值进行比较，如果计数值小于命中块的计数值，则该块计数值加“1”；如果块的计数值大于命中块的计数值，则数值不变，而命中块的计数器清为0;

3. 需要替换时，则选择计数值最大的块被替换。

✓计数器方法

主存访问块地址	块4		块2		块3		块5	
	块号	计数器	块号	计数器	块号	计数器	块号	计数器
Cache块0	1	10	1	11	1	11	5	00
Cache块1	3	01	3	10	3	00	3	01
Cache块2	4	00	4	01	4	10	4	11
Cache块3	空	××	2	00	2	01	2	10
操作	起始状态		调入		命中		替换	

寄存器栈法：

设置一个寄存器栈，其容量为Cache中替换时参与选择的块数。如在组相联方式中，则是同组内的块数。堆栈由栈顶到栈底依次记录主存数据存入缓存的块号，现以一组内4块为例说明其工作情况，

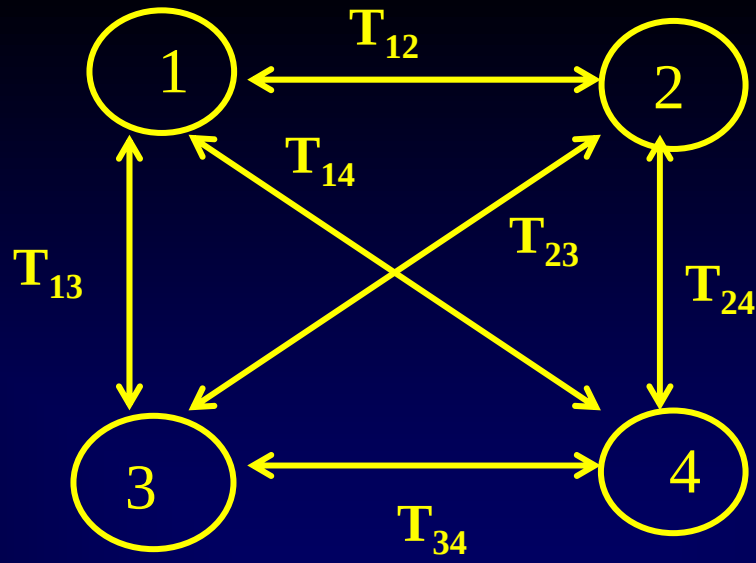
1）当缓存中尚有空闲时，如果不命中，则可直接调入数据块，并将新访问的缓存块号压入堆栈，位于栈顶。其他栈内各单元依次由顶向下顺压一个单元，直到空闲单元为止。

2）当缓存已满，如果数据访问命中，则将访问的缓存块号压入堆栈，其他各单元内容由顶向底逐次下压直到被命中块号的原来位置为止。如果访问不命中，说明需要替换。此时栈底单元中的块号即是最久没有被使用的。所以将新访问块号压入堆栈，栈内各单元内容依次下压直到栈底。自然，栈底所指出的块被替换。

✓寄存器栈法

缓存操作	初始状态	调入2	命中块4	替换块1
寄存器0	3	2	4	1
寄存器1	4	3	2	4
寄存器2	1	4	3	2
寄存器3	空	1	1	3

✓比较对法



✓假设Cache的每组中有4块，在替换时，是比较4块中哪一块是最久没使用的。4块之间两两相比可以有六种比较关系。

✓如果每两块之间的对比关系用一个触发器表示，则可以有6个触发器：T_{12}，T_{13}，T_{14}，T_{23}，T_{24}，T_{34}。

✓设定T_{12}=0，说明块1比块2最久没使用；T_{12}=1说明块2比块1最久没有被使用。

✓在每次访问命中，或者新调入块时，与该块有关的触发器的状态都要进行修改。

	T_{12}	T_{13}	T_{14}	T_{23}	T_{24}	T_{34}
块1	0	0	0			
块2	1			0	0	
块3		1		1		0
块4			1		1	1

被替换的编码表

写成表达式为各块的被替换编码为：

$$\text{Cache1} = \overline{T_{12}} \cdot \overline{T}_{13} \cdot \overline{T}_{14}$$

