第四章　存 储 器 管 理

BACK

4.5 基本分页存储管理方式

连续分配方式会形成“碎片”
“紧凑”须付出很大开销 } ⇨ 产生了离散分配方式

4.5.1 页面和页表

1. 页面

1）页面和物理块

- 将用户程序的地址空间（逻辑地址）分成若干大小相等的片，称为页面或页(page)。页号从0开始，如第0页、第1页等。
- 内存空间分成与页大小相同的若干存储块，称为块或页框(frame)。也从0开始编号，如0#块、1#块等。

最后一个页不满，称为页内碎片。

为进程分配内存时，以块为单位，将进程中的若干页分别装入到多个可以不相邻接的物理块中。

4.5 基本分页存储管理方式

连续分配方式会形成“碎片”
“紧凑”须付出很大开销 } ⇨ 产生了离散分配方式

4.5.1 页面和页表

1. 页面

1）页面和物理块

- 将用户程序的地址空间（逻辑地址）分成若干大小相等的片，称为页面或页(page)。页号从0开始，如第0页、第1页等。
- 内存空间分成与页大小相同的若干存储块，称为块或页框(frame)。也从0开始编号，如0#块、1#块等。

最后一个页不满，称为页内碎片。

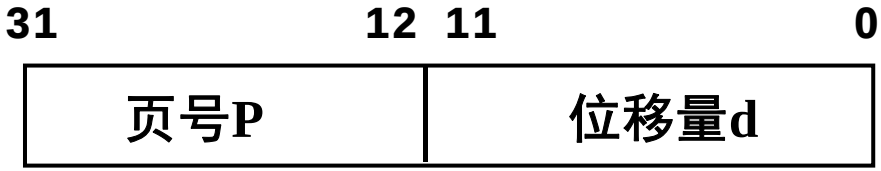
2）页面的大小

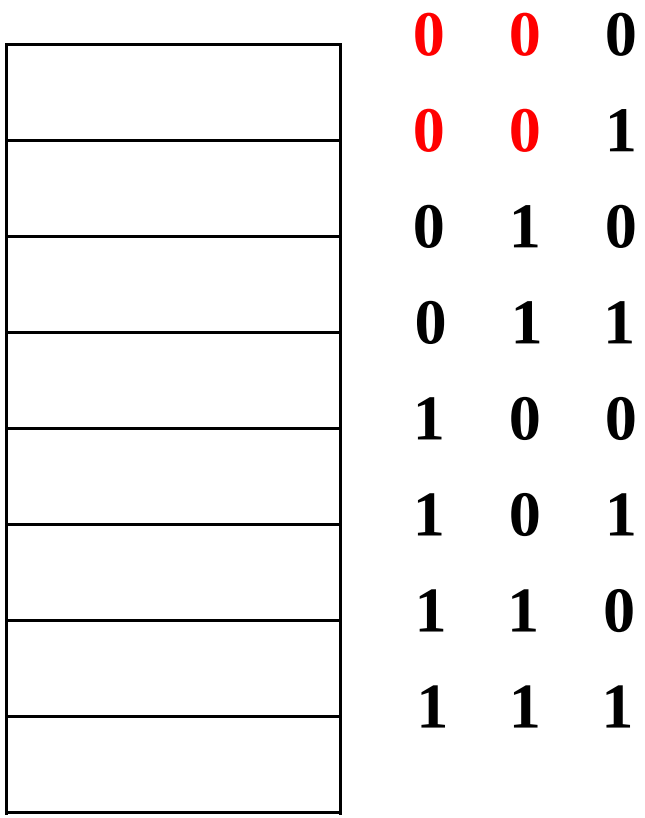
应是2的幂。通常为512B～8KB——不宜过小，也不宜过大

商用计算机的页面大小在512B～64KB之间，以往的典型值为1KB，而现在更常见的页面大小为4KB和8KB。——《现代操作系统》。

2．地址结构

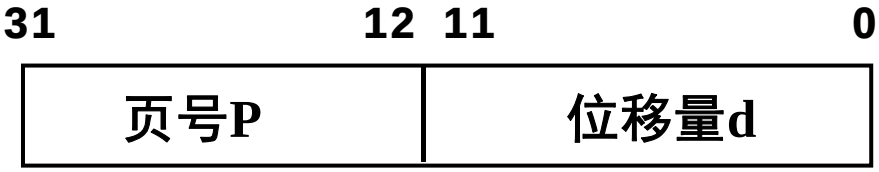
分页的逻辑地址结构如下：





2．地址结构

分页的逻辑地址结构如下：



位移量d也称页内地址。图中的地址长度为32位，每页大小为4KB，地址空间最多2^{20}（1M）个页。

对于特定的机器，其地址结构是一定的。

若逻辑地址为A, 页面大小为L, 则页号P和页内地址d可按下式求得：

$$P=int(A/L)$$

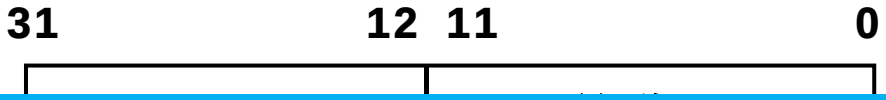
