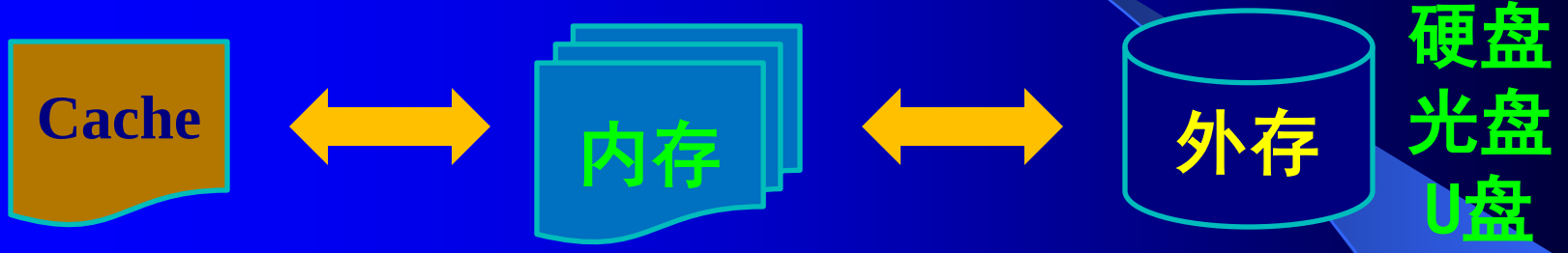
4.6 三级存储体系

计算机中的存储系统，分为三个层次：

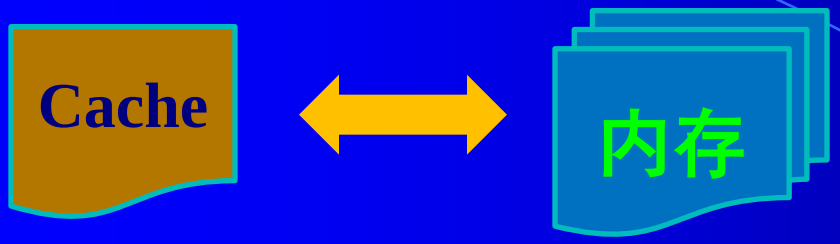


外　存：确保计算机具有足够大的存储容量；
确保数据能脱机保存；

内　存：存储运行期指令/数据，确保CPU能快速读取；

CACHE：强化CPU快速读取指令和数据的速度；

※Cache和内存的关联



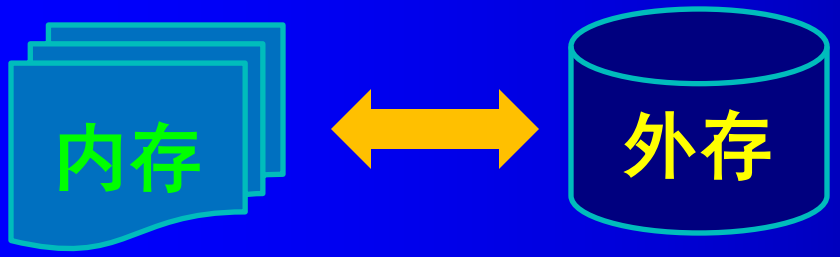
内存中的数据，调入Cache

Cache中的数据，写回内存

以数据块为单位整体操作：

直接映射、全相联映射、组相联映射

※内存和外存的关联



外存中的数据，调入内存

内存中的数据，写回外存

虚拟存储技术： 页式、段式、段页式

4.6.1 Cache与主存映射

1. 设置Cache的原因

※为解决CPU和主存速度不匹配而采用的一项技术，使访问主存的平均速度接近于访问Cache的速度。

※由硬件系统实现，对用户透明。

※已在CPU内集成，两级以上的Cache系统。

2. Cache的前提条件

※指令的执行具有局部性特征：

CPU从主存中取指令、数据，在一定时间内地址范围常局限于主存的某个小区域。

因此可以将正在使用的部分（热点区指令和数据），提前预取并存储到一个高速的、小容量的Cache中。

CPU访存 ⇒ CPU访问Cache

[技术效果]

能使CPU读写指令、数据的速度大大提高。

3. 主存与Cache的地址映射

[说明]

主存↔Cache之间是以固定大小的数据块为单位进行整体调度（交换）；

基于下列条件，分析3种主存↔Cache映射：

- ✓ 存储器均按字节编址：1B/每个地址；
- ✓ 数据块大小：512B；
- ✓ Cache容量：8KB →分成16块（$2^{13}/2^{9}$）
- ✓ 主存容量大小：1MB →分成2048块（$2^{20}/2^{9}$）