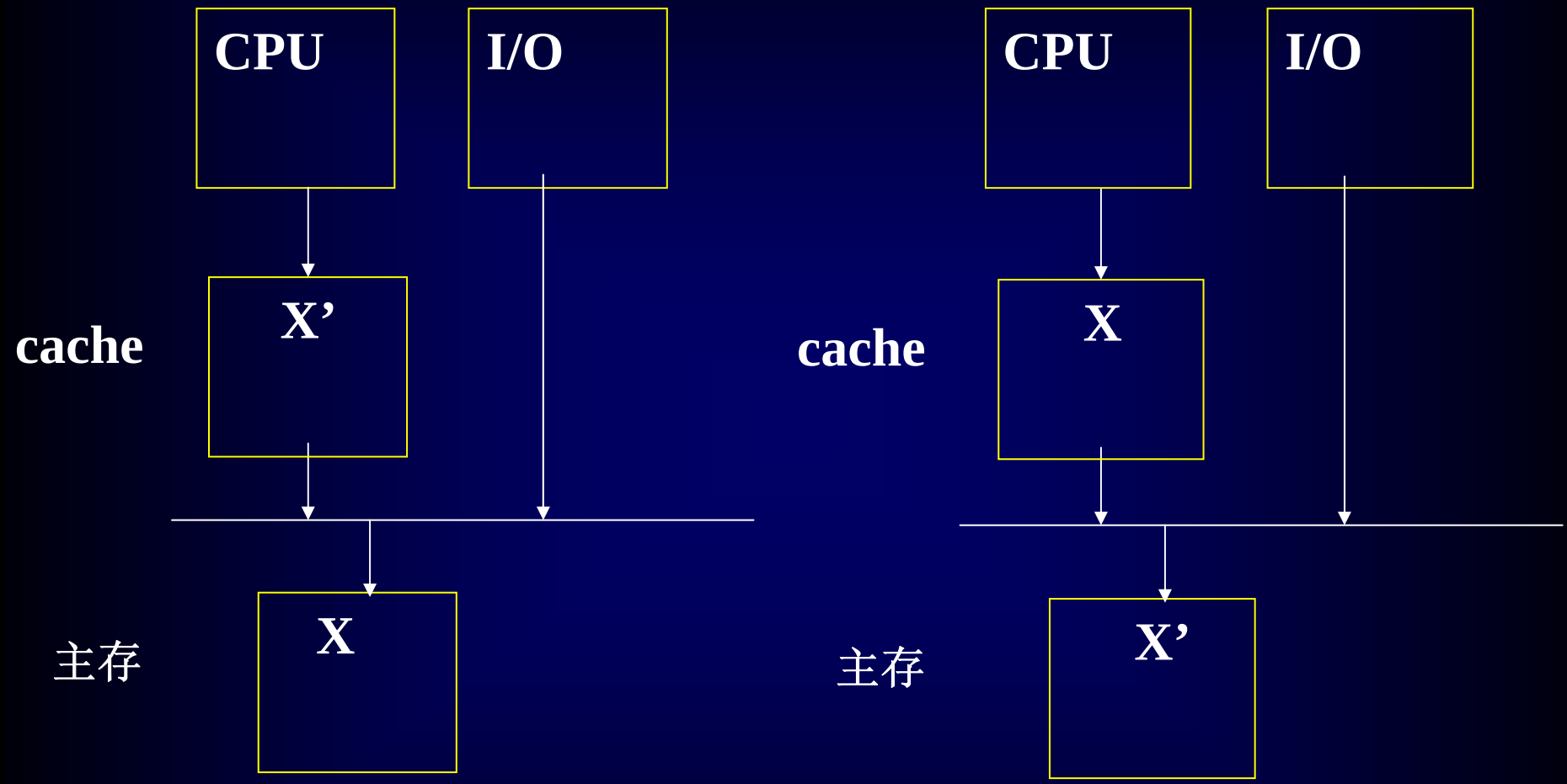
$$\text{Cache2} = T_{12} \cdot \overline{T}_{23} \cdot \overline{T}_{24}$$

$$\text{Cache3} = T_{13} \cdot T_{23} \cdot \overline{T}_{34}$$

$$\textbf{Cache4} = T_{14} \cdot T_{24} \cdot T_{34}$$

根据逻辑表达式可以设计硬件逻辑的部件。

4、 Cache的写操作



Cache与主存不一致的两种情况

不按写分配法

（1）全写法，亦称写直达法(WT法——Write through)：在对Cache进行写操作的同时，也对主存该内容进行写入 。

优点：Cache及主存与内容同时更新。所以一致性保持的比较好，可靠性比较高，操作过程比较简单。如果Cache发生错误，可以从主存得到纠正。

缺点：全写法每次写操作都要访问主存，所以，写操作的速度得不到改善，仍然是访主存的速度。

为改善这一缺点，通常采用一个高速的小容量的缓冲存储器，将要写入的数据和地址先写到这个缓冲存储器，使得Cache可以尽快的执行下次访问，然后再将缓冲存储器的内容写入主存。