$$d=A \bmod L$$

举例说明

页面大小为4KB，逻辑地址A为4170及5F86H，求它们的页号和页内偏移。

2．地址结构

分页的逻辑地址结构如下：



【A=4170】

页号：1；页内偏移：74

地址：0000 0000 0000 0000 0001　0000 0100 1010

【A=5F86】

页号：5；页内偏移：3974

地址：0000 0000 0000 0000 0101　1111 1000 0110

举例说明

页面大小为4KB，逻辑地址A为4170及5F86H，求它们的页号和页内偏移。

3. 页表

系统为每个进程建立一张页表，记录了相应页在内存中对应的物理块号，实现从页号到物理块号的地址映射。

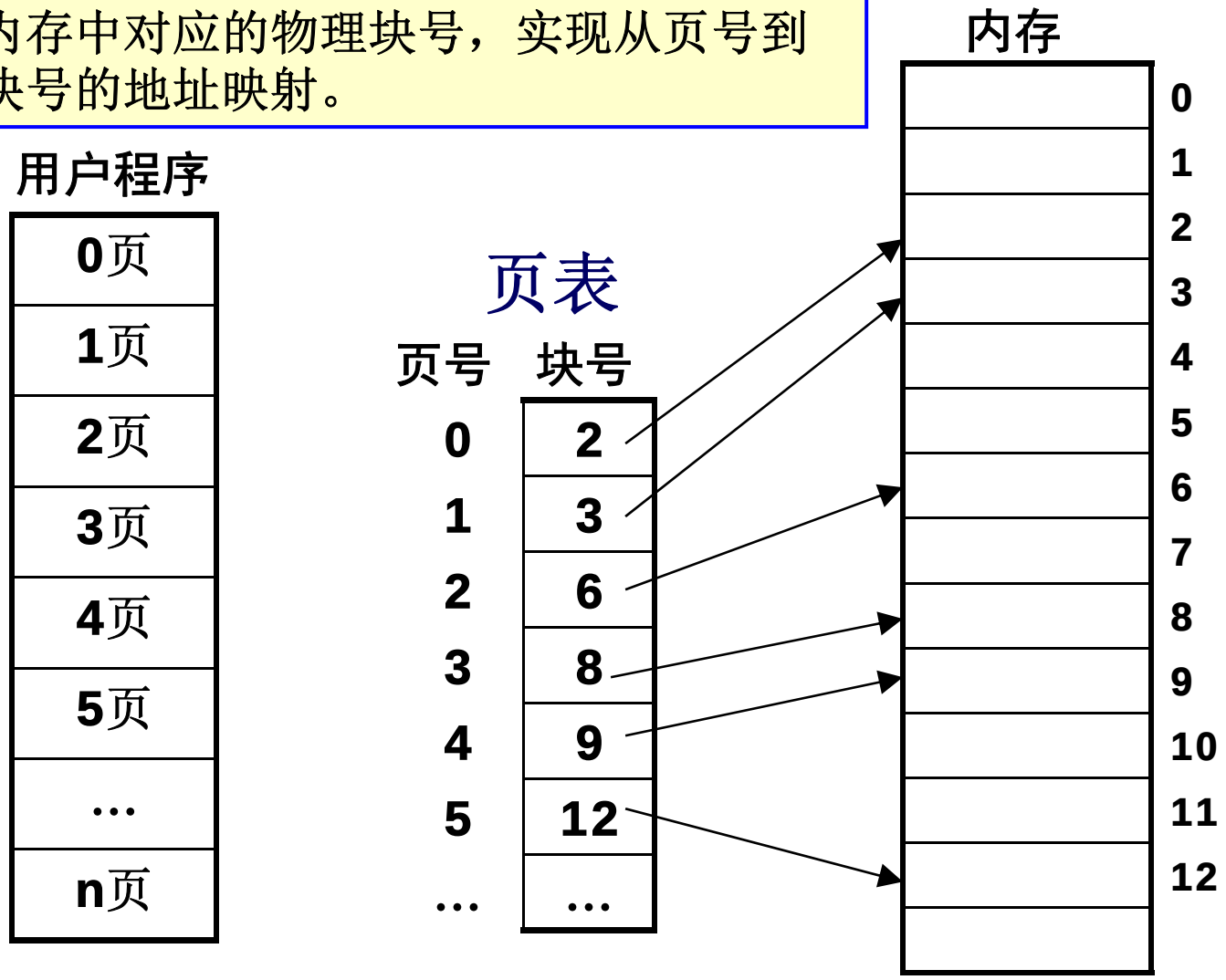


图4-14 页表的作用

3. 页表

系统为每个进程建立一张页表，记录了相应页在内存中对应的物理块号，实现从页号到物理块号的地址映射。

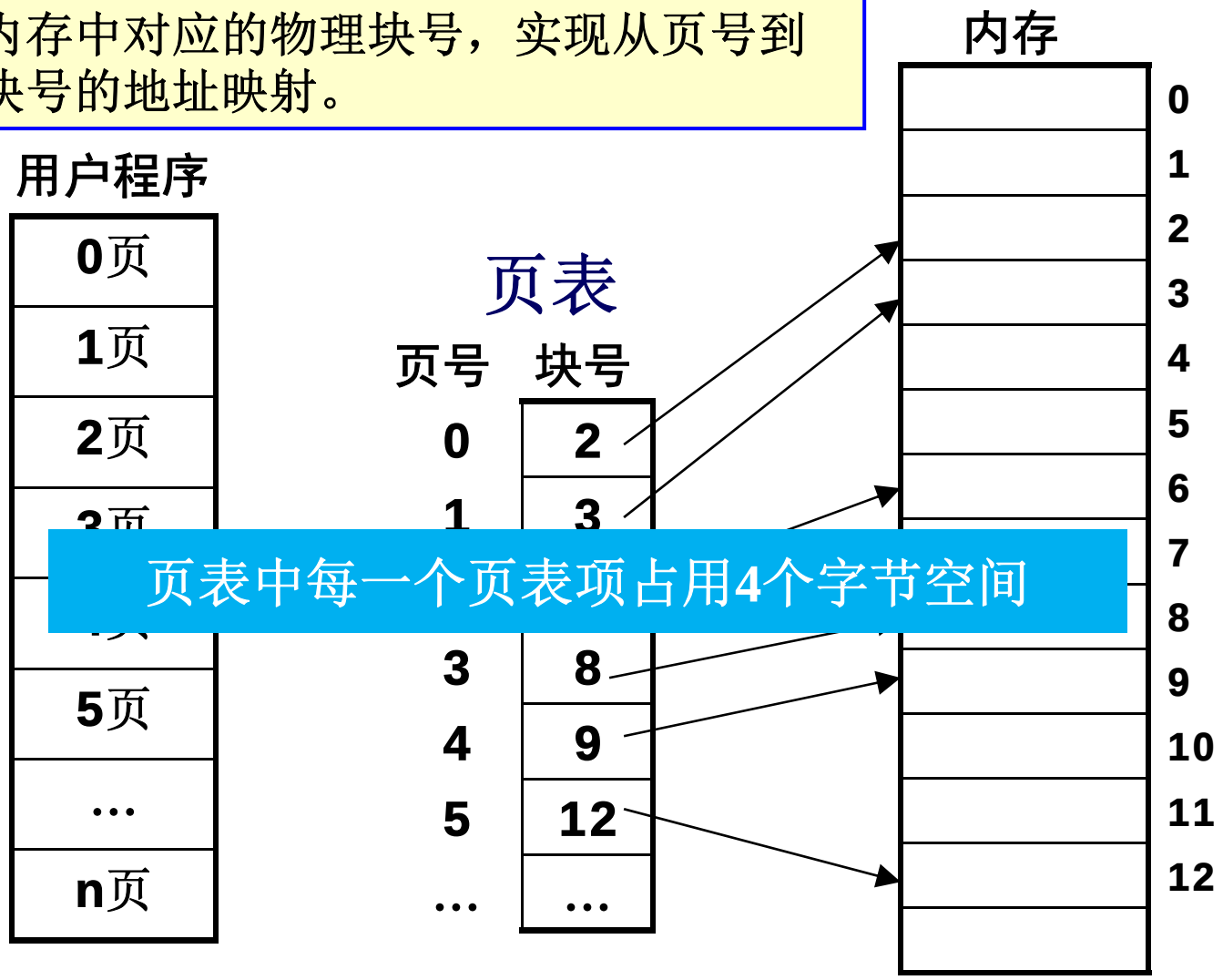


图4-14 页表的作用

4.5.2 地址变换机构

用户地址空间中的逻辑地址转换为内存空间中的物理地址

——地址转换机构。

页内地址和物理地址是一一对应的，因为页大小和物理块大小一致，页面大小为1KB的页内地址是0~1023，对应的物理块内地址也是0~1023，无需转换。

因此，地址变换机构的任务实际上只是将逻辑地址中的页号转换为内存中的物理块号。

地址变换任务是借助于**页表**来完成的。

1. 基本的地址转换机构

逻辑地址和物理地址的转换在进程运行期间执行的频率非常高，每条指令的地址都需要转换，因此需要采用硬件来实现。

页表功能由一组专门的寄存器来实现。一个页表项用一个寄存器。

但寄存器成本很高，现代计算机的页表又可能很大，使页表项的总数可达几千甚至几十万个，这么多页表项不可能都用寄存器来实现。

1. 基本的地址转换机构

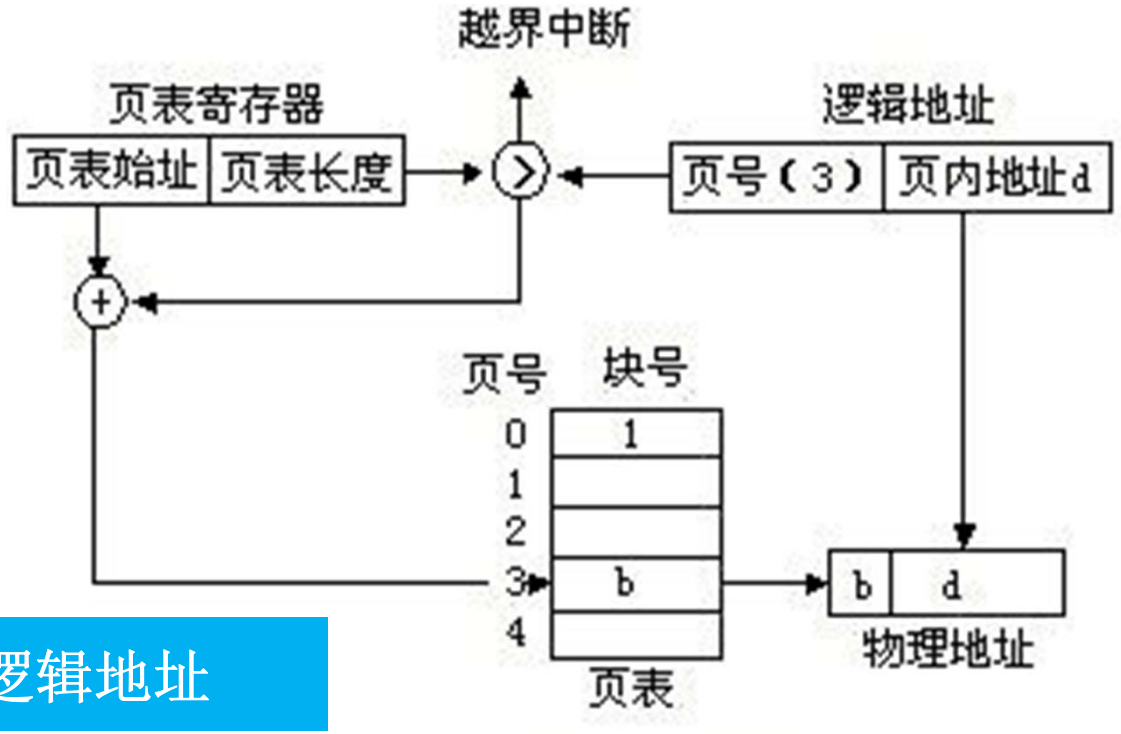
页表大多驻留在内存中，设置一个**页表寄存器PTR**存放页表在内存中的始址。地址转换的原理如图**4-15**所示。