（1）直接映射

Cache：只分块、不分组

主存：既分块、也分组（每组的块数 = Cache块数）

[映射规则]主存的每一个数据块，只能映射到与其组内序号相同的Cache数据块位置。

如果：K为Cache的块序号，J为主存块的序号，C为Cache块号的位数。

则 $K=J \bmod 2^c = J \bmod 2^4$

直接映射的规则，如图4-47示

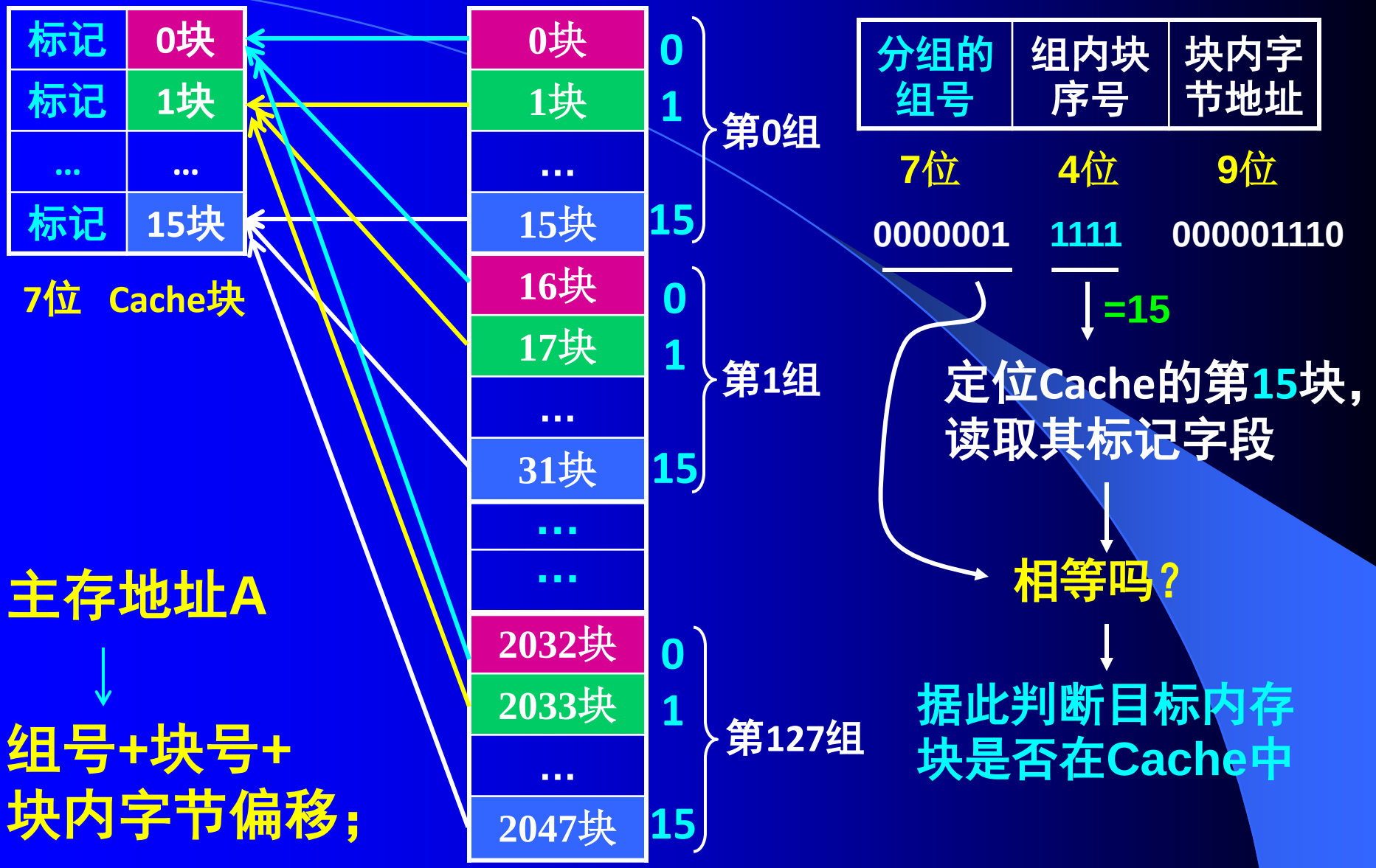


图4-47 直接映射 C=4、N_a=20，每块512字节

（2）全相联映射

Cache：只分块、不分组

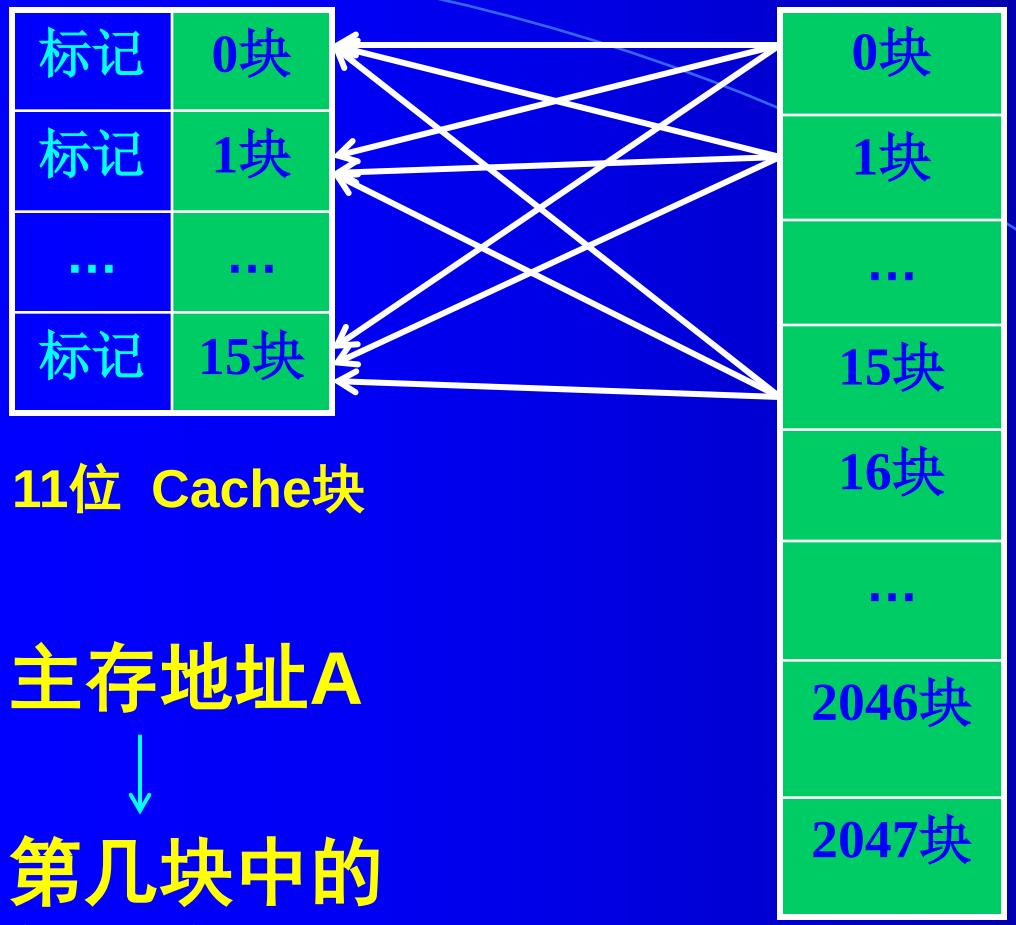
主　存：只分块、不分组

[映射规则]主存任何一个块都可以映射到Cache的任何一个数据块位置上，如图4-48示。

存在的缺点：

✓Cache标记太长，判断时间太长。

✓硬件复杂、成本高、实现相对困难。



主存块号	块内地址
11位	9位
00000011111	000001110

=31

主存地址A

↓

第几块中的
第几个地址？

将主存块的块号与Cache块的标记字段比较，判断主存块是否已映射到缓存中

图4-48 全相联映射 N_a=20， 每块512字节

（3）组相联映射

Cache：既分块、也分组

主 存：既分块、也分组（组内块数 = Cache组数）

[映射规则]