按写分配法

（2）写回法(WB法——Write back)：在CPU执行写操作时，只写入Cache，不写入主存；需要替换时，把修改过的块写回主存。

优点：Cache的速度比较高，Cache的命中率比较高，所以Cache与主存之间的通信量大大降低。

缺点：有一段时间Cache内容与主存内容不一致，所以可靠性比全写法差，而且控制操作比较复杂。

改善：写回法在替换时要将一块内容写回主存；此时CPU不能继续访问Cache及主存，可能处于等待状态。为改善这一缺陷，可以设置一个高速的数据缓冲存储器，先将写入的数据与地址存入此缓冲区，以使CPU可以继续工作。而缓冲存储器可以与CPU的处理工作并行，将数据写入主存。

（3）写不命中：

不按写分配法	按写分配法
直接将数据写入主存，不调入缓存	数据写入主存，并将该数据调入 Cache

5、 Cache性能分析

（1）Cache透明性分析

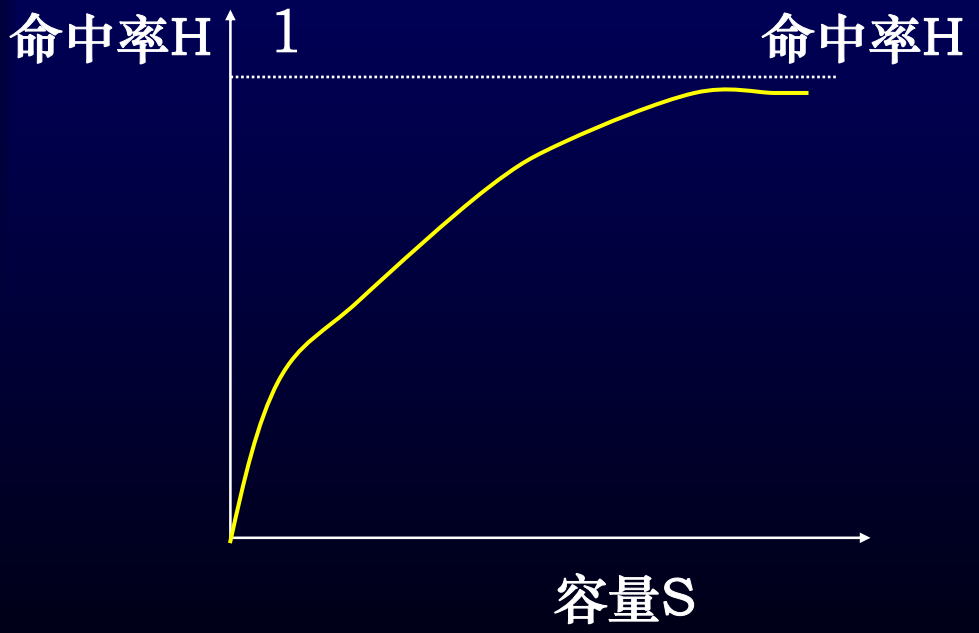
- Cache存储器的地址变换和块替换算法全由硬件实现；

- Cache-主存存储层次对应用程序员和系统程序员都是透明的；

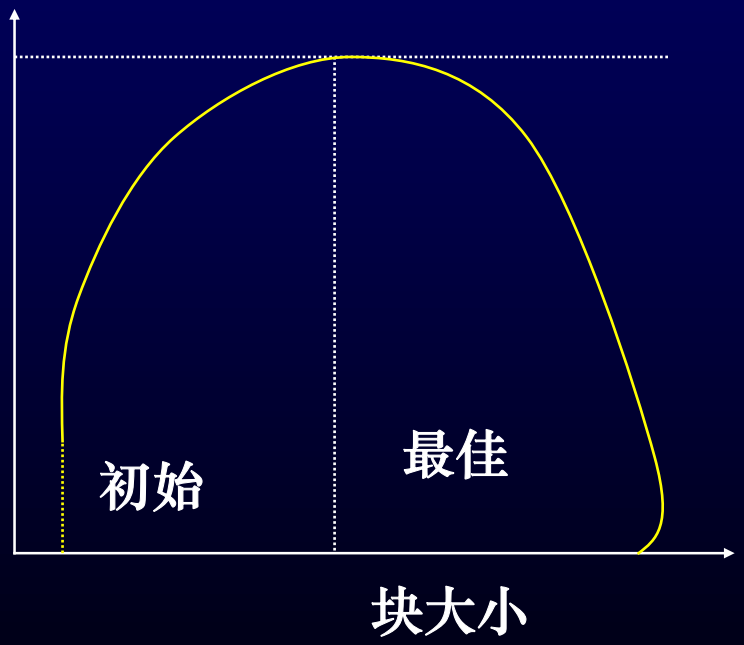
- Cache对处理机和主存之间的信息交往是透明的。

(2) Cache的命中率

1. Cache的容量对命中率的影响
2. Cache块的大小对命中率的影响
3. 地址映象方式对命中率的影响



Cache命中率H与容量S的关系



Cache命中率H与块大小的关系

（一）．Cache的容量对命中率的影响

容量越大则命中率越高。当容量由很小开始增加时命中率增加的比较明显当容量达到一定程度，容量增加命中率改善的并不大。

（二）. Cache块的大小对命中率的影响

当块的容量加大命中率明显的增加，但增加到一定值后反而出现块增加命中率下降的现象。这是因为块容量大到一定程度，进入块内的数据，已不符合程序局部性规律了；块越大在一定量的Cache中包含的块数就越小，则命中率就降低了。