进程未执行时，页表的始址和页表长度存放在本进程的**PCB**中；
当调度程序调度到该进程时，才将两个数据装入页表寄存器中。

单处理机系统中，只需要一个页表寄存器

1. 基本的地址转换机构

页表大多驻留在内存中，设置一个**页表寄存器PTR**存放页表在内存中的始址。地址转换的原理如图**4-15**所示。



举例说明

有效地址=逻辑地址

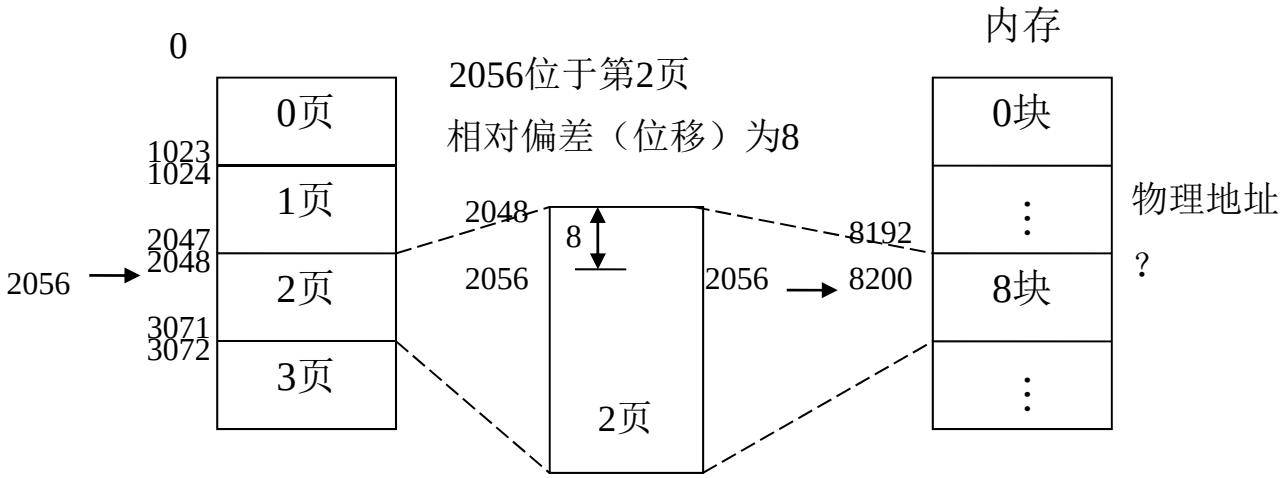
图**4-15** 分页系统的地址变换机构

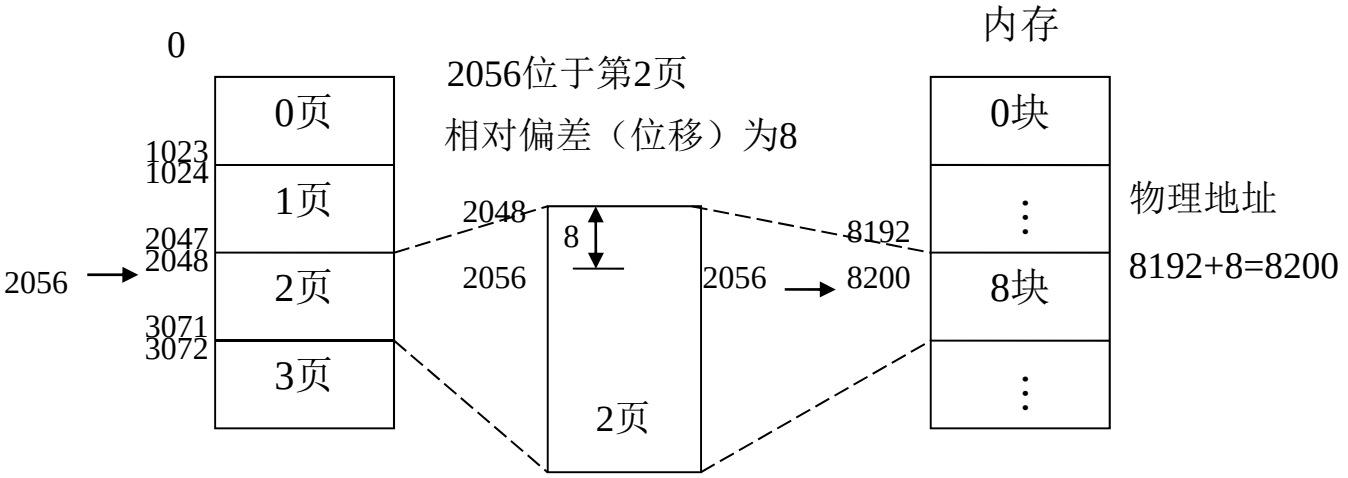
当进程要访问某个逻辑地址中的数据时，分页地址变换机构会自动地将有效地址(相对地址)分为页号和页内地址两部分，再以页号为索引去检索页表。查找操作由硬件执行。在执行检索之前，先将页号与页表长度进行比较，如果页号大于或等于页表长度，则表示本次所访问的地址已超越进程的地址空间。于是，这一错误将被系统发现并产生一地址越界中断。若未出现越界错误，则将页表始址与页号和页表项长度的乘积相加，便得到该表项在页表中的位置，于是可从中得到该页的物理块号，将之装入物理地址寄存器中。与此同时，再将有效地址寄存器中的页内地址送入物理地址寄存器的块内地址字段中。这样便完成了从逻辑地址到物理地址的变换。

页表项长度：4字节

2 工作原理

考察逻辑地址2056，假定一页大小为1KB，其中0号页放到了3号块中，1号页放到了5号块，2号页放到了8号块，求此逻辑地址对应的物理地址。（假设地址是16位）





1. 先计算逻辑地址的页号和页内位移量：

   页号 = int(逻辑地址/页大小) = int(2056/1024) = 2

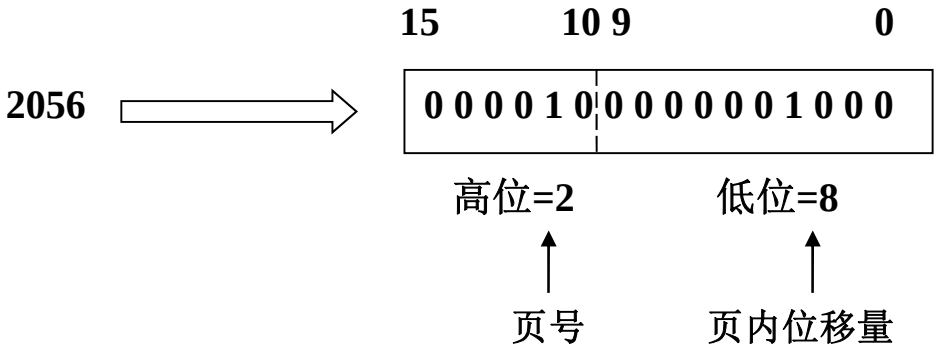
   页内位移量 = 逻辑地址 MOD 页大小 = 8

2. 通过页号2，查询页表可得到物理块号为8

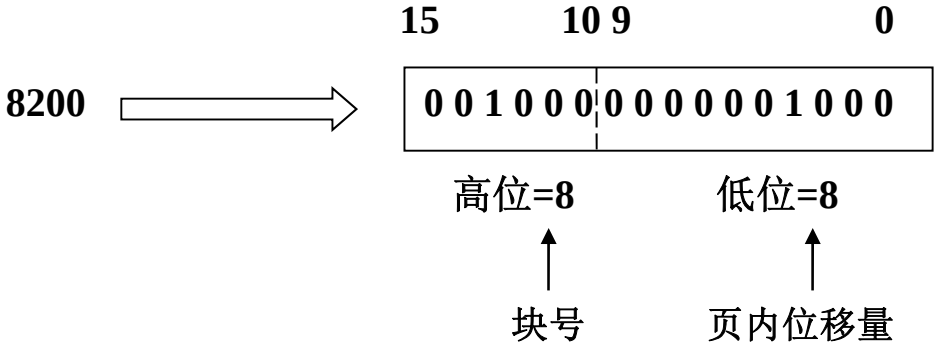
3. 物理地址 = 块号*页大小+页内位移量 = 8*1K + 8 = 8200

* 转换过程分析

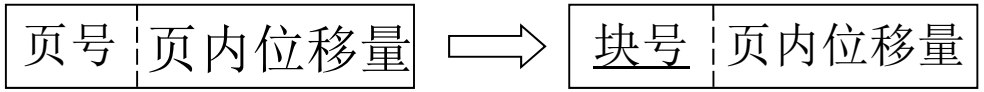
逻辑地址机内表示
（以16位为例）



物理地址机内表示
（以16位为例）



总结：逻辑地址转换成物理地址时，无需计算，只需用块号代替高位的页号，就可立即得到对应的物理地址



2．具有快表的地址转换机构

由分页系统的地址转换可知，每存取一个数据，要访问2次内存，计算机处理速度降低近1/2。为提高地址转换速度，在MMU中增设一个具有并行查寻能力的特殊高速缓冲寄存器，称为**联想寄存器**，或称**快表**。