主存数据块，映射到与自己组内块序号相同的Cache分组，可占据Cache分组中的任意数据块位置。

定位Cache的分组：直接映射；

定位Cache数据块：全相联映射；

→直接映射和全相联映射的折衷

速度快、硬件简单、成本低、易实现（图4-49示）

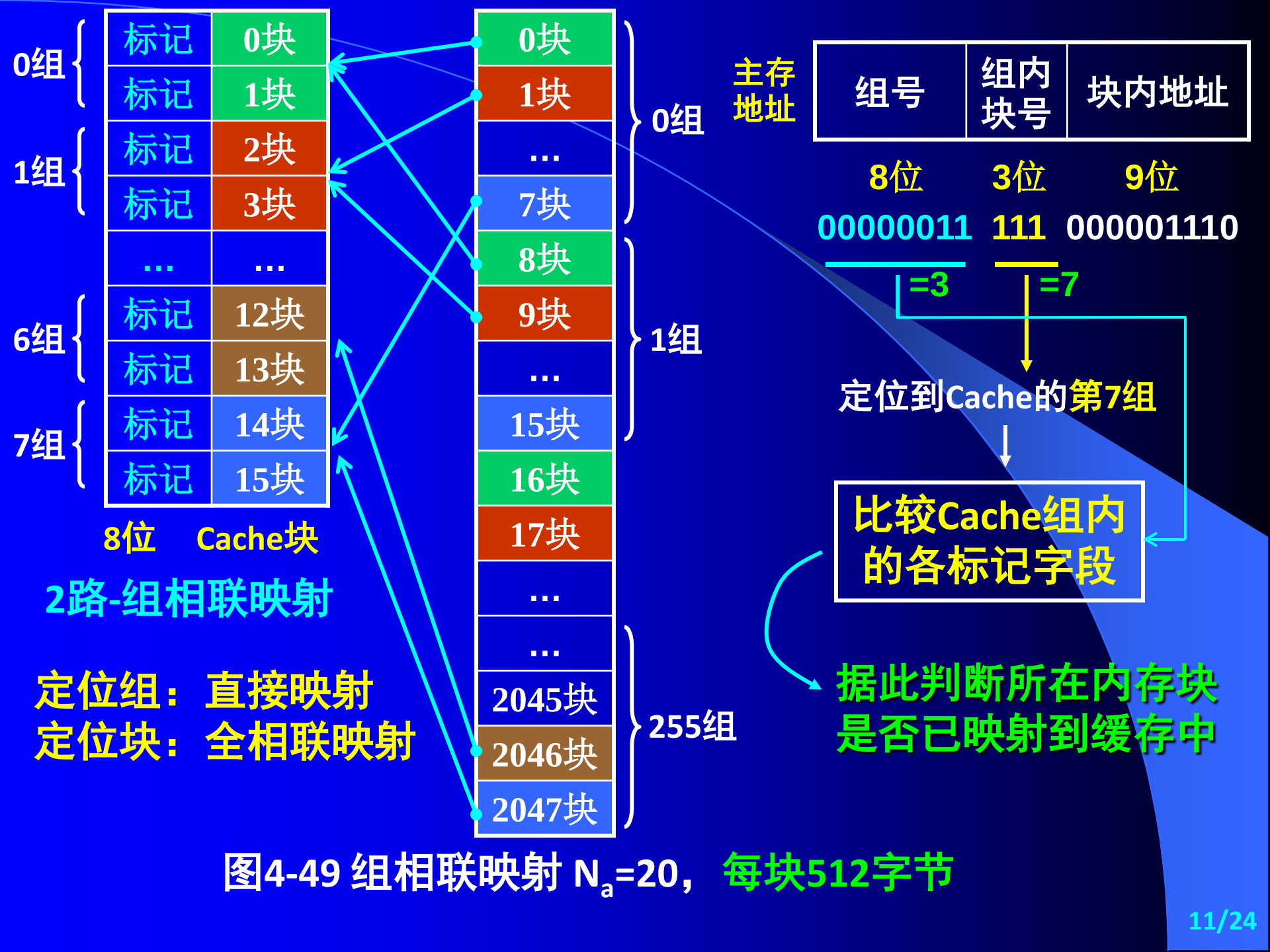


图4-49 组相联映射 N_a=20，每块512字节

[举例]某计算机的Cache共有16块，采用2路-组相联映射方式(即每组包括2块)。存储器按字节编址，每个主存块大小为32字节，那么129号主存单元所在的主存块应装入到的Cache组号是(　)：

A. 0　B. 2　**C. 4**　D.6

[解题分析]

Cache如何分组、分块？

Cache分8组，每组2块，每块32B

主存如何分组、分块？

主存分若干组，每组又分成8块

129=10000001 ⟶ 0…010000001（组内块序号100）

4. 常用的替换算法

（1）最不经常使用(LFU, Least-Frequently Used)

将一段时间内被访问次数最少的那块从Cache中置换出去。

（2）近期最久未使用(LRU, Least-Recently Used)

将近期内最久未被访问过的Cache块置换出去。

（3）随机替换

随机确定将哪块从Cache中替换出去。

5. Cache的读/写操作

※Cache的写操作

当CPU发出写请求时，如果Cache命中，可以有两种处理方案：

①Cache单元和主存单元同时写，使Cache和主存保持一致，称为通写(write-through)。

②只修改Cache单元，并用标志将该块加以注明，直到该块从Cache中替换出来时才一次性写入主存，称为回写(write-back)。

※Cache的读操作

① 旁路式读 (Look-Aside)

CPU向Cache和主存同时发读命令和地址。

Cache命中，则Cache回送数据并中断读主存命令；

Cache未命中，则直接访问主存读取数据。

② 通过式读(Look-Through)

CPU首先向Cache发读命令和地址。

Cache命中，则从Cache中读出数据；

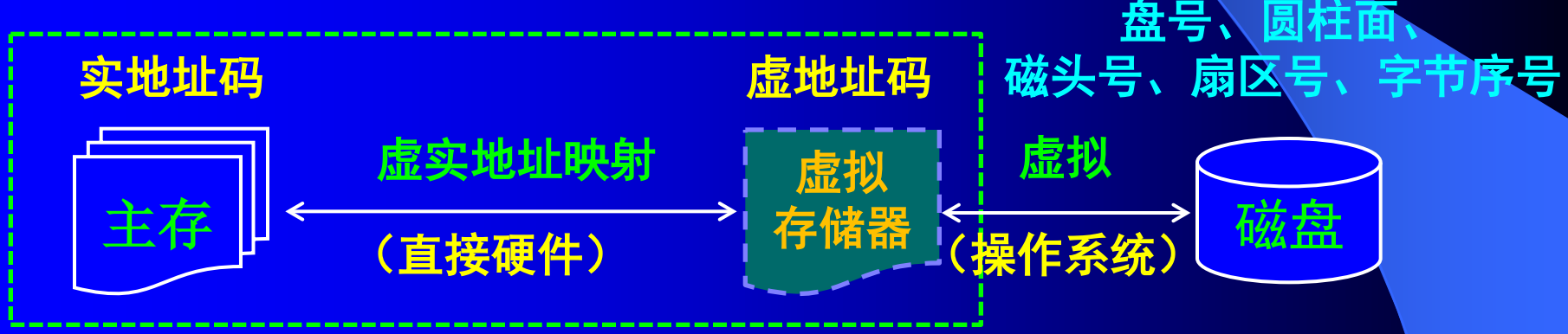
Cache未命中，再将读命令和地址传给主存并读主存。

[讨论] 命中率、平均访问时间和访问效率?