（三）. 地址映象方式对命中率的影响

（1）直接相联法命中率比较低。

（2）全相联方式命中率比较高，但难以实现。

（3）组相联方式中，主要是分组的数目对命中率的影响比较明显。由于主存与Cache的组之间是直接相联方式，当组数分的越多，则命中率就要下降，当组数比较少时这种影响不明显，当组数大到一定程度，则影响就很大。

（3）Cache系统的加速比

Tc：Cache的访问周期；

Tm：主存储器的访问周期；

Hc：Cache的命中率

等效的访问周期为T

$$T = H_c T_c + (1 - H_c) T_m$$

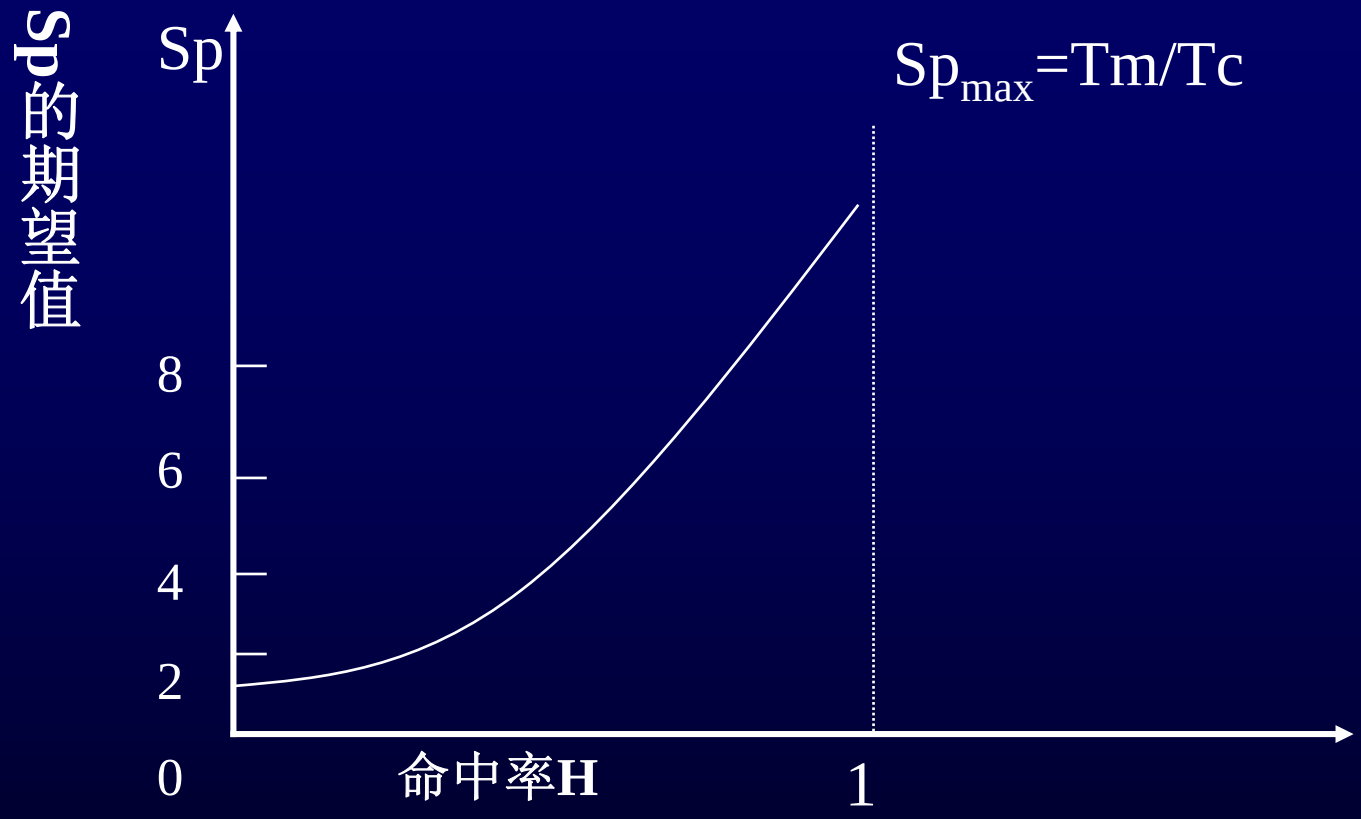
Cache系统的加速比S_p

$$S_p = \frac{T_m}{T} = \frac{T_m}{H_c T_c + (1 - H_c) T_m} = \frac{1}{H_c \frac{T_c}{T_m} + (1 - H_c)}$$