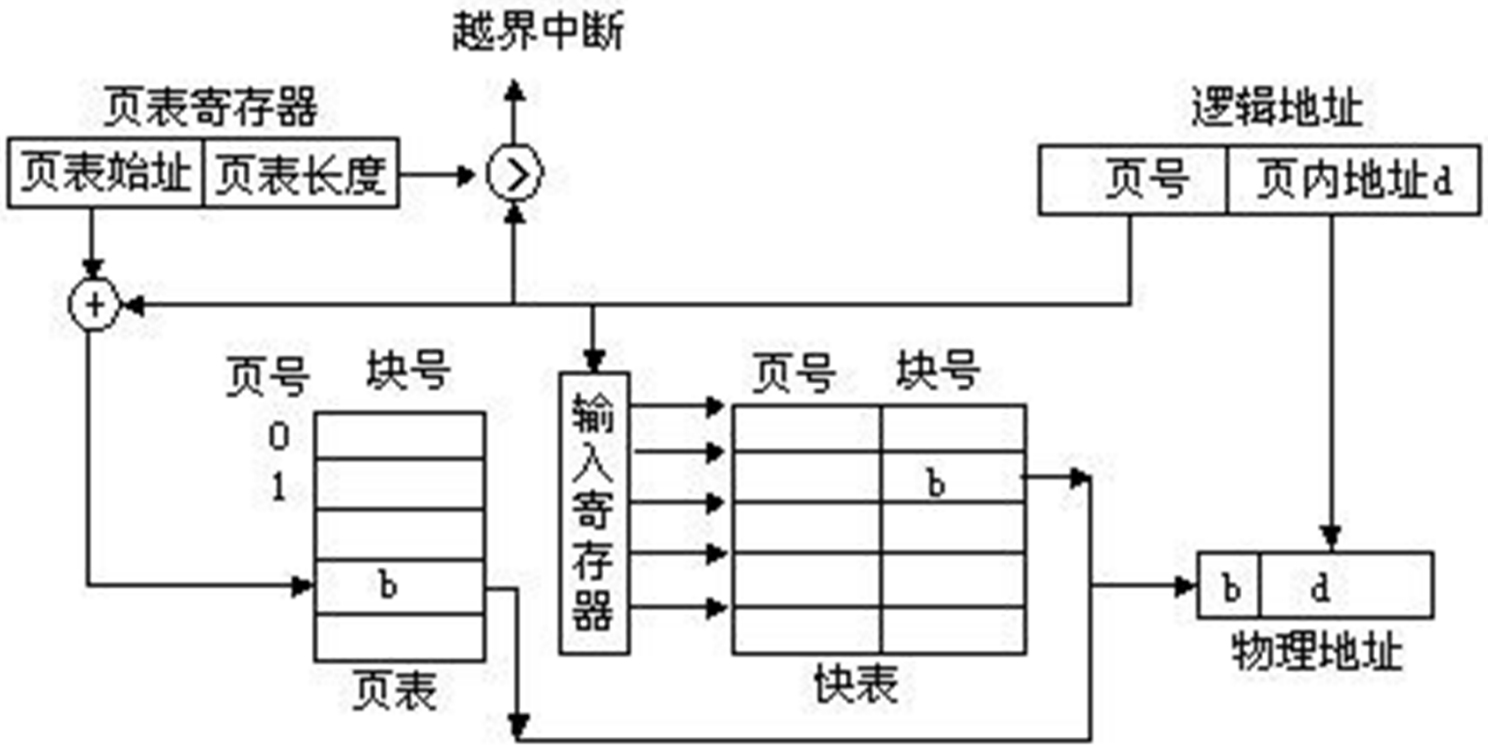
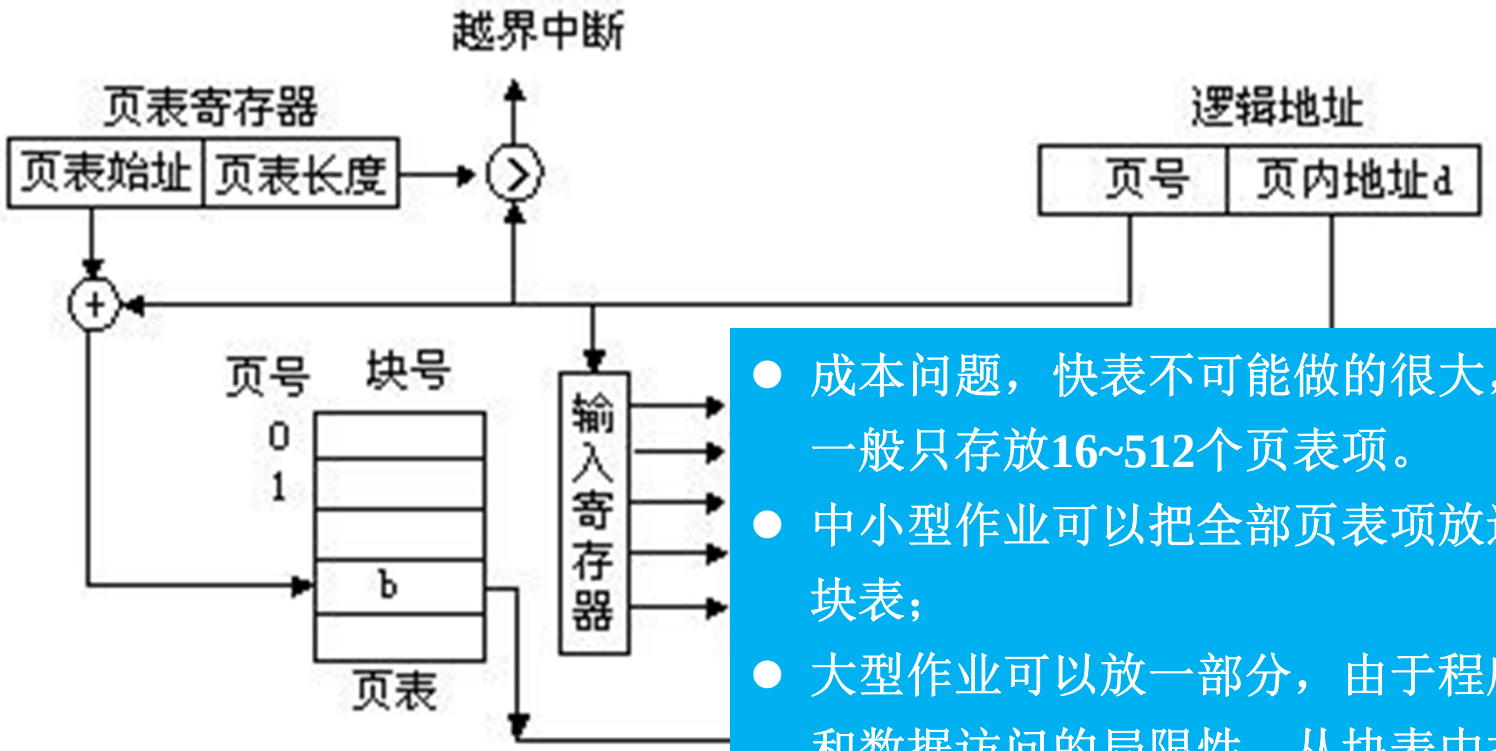


图4-16 具有快表的地址转换机构

2．具有快表的地址转换机构

由分页系统的地址转换可知，每存取一个数据，要访问2次内存，计算机处理速度降低近1/2。为提高地址转换速度，在MMU中增设一个具有并行查寻能力的特殊高速缓冲寄存器，称为**联想寄存器**，或称**快表**。



- 成本问题，快表不可能做的很大，一般只存放16~512个页表项。
- 中小型作业可以把全部页表项放进块表；
- 大型作业可以放一部分，由于程序和数据访问的局限性，从块表中找出所需页表项的概率在90%以上。

图4-16 具有

4.5.3 两级和多级页表

现在大多数计算机系统，都支持非常大的逻辑地址空间（2^{32}～2^{64}）。

考虑具有32位逻辑地址空间的分页系统：

若页面大小为4KB（即2^{12}B），则每个进程页表中的页表项可达1M个，因每个页表项占用4个字节，故每个进程仅仅页表就要占用4MB的内存空间，而且还要求是连续的。——显然这是不现实的。

解决此问题的办法有：

①采用离散分配方式来解决难于找到一块连续的大内存空间的问题；
办法：两级和多级页表

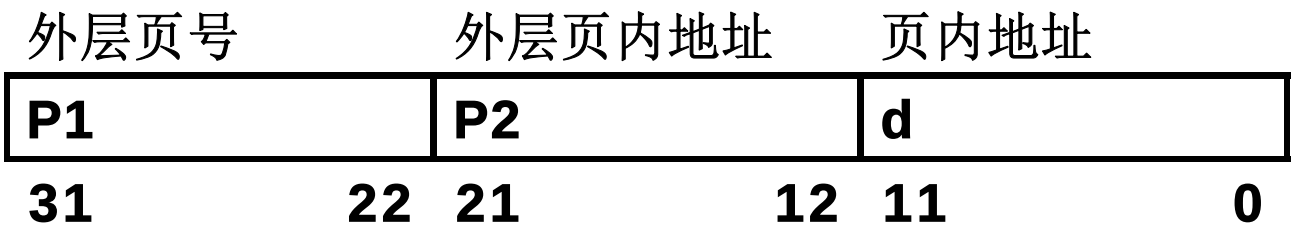
②只将当前需要的部分页表项调入内存，其余的页表项驻留在磁盘上，需要时再调入；

1. 两级页表

采用离散方式来解决难于找到一块连续的大内存空间的问题，解决的办法是：

- 将页表分页
- 将各个页面离散地存放在不同的物理块中
- 为离散分配的页表再建立一张页表，称为外层（外部）页表

【例 】以32位逻辑地址空间为例，当页面大小为4KB（12位）时，采用两级页表结构时，再对页表分页，使每个页中包含2^{10}（1024）个页表项，则最多需要1024个页存放页表，即外部页表中页号P1为10位，外部页表中的外部页内地址P2也是10位。此时逻辑地址结构如下：



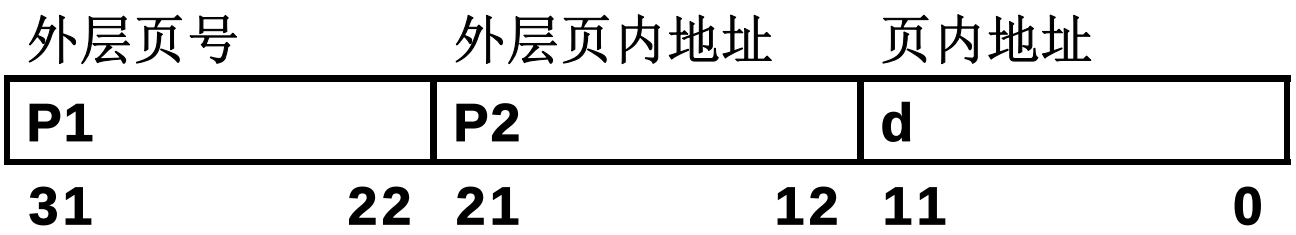
1. 两级页表

采用离散方式来解决难于找到一块连续的大内存空间的问题，解决的办法是：

- 将页表分页
- 将各个页面离散地存放在不同的物理块中

- 页表的每个表项中，存放的是进程的某页在内存中的物理块号，如0#页存放在1#物理块中，1#页存放在4#物理块中。
- 外层页表的每个页表项存放的是某页表分页的首址，如0#页表存放在1011#物理块中。