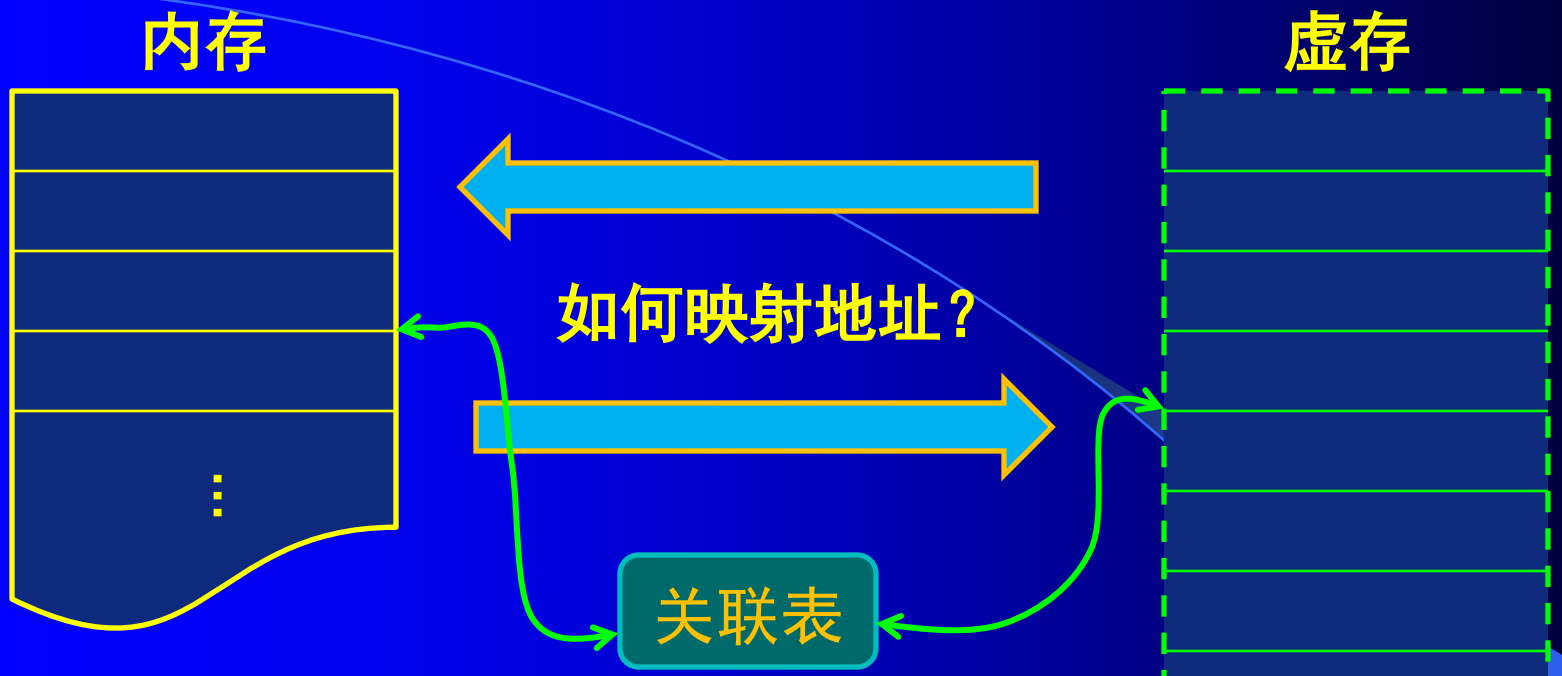
4.6.2 内存与外存的映射

※虚拟存储的基本概念

在内存和外存之间，由操作系统存储管理模块及相关硬件（存储器管理部件）实现的一种存储映射技术。



逻辑上能提供比物理存储器更大的虚拟存储空间，相关地址称为虚拟地址或逻辑地址。



基本方法：

在两者之间，增加关联表。

（1）内存、虚存都分成页： 页表

（2）内存、虚存都分成段： 段表

（3）内存、虚存都分段、每段再分页： 段页表

1、页式虚拟存储管理

主存和外存统一分页后进行管理。

※页表