命中率越高，加速比越大。当Hc→1

$$S_p \to \frac{T_m}{T_c}$$

Cache加速比与命中率的关系



存储系统的访问效率：指高一级存储器的访问速度(容量小速度高的一级)与系统等效的访问速度之比。

$$e = \frac{T_c}{T} = \frac{1}{H_c + (1 - H_c)\dfrac{T_m}{T_c}}$$

说明：等效的访问周期越接近于Cache的Tc，则访问效率越高。将T式代入，可以看出e仍是Hc和Tm/Tc的函数。从设计者的角度看，如果的比值越大，而要求访问效率比较高，那么就要求有极高的命中率。

如果Tm/Tc的比例很小，要求具有同样的访问效率，命中率的要求就可以低得多。例如Tm/Tc =100，为使e>0.9，则命中率应满足Hc>0.998，如果Tm/Tc =2 ，同样使e>0.9，则只需Hc>0.889即可。

6、 Cache的实用举例

- （1）Cache的分体

多体存储器：分为数据体Cache与指令体Cache。

原因：

- 数据与指令不在一体可以减少多个访问源访问存储器的冲突；
- 两个体的访问操作不完全相同，数据体有读操作和写操作，而指令体只有读操作。因此在替换时，只有数据体有写回的问题。在Cache容量相等的情况下，指令与数据分体的Cache比一体化的Cache命中率要高。

（2）Cache的分级

参考书目：《计算机系统结构》——奔腾PC　张昆藏著　科学出版社

实例：**Pentium PC的Cache**

Pentium PC采用2级Cache结构。

（1） L1：1级Cache容量为16KB，集成在处理机芯片内部，采取两路组相联的地址映象方式。

（2） L2：外部有一个容量为256KB或512KB的2级Cache安装在主板上，采取两路组相联的地址映象方式。

（3） L1是L2的子集。

所谓两路组相联，是指8KB的数据Cache是由两个存储体组成，分成128组，每组内包含两块，分别在两个体中。采用LRU替换策略，每路具有自己的目录表及存储体，所以数据Cache可以在一个时钟周期内取出两个32位的数据。

特点：

（1） L1的16KB分为两个体8KB指令体与8KB数据体。统计表明：取指令占63%，取数占25%，写数占12%，说明二者分开有好处。

（2）指令Cache是只读的，单端口256位，与指令预取的缓冲器相连；

（3）数据体Cache是双端口的存储体。数据cache是读写的，双端口，每端口32位，两个端口还可组合成一个64位端口与浮点运算单元相接。

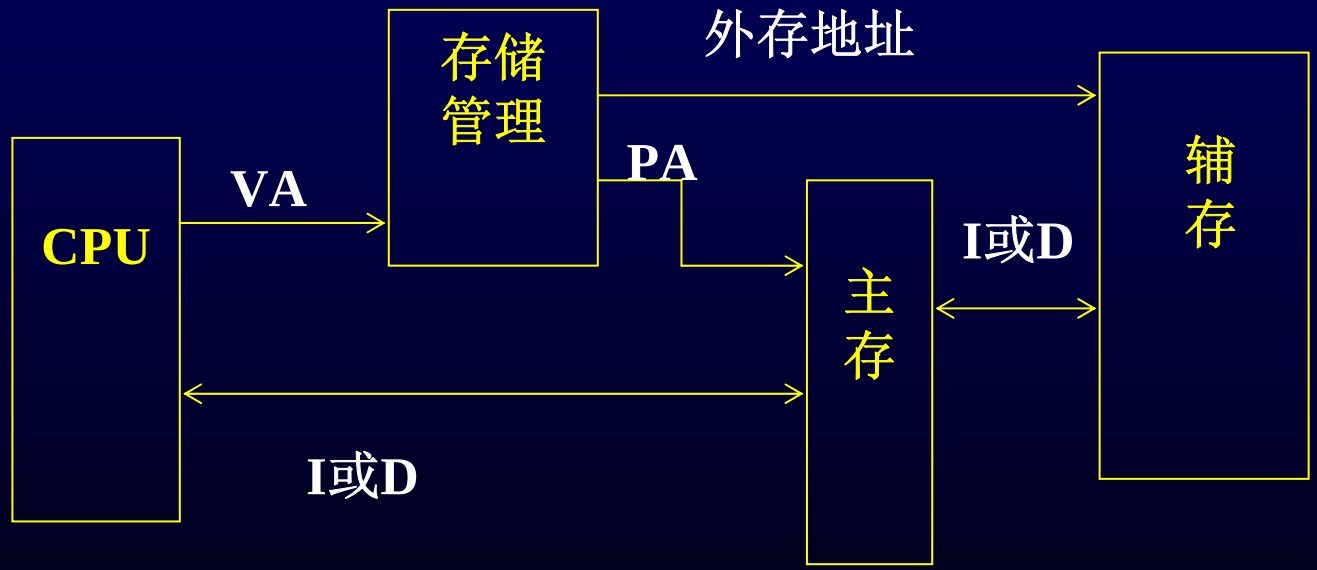
（4）采用LRU替换策略。指令体和数据体都有一个缓冲存储器，存放与主存交换的指令和数据。