如下：



两级页表结构示意图

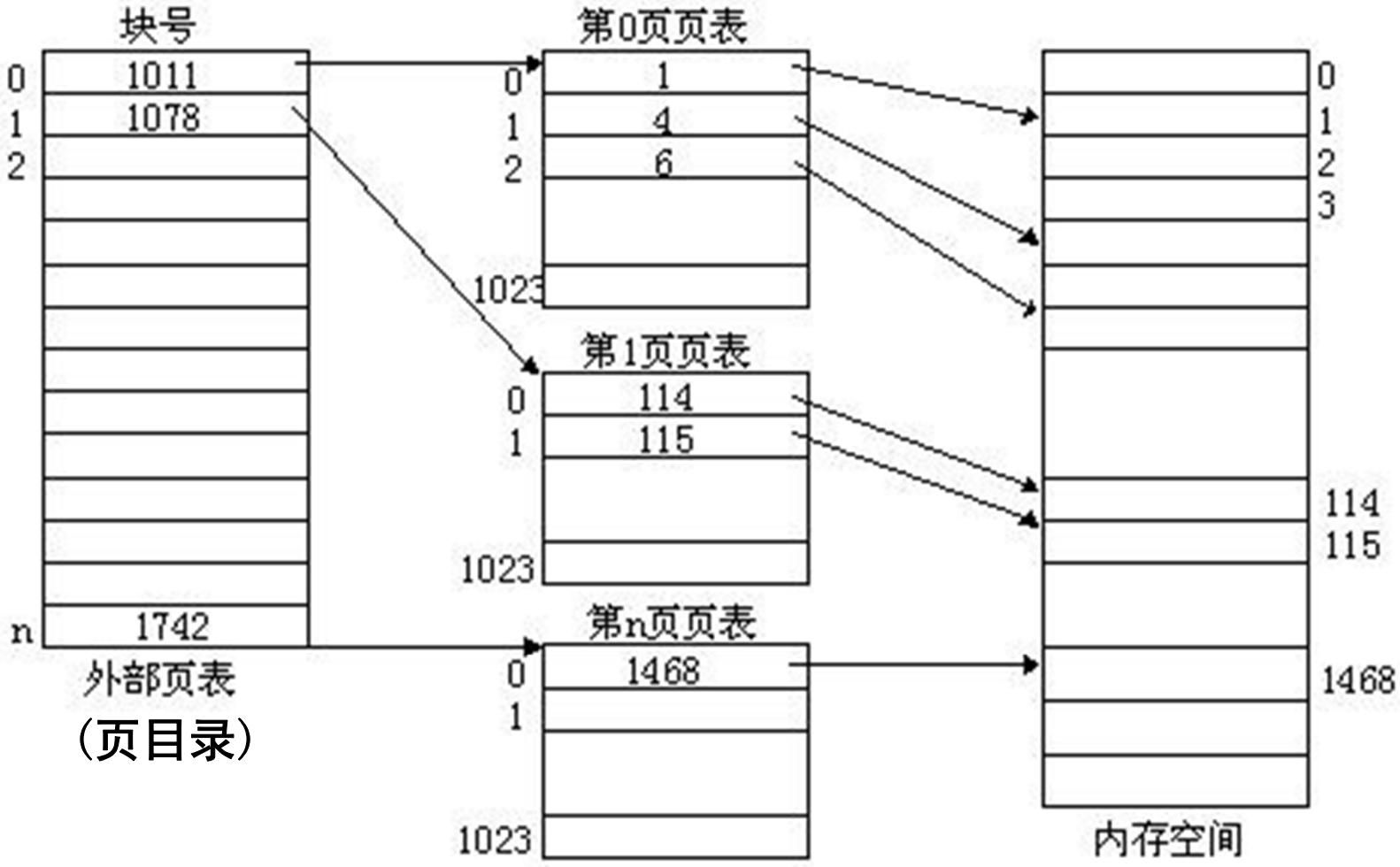


图4-18 两级页表结构

两级页表结构示意图

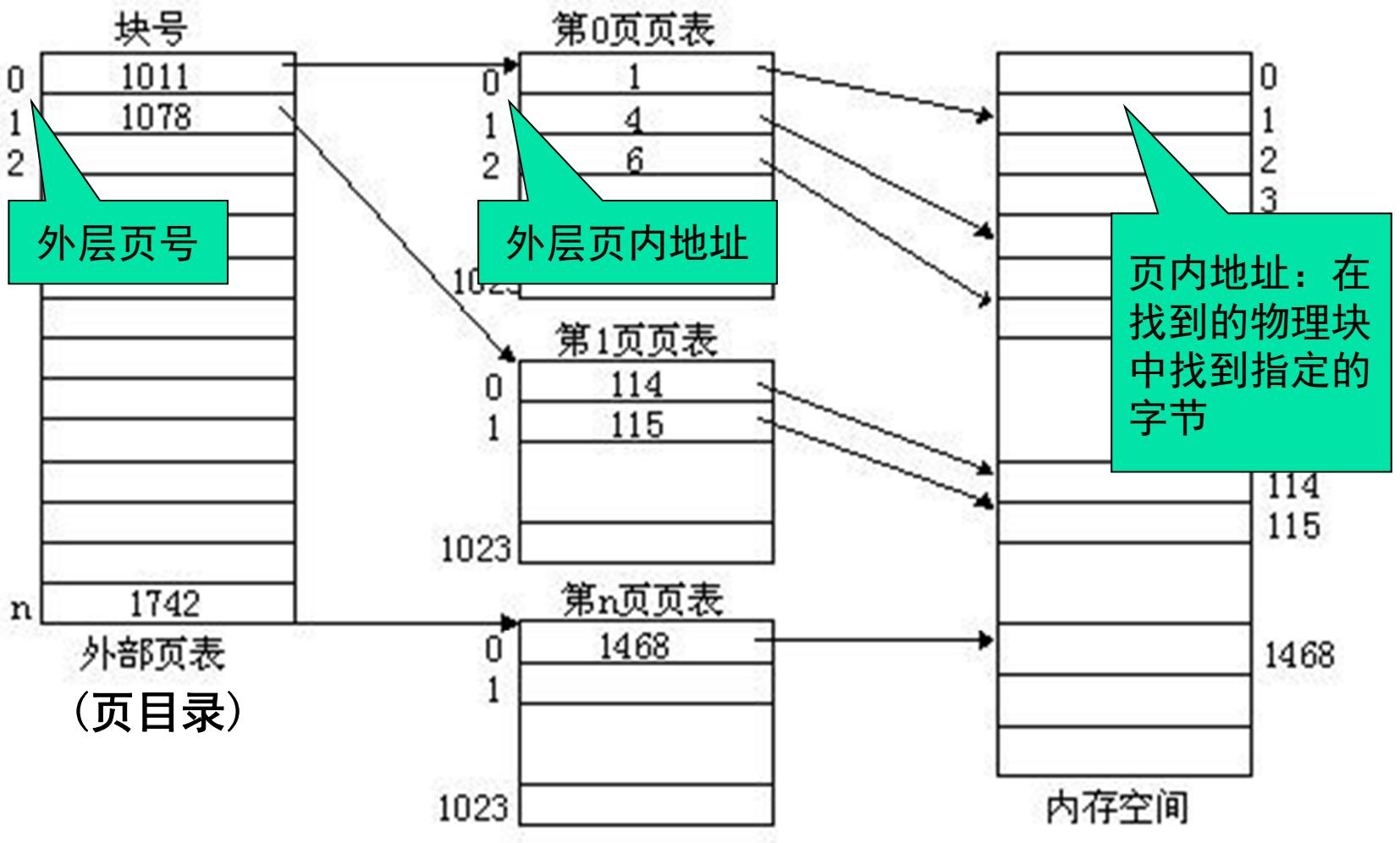
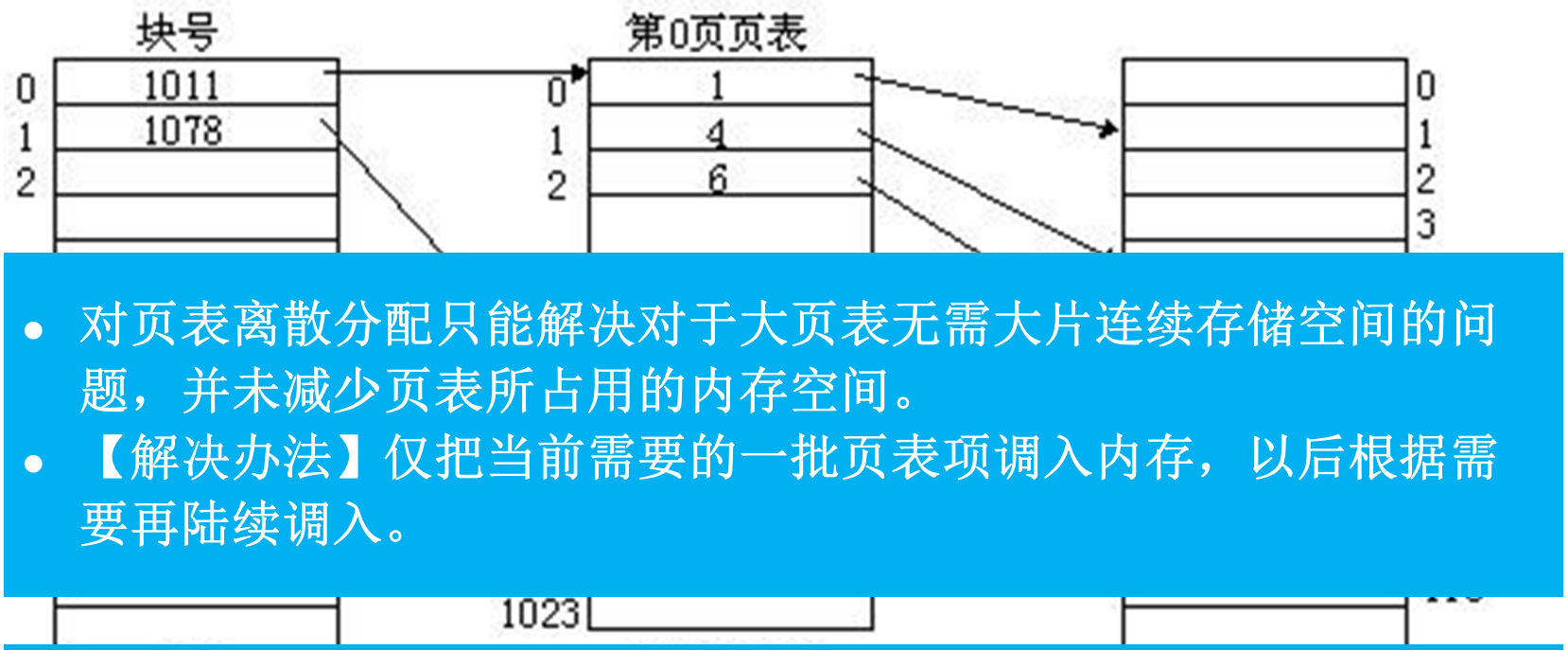