记录虚地址页号与实地址页号的对应关系，即虚页面调入主存时被安排在主存中的位置（实页号）

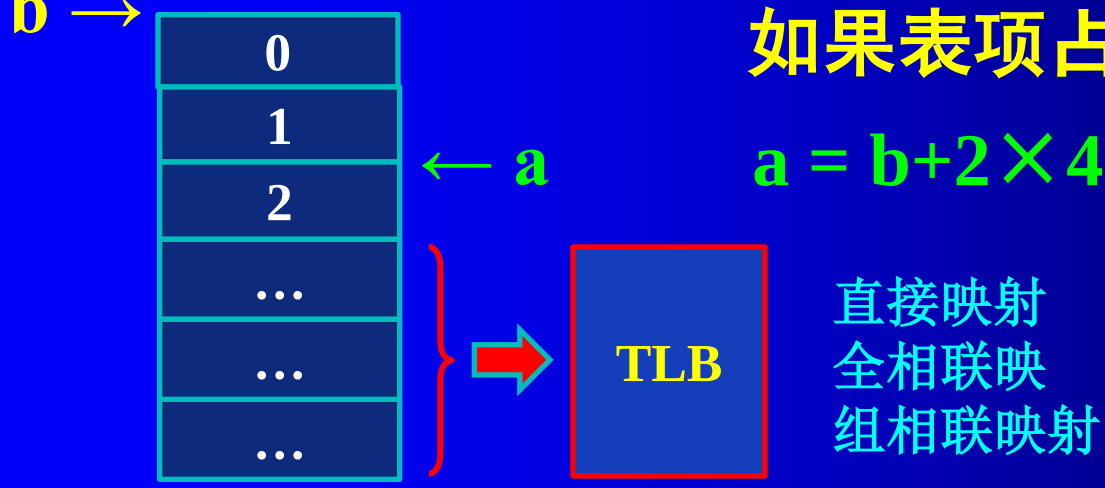
页表中的每一行，称为页表项；

虚页号	有效位	实页号	…	
0000	1	0101	…	←页表项
0001	1	1011	…	←页表项
0010	0	0010	…	←页表项
…	…	…	…	

※页表基址寄存器

记录页表在主存中的起始地址；

页表项的地址a=页表基址b+页号n×表项的字节数w

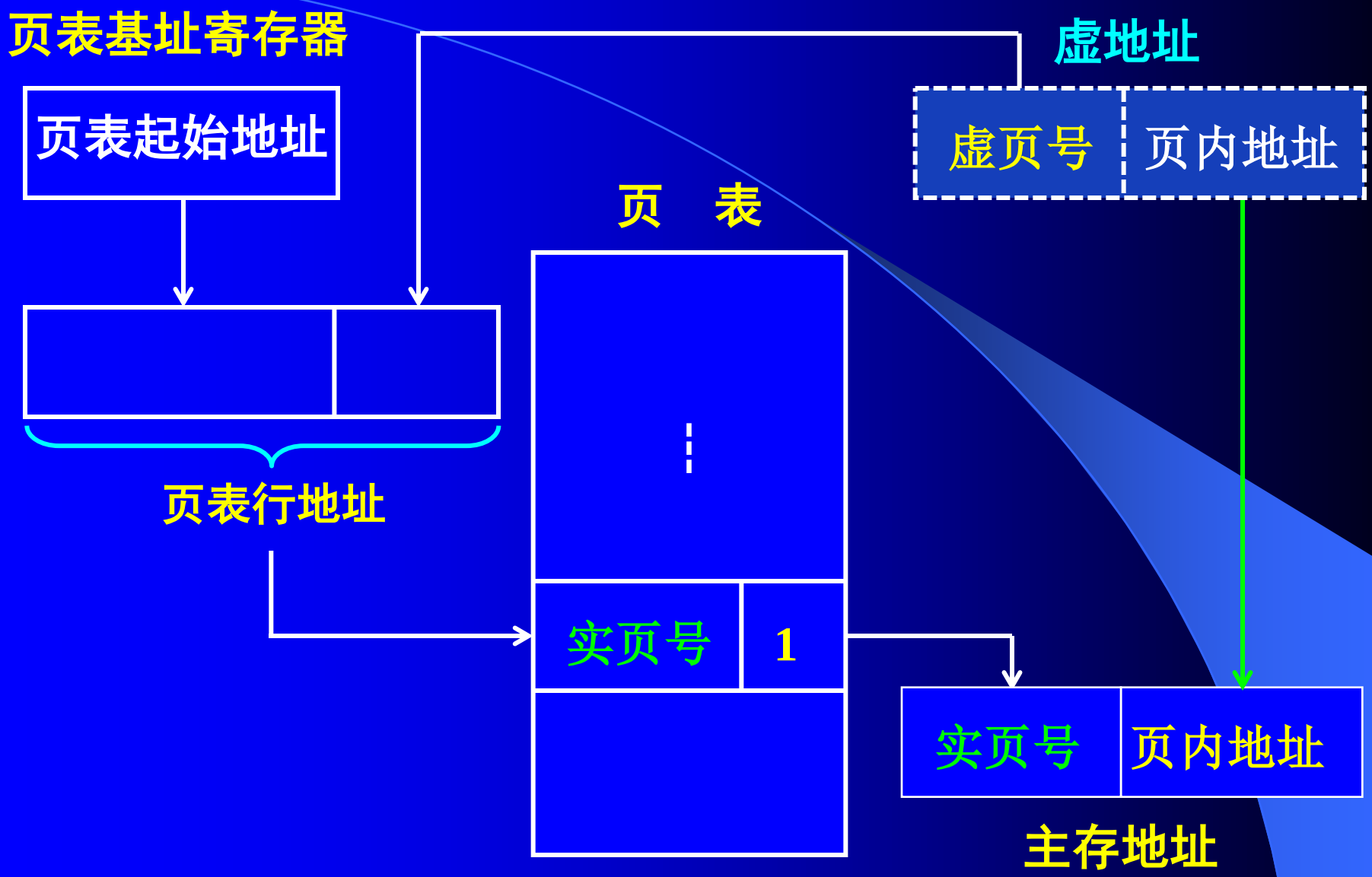


如果表项占4B

a = b+2×4

※快表(TLB)