四、虚拟存储器

1、 概述

虚拟存储器： 由主存储器和辅助存储器组成，通过必须的软件和硬件的支持，使得CPU可以访问的存储器具有近似于主存的速度和近似于辅存的容量。

虚空间： 程序所能利用的空间。
实地址： 主存物理空间的编址；
虚地址： 编程序时程序员所用的地址，在编译程序中由处理机生成。



虚拟存储器中CPU、主存、辅存间关系

*虚拟存储器与辅存的关系：*依据就是程序局部性原理。虚拟存储器通过存储管理部件，使辅存中的内容，动态的调入主存，以满足程序对大容量存储器的要求。

虚拟存储器与高速缓冲存储器区别

	Cache	虚拟存储器
功能	提高了主存储器的速度	扩大了主存储器的容量
实现技术	硬件	以软件为主
透明性	透明	不透明
地址转换	简单	复杂、速度慢

五、存储保护

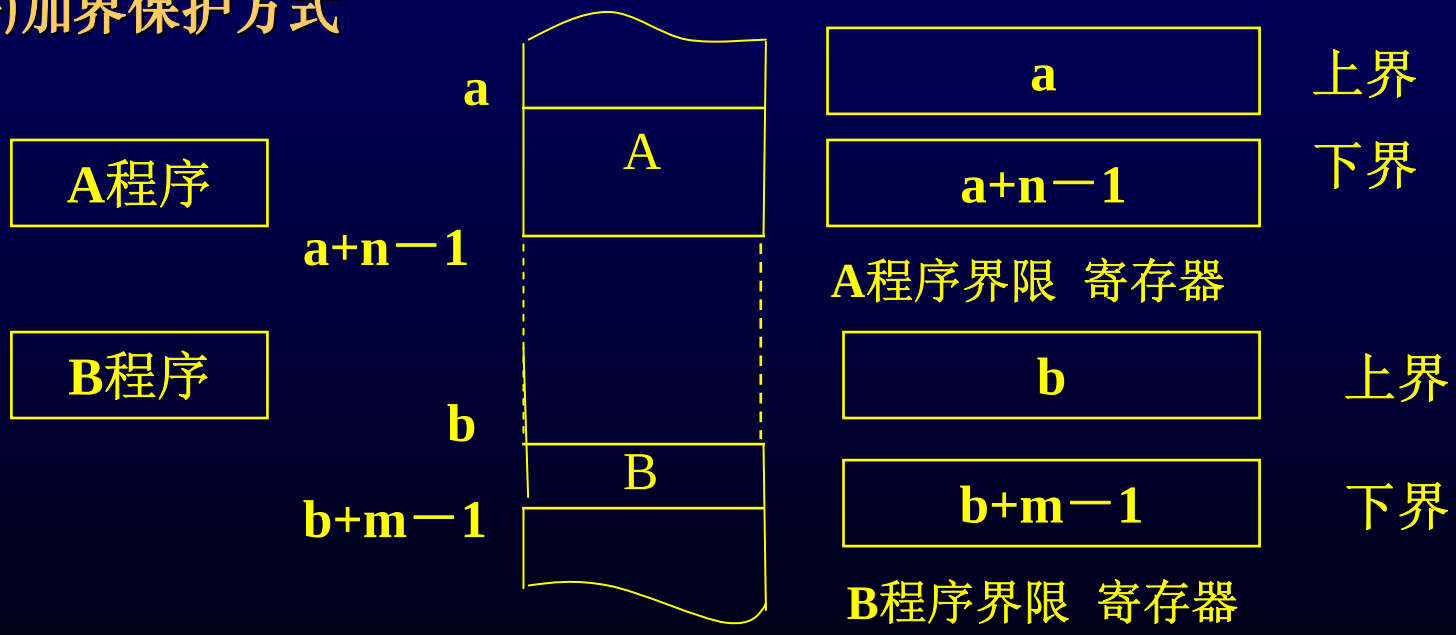
1、存储保护的方式

原因：

（1） 为了防止由于一个用户程序出错而破坏主存中其他用户的程序或系统软件；

（2）防止一个用户程序不合法地访问不是分配给它的主存区域，即使不会引起破坏。

(一)加界保护方式