图**4-18** 两级页表结构

两级页表结构示意图



- 对页表离散分配只能解决对于大页表无需大片连续存储空间的问题，并未减少页表所占用的内存空间。
- 【解决办法】仅把当前需要的一批页表项调入内存，以后根据需要再陆续调入。

- 对于正在运行的进程，先将其外层页表调入内存，对于页表分页，只需要调入一页或几页。
- 增设状态位S，值为0表示该页表分页不在内存中，为1表示已调入内存。

图4-18 两级页表结构

为地址变换方便，需设置一个外层页表寄存器。两级页表地址变换机构如图4-6所示(图中假设页的大小为4KB)。

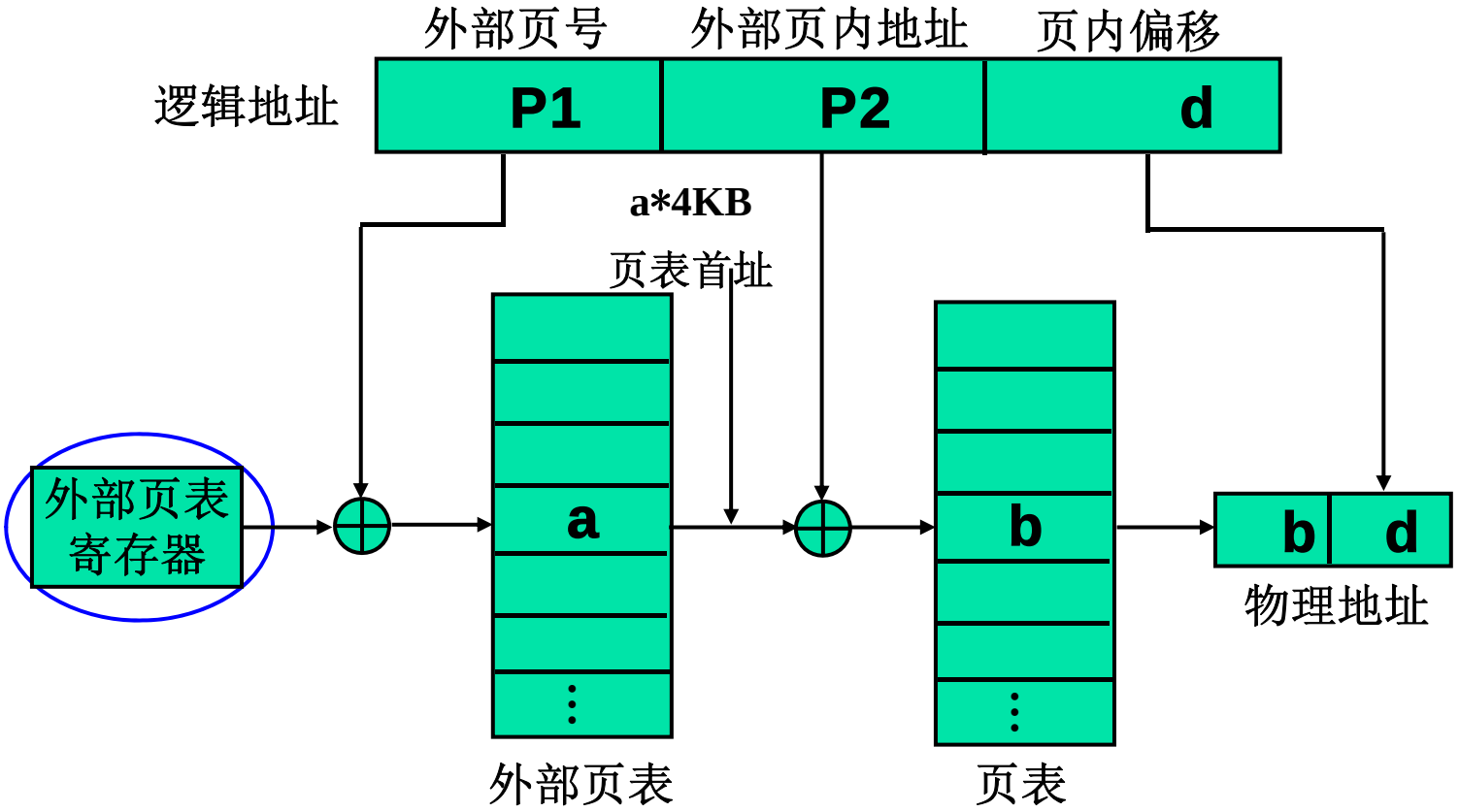


图4-15 具有两级页表的地址变换机构

为了地址变换实现上的方便起见，在地址变换机构中同样需要增设一个外层页表寄存器，用于存放外层页表的始址，并利用逻辑地址中的外层页号，作为外层页表的索引，从中找到指定页表分页的始址，再利用P2作为指定页表分页的索引，找到指定的页表项，其中即含有该页在内存的物理块号，用该块号和页内地址d 即可构成访问的内存物理地址。

2. 多级页表

32位的机器，采用两级页表结构很合适，但对于64位的机器，采用两级页表可能仍不合适。

- 假设页面大小仍是4KB，即2^{12}B，64位的地址还剩下52位；
- 若仍按照物理块的大小2^{12}来划分页表，需要10位地址（每个页表项占4B空间），则余下的42位用于外层页号。外层页号有4096G个页表项，每个页表项占4B，则一共占据16384GB的连续内存空间；
- 对于64位的计算机，至少需要六级页表，实际推出的64位OS中，把可直接寻址的存储空间减少为40多位，才可以用三级页表管理。

4.6 分段存储管理方式

4.6.1 分段存储管理方式的引入

存储管理方式从从单一连续分配到固定分区分配，再到动态分配和分页存储管理方式，推动其发展的主要动力都是直接或间接地出于提高内存利用率的目的。

4.6 分段存储管理方式

4.6.1 分段存储管理方式的引入

主要是为了满足用户和程序员在编程和使用上的多方面需要：

方便编程

通常用户把自己的程序按逻辑关系分为若干个段，每段都从0开始编址，并有自己的名字和长度。因此，希望要访问的逻辑地址是由段名(段号)和段内偏移量(段内地址)决定的。

信息共享

在实现对程序和数据的共享时，是以信息的逻辑单位为基础的，比如，共享某个例程和函数。分页系统中的“页”只是存放信息的物理单位（块），并无完整的意义，不便于实现共享；而段却是信息的逻辑单位。

信息保护

信息保护同样是对信息的逻辑单位进行保护，因此，分段管理能更有效地实现信息保护功能。

动态增长

在实际应用中，往往有些段，特别是数据段，在使用过程中会不断增长，而事先又无法确切地知道数据段会增长到多大。前述的其它几种存储管理方式，都难以应付这种动态增长的情况，而分段存储管理方式却能较好地解决这一问题。