把活跃的页表项用高速存储器单独存储，访问速度更快，它是页表的子集。



页式虚拟存储器地址转换

2、段式虚拟存储器管理

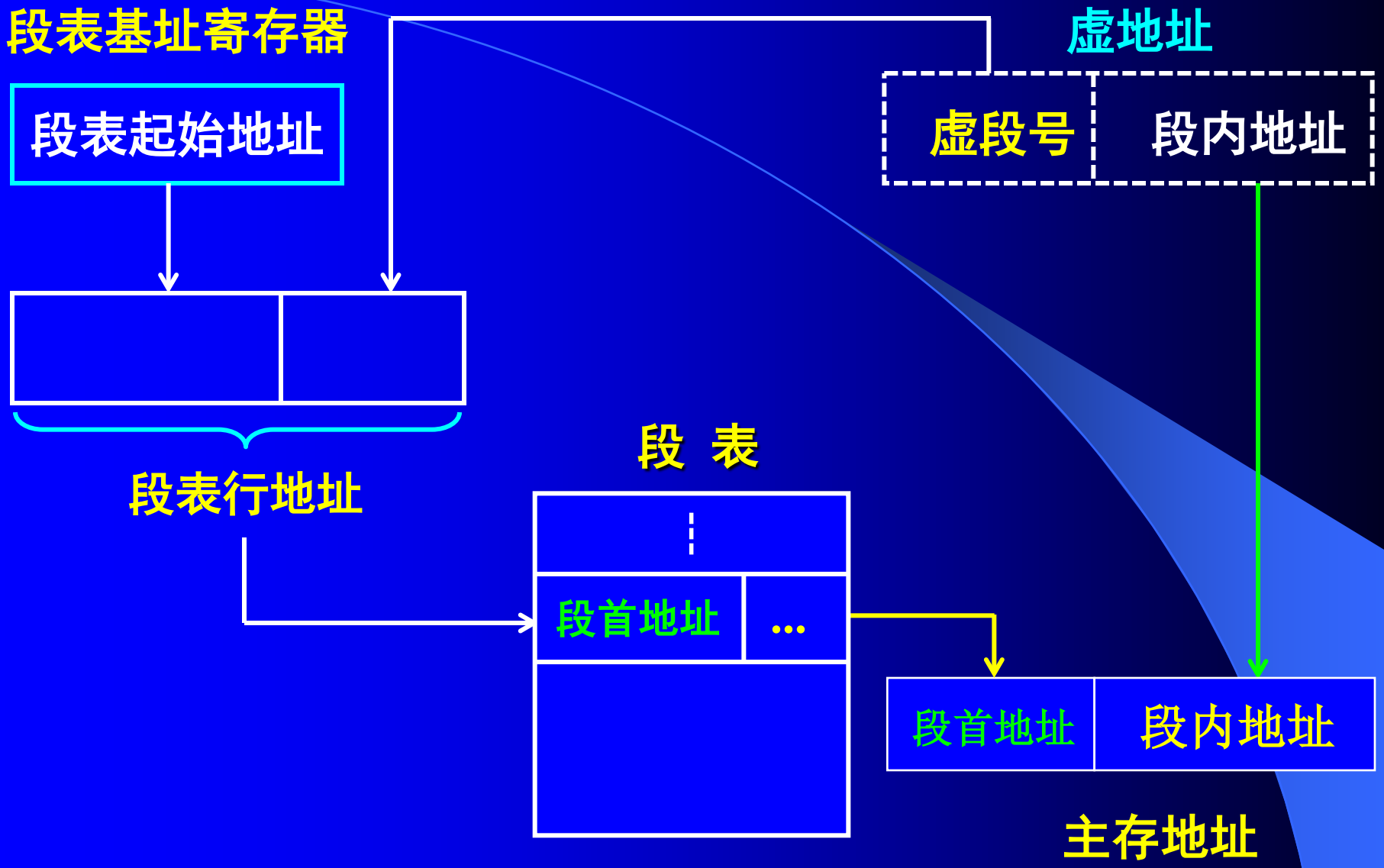
虚存中的程序分段（按照代码段、数据段和共享段等）进行管理。

为了将虚拟地址变换成主存实地址，操作系统创建1个段表。

每段在段表中都占有一登记项，内容包括：段号、段起点、段长、装入位等。

根据虚段号查段表 →在内存中的首地址

虚实地址变换，如后图示。