程序的上下界：一个用户(或一段，或一进程)在采用段式管理的方式下，调入主存时是存放在一个连续的存储空间，此空间的开始与结束所对应的地址。

加界保护法：在CPU中设置了多个界限寄存器，由系统软件经特权指令指定，禁止越界。

当程序运行过程中，每当访问主存时，首先将访问地址与上下界寄存器进行比较，如果在此区域之内，则允许访问；如果不在此区域之内，即小于上界，大于下界，即说明出现了错误，称为越界错。这种保护方式是对存储区的保护、运用于段式管理。

(二) 键保护方式

锁：将主存的每一页都设置一个存储键，给予一个键号，此键号存放在快表的表目中，相当于一把“锁”。所有页的存储键在主存相应的快速寄存器内，每个用户的各实页的存储键都相同。

钥匙：访问键。由操作系统给定，存在程序状态字中。

过程：每次访问主存，首先进行键号比较，如果键号相等才允许访问。如同一把钥匙开一把锁。存放键与程序键键号的分配，由操作系统完成。

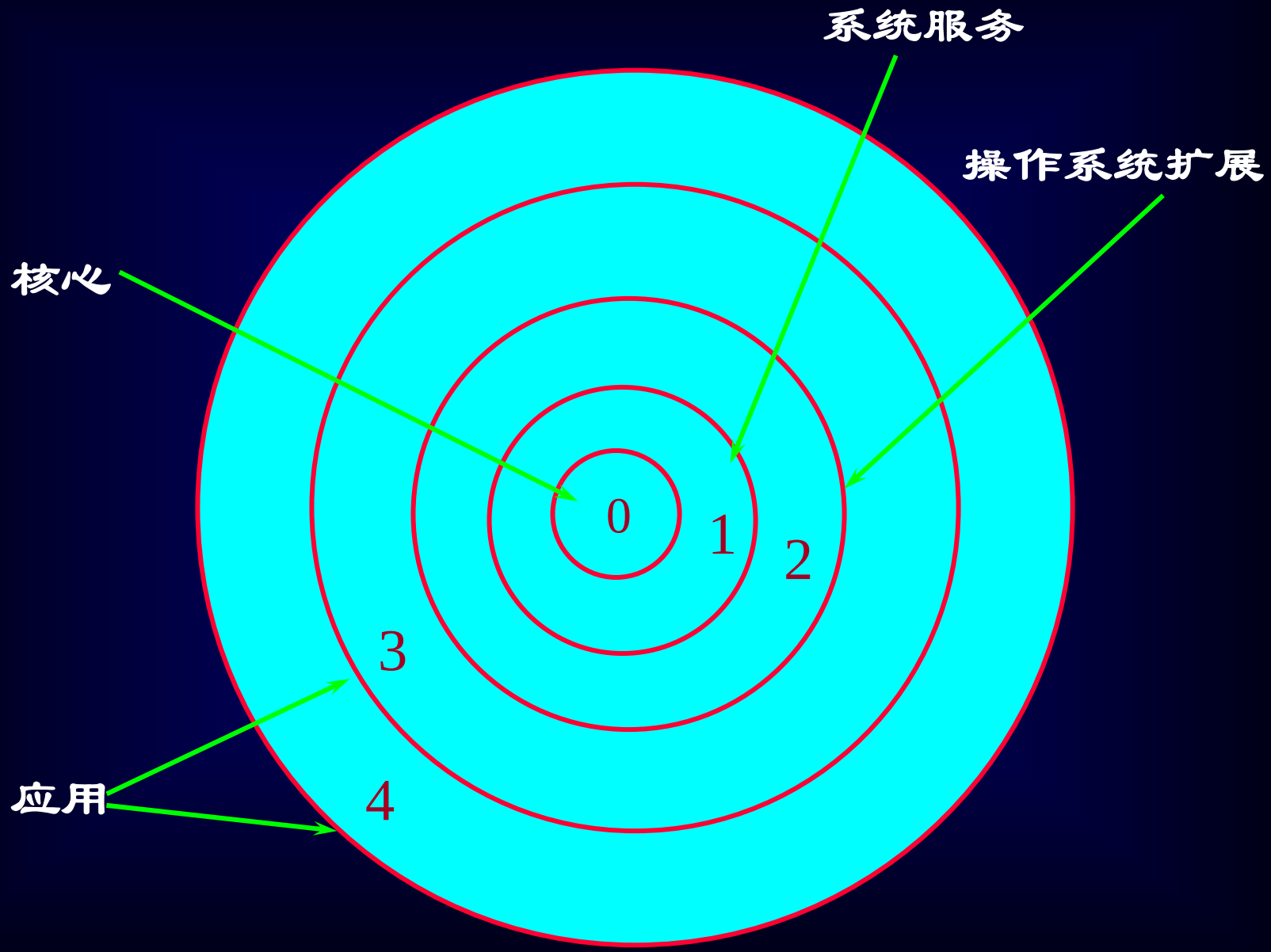
例如：IBM370的保护键有4位，能表示已调入主存的16个活跃的程序。其中“0000”访问键是操作系统的，对这个访问键不论是否和存储键相符都可访问，这是操作系统应能访问到主存整个区域所要求的。