动态链接

动态链接是指在作业运行之前，并不把几个目标程序链接起来。要运行时，先将主程序所对应的目标程序装入内存并启动运行，当运行过程中需要调用某段时，才将该段（目标程序）调入内存并进行链接。可见，动态链接也要求以段作为管理的单位。

4.6.2 分段系统的基本原理

1. 分段

(1) 作业地址空间划分成若干段，每段定义了一组逻辑信息。如

☻主程序段MAIN； ☻子程序段X

☻数据段D； ☻栈段S ……

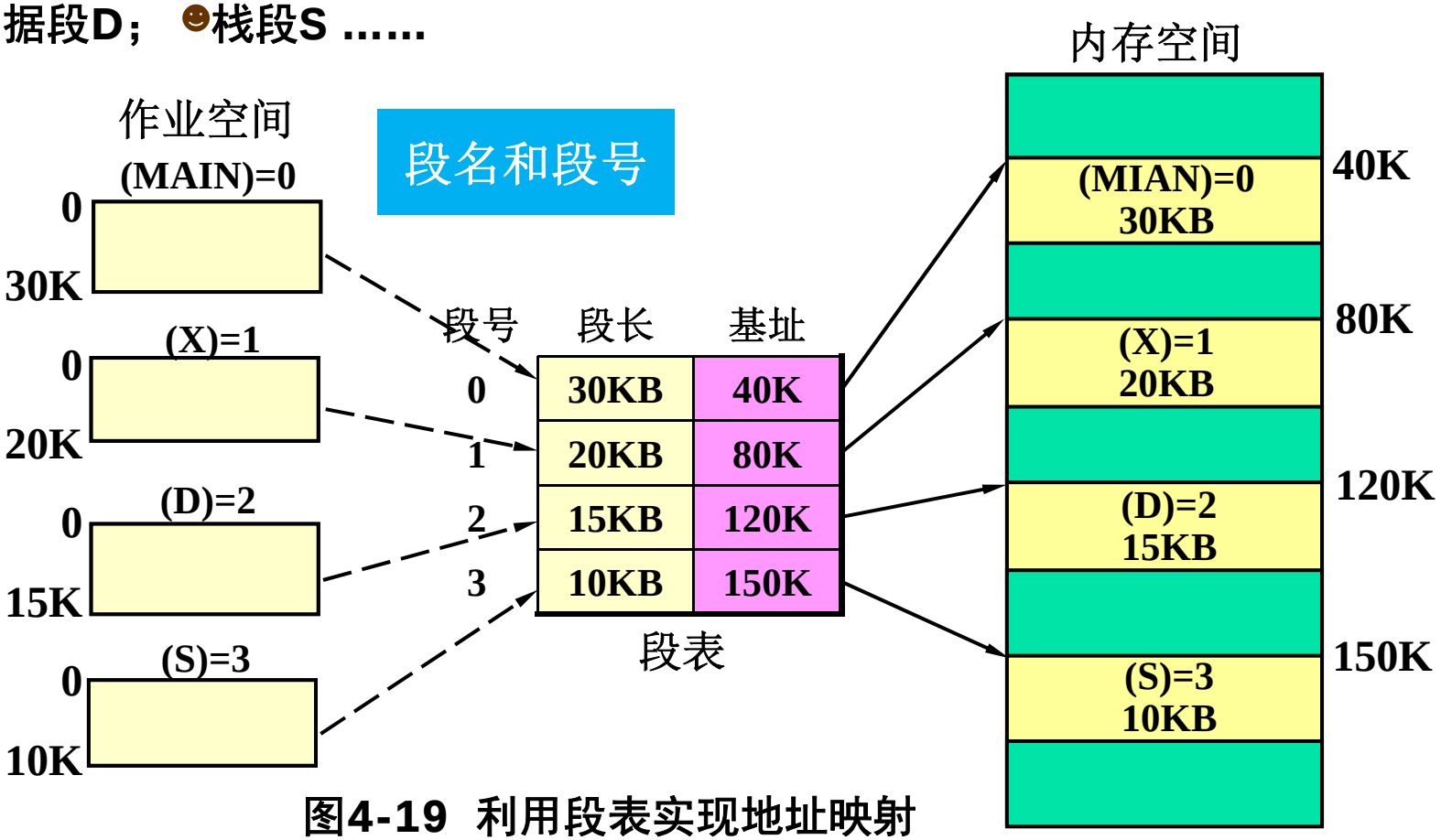


图4-19 利用段表实现地址映射

4.6.2 分段系统的基本原理

1. 分段

(1) 作业地址空间划分成若干段，每段定义了一组逻辑信息。如

☻主程序段MAIN； ☻子程序段X

☻数据段D； ☻栈段S ……

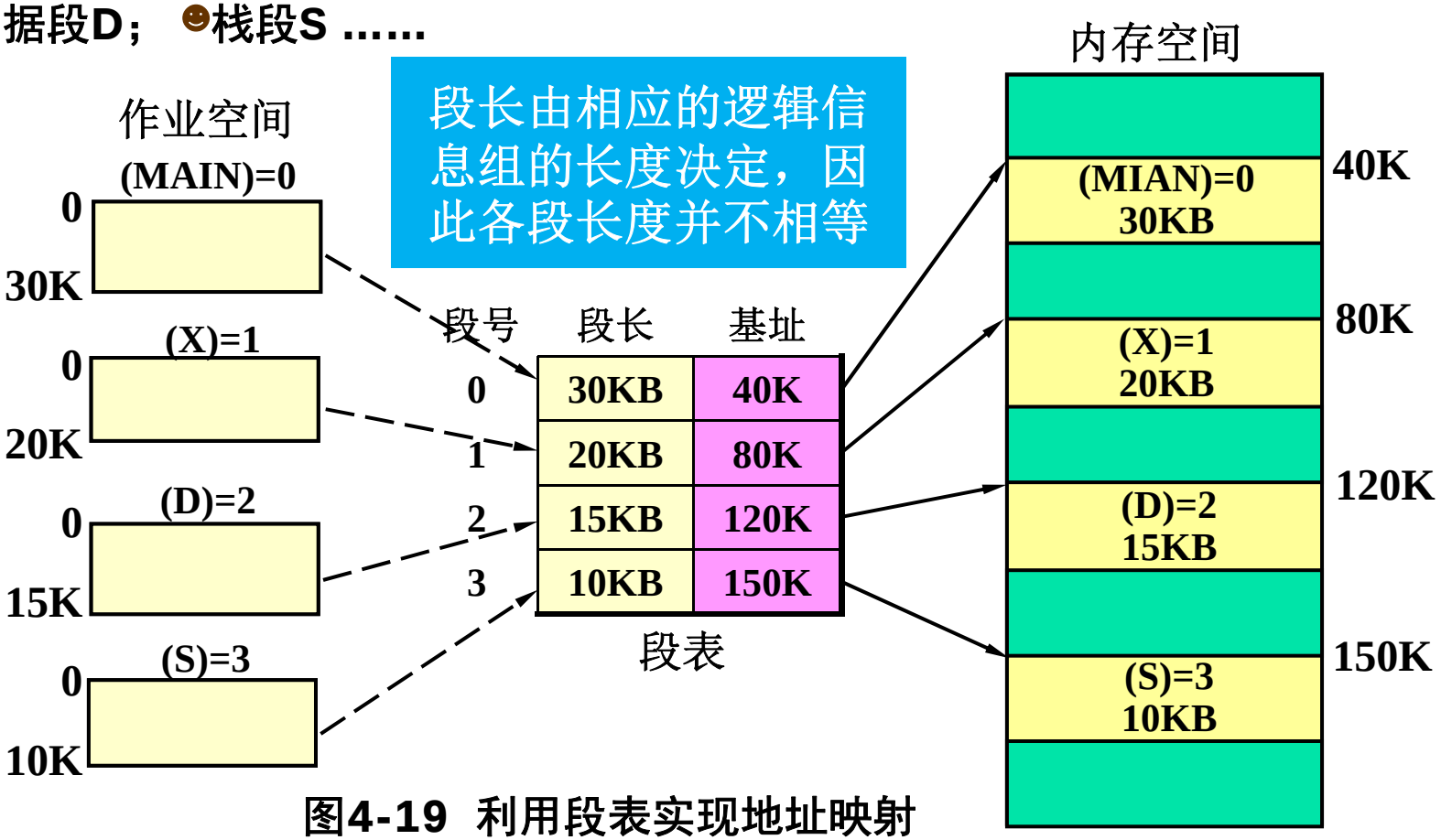
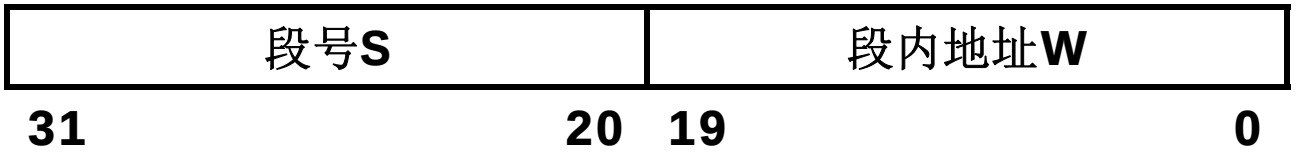


图4-19 利用段表实现地址映射

(2)每个段都有名字。为实现简单，常用段号代替段名（段号从0开始）；

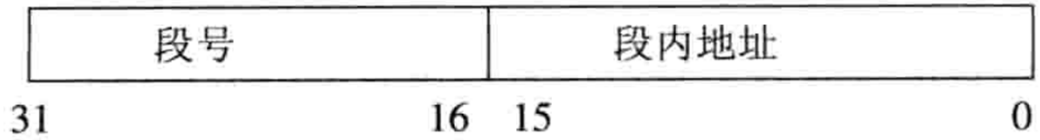
(3)每个段内都从0开始编址，并采用一段连续的地址空间。逻辑地址是二维的，由段号S和段内地址W组成。

具体结构举例如下：



该地址结构中允许作业最多有4K(2^{12})个段，每段最大长度为1MB(2^{20})。

教科书上为段号16bit，段内地址16bit，故最多有64K个段…



2. 段表

- 每个段占一个连续内存空间，各个段之间不要求连续（可以在不同的分区中）
- 故需要利用段表来进行地址变换（动态重定位）
- 段表结构：段长，基址（见图4-19）——逻辑地址→物理地址的映射。

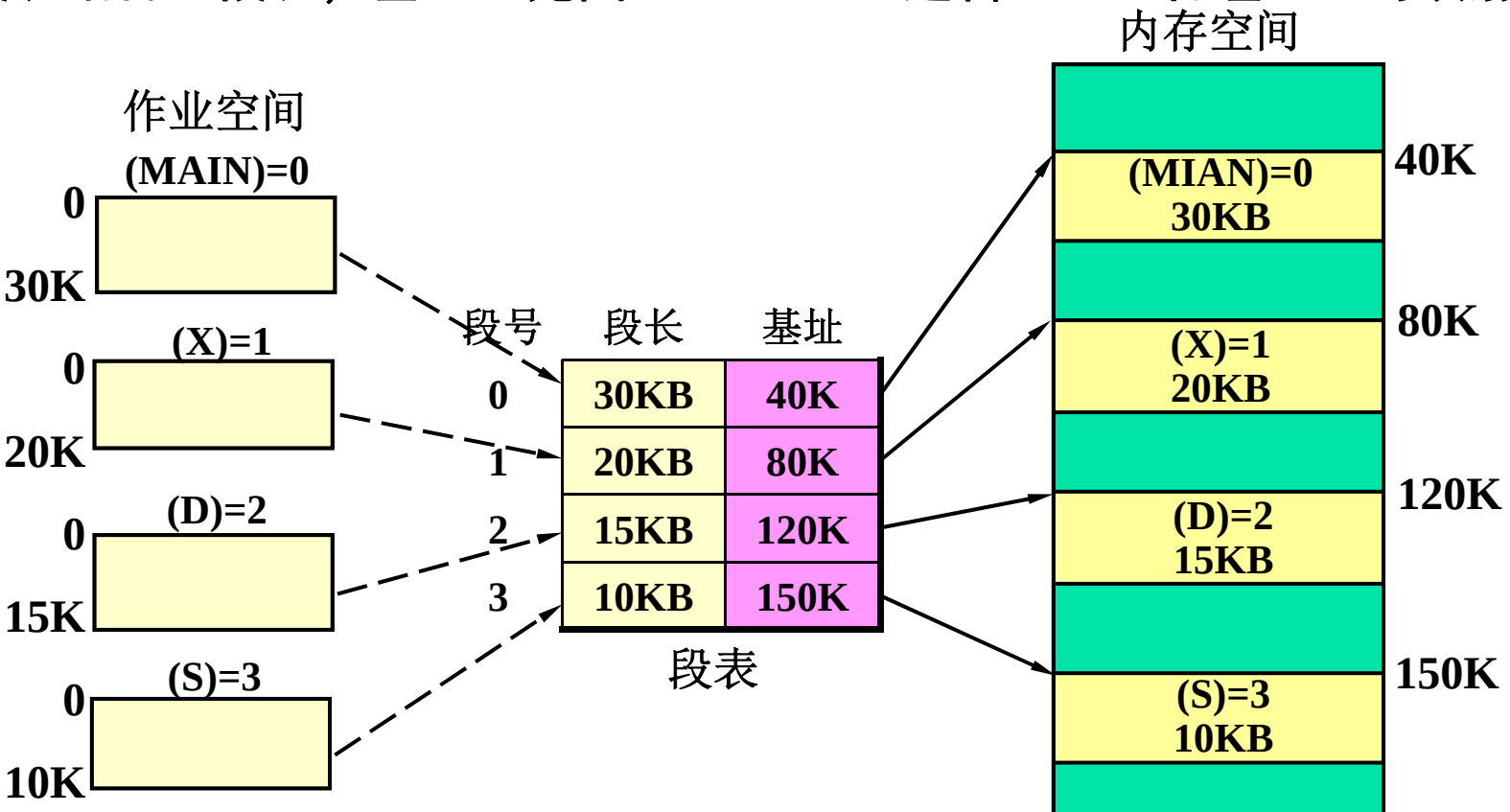
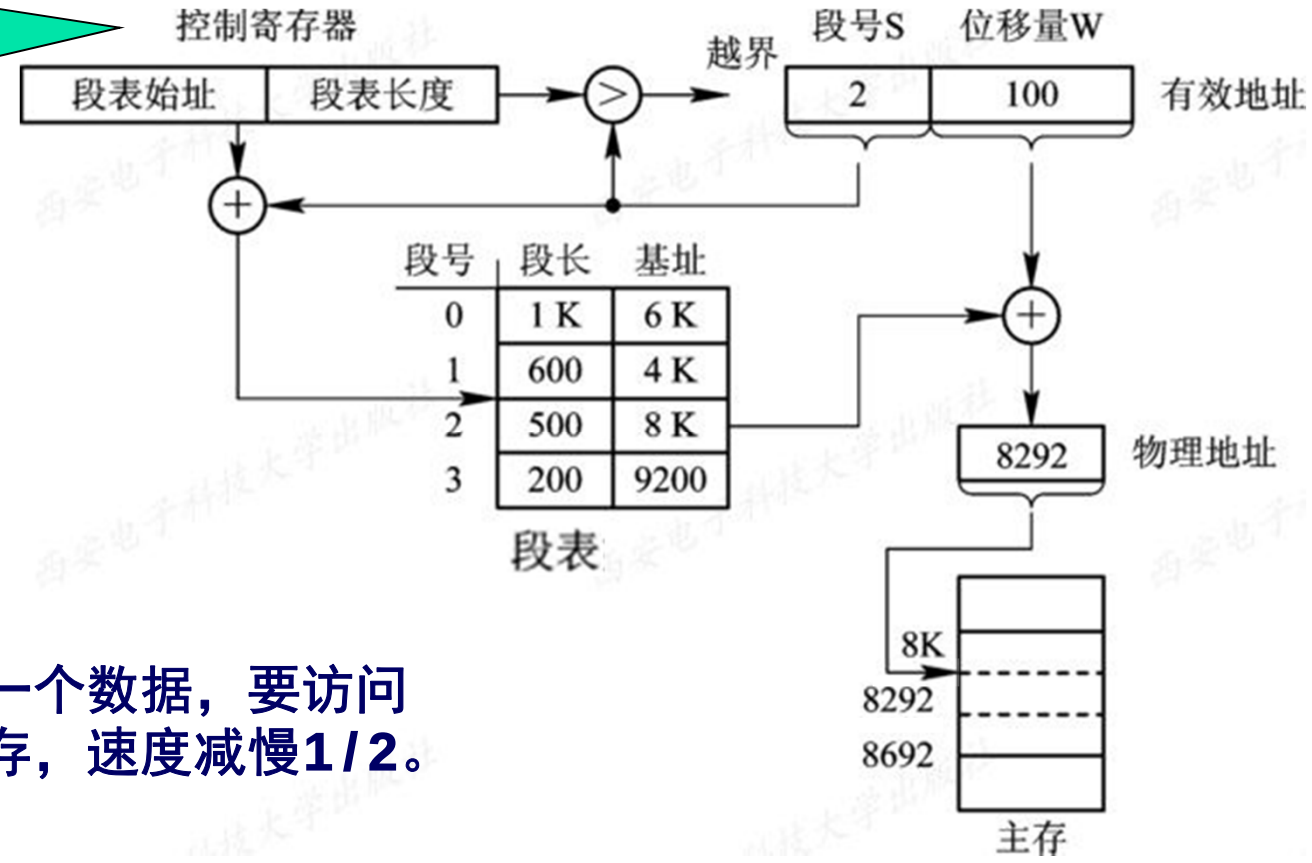


图4-19 利用段表实现地址映射

3．地址变换机构

段表寄存器：记录段表始址和段表长度TL



每访问一个数据，要访问两次内存，速度减慢1/2。

分页和分段的主要区别

1.信息单位不同

页是信息的物理单位，段则是信息的逻辑单位。

2.大小不同

页的大小固定，而段的大小不固定。

3.维数不同

分页存储管理方式的程序逻辑地址空间是一维的，而分段存储管理方式的程序逻辑地址空间是二维的。

4.6.3　信息共享

1. 分页系统中对程序和数据的共享

在分页系统中，虽然也能实现对程序和数据的共享，但远不如分段系统来得方便。

我们通过一个例子来说明这个问题。

文本编辑程序Text Editor，程序有160KB的代码和另外40KB的数据区。
可同时接纳40个用户

若代码不可重入（不可共享）——8000KB

代码可重入（可共享）——1760KB

假定每个页面大小为4KB