段式虚拟存储器地址转换

3、段页式虚拟存储管理

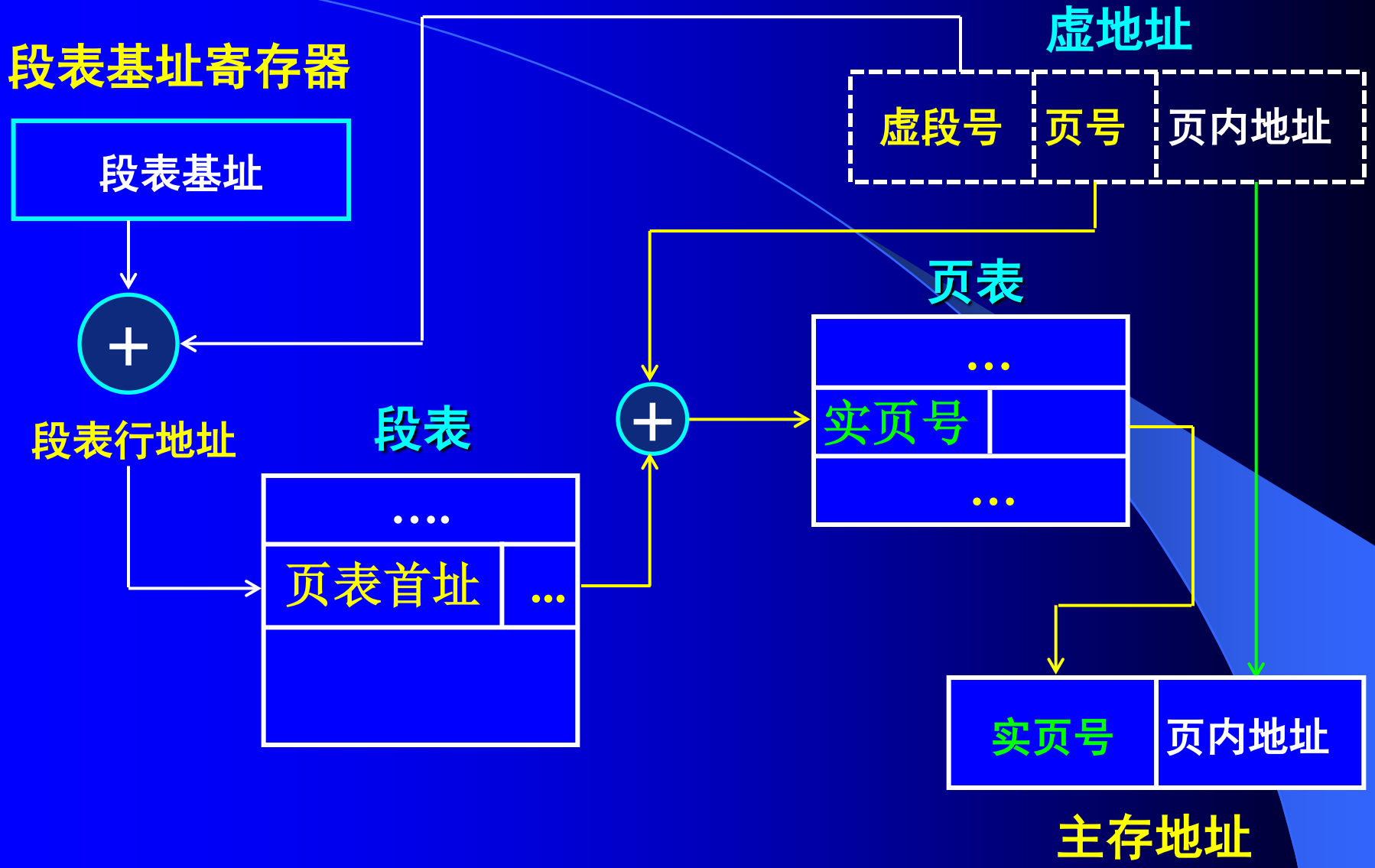
每个程序按逻辑模块分段，每段再分页，页面大小与内存页面相同；

由操作系统创建两表：段表、页表；

- 虚地址格式：段号+页号+页内地址
- 实地址格式：页号+页内地址

先查段表→页表首地址
再查页表→内存实页号
} 内存地址

要经过两次查表才能完成地址转换，速度较慢；



段页式虚拟存储器地址转换过程