(三) 环保护方式

作用：上面的保护是被保护的程序区域不被侵犯，但不是保护正在执行的程序本身不被侵犯。

环保护方式：将系统程序和各用户程序按其功能的性质和要求分为几个级别，分别授予不同的权限，如系统程序对安全的要求比较高，授权级别就比较高，用户程序的级别就可以低些，各级的关系如图所示。

各级之间如同心环的关系由内层向外层级别逐渐降低，各层的环号由操作系统给定，每个程序都具有一个环号，此环号说明该程序只能访问它的同级或低级的程序，不可访问级别比它高的程序，一般情况下，第0层为系统程序的核心，安全级别最高，第1层为系统服务程序，第2层为操作系统扩展，第3层以下为应用程序。


系统服务
操作系统扩展
核心
0
1
2
3
应用
4

访问方式的保护：

对内存的信息可以有三种访问操作，即读、写、执行访问方式

(1) 可读，可写，可以执行
(2) 可读，可执行，不可写
(3) 只可读，不可写，不可执行
(4) 只可读，可写，不可执行，例如数据
(5) 只能执行不可读写，例如专用程序

与上述方法结合使用，在寄存器中增设一位访问位即可。

2、虚拟存储器与存储保护举例

（一）Pentium处理机：

外部数据总线宽度为64位，内部寄存器是32位，外部地址总线为36位。

目前普遍的作法是使用32位的地址空间，因此认为Pentium使用的是32位地址总线，最大物理地址空间是2^{32}=4GB。

（二）Pentium处理机的工作模式

1. 实地址模式：MS-DOS操作系统、Windows3.X和它们的16位程序采用实模式。
2. 保护模式：段式、页式、段页式
3. 虚拟86模式：80386开始具备，一直延续至Pentium
4. 系统管理模式SMM（System Management Mode）

(三) Pentium的存储保护

Pentium的存储保护包括特权级保护和存储区域保护。

特权保护功能的主要目的是不允许应用程序错误的修改操作系统的数据，而又允许应用程序调用OS提供的例行服务子程序，Pentium采用4级特权PL的环保护 。

PL=0级具有最高特权，PL=3级权限最低，保护规则是:

1. 特权级PL的存储数据只允许PL级和PL以上的级的代码进行访问;

2. 特权级PL的代码，或过程只允许PL级或PL以下的各级的任务调用。

作业：

1、假设在一个采用组相联映象方式的Cache中，主存由B0～B7共8块组成，Cache有2组，每组2块，每块的大小为16个字节，采用LFU块替换算法。在一个程序执行过程中依次访问这个Cache的块地址流如下:

6，2，4，1，4，6，3，0，4，5，7，3

(1) 写出主存地址的格式，并标出各字段的长度。

(2) 写出Cache地址的格式，并标出各字段的长度。

(3) 画出主存与Cache之间各个块的映象对应关系。

(4) 如果Cache的各个块号为C0、C1、C2和C3，列出程序执行过程中Cache的块地址流情况。

(5) 如果采用FIFO替换算法，计算Cache的块命中率。

(6)采用LFU替换算法，计算Cache的块命中率。

2、假设机器的时钟周期为10ns, Cache失效时的访存时间为20个时钟周期，Cache的访问时间为一个时钟周期。

（1）设失效率为0.05，忽略写操作时的其它延迟，求机器的平均访存时间。

（2）假设通过增加Cache容量一倍而使失效率降低到0.03，但使得Cache命中时的访问时间增加到了1.2时钟周期（即12ns），指出这样的改动设计是否合适？

(3) 如果时钟周期取决于Cache的访问时间(也就是用延长时钟周期的方法)，上述改动设计是否合适？

本章要点

- 存储系统的概念
- 高速缓冲存储器
- 地址映象与变换
- LRU替换算法
- 存储保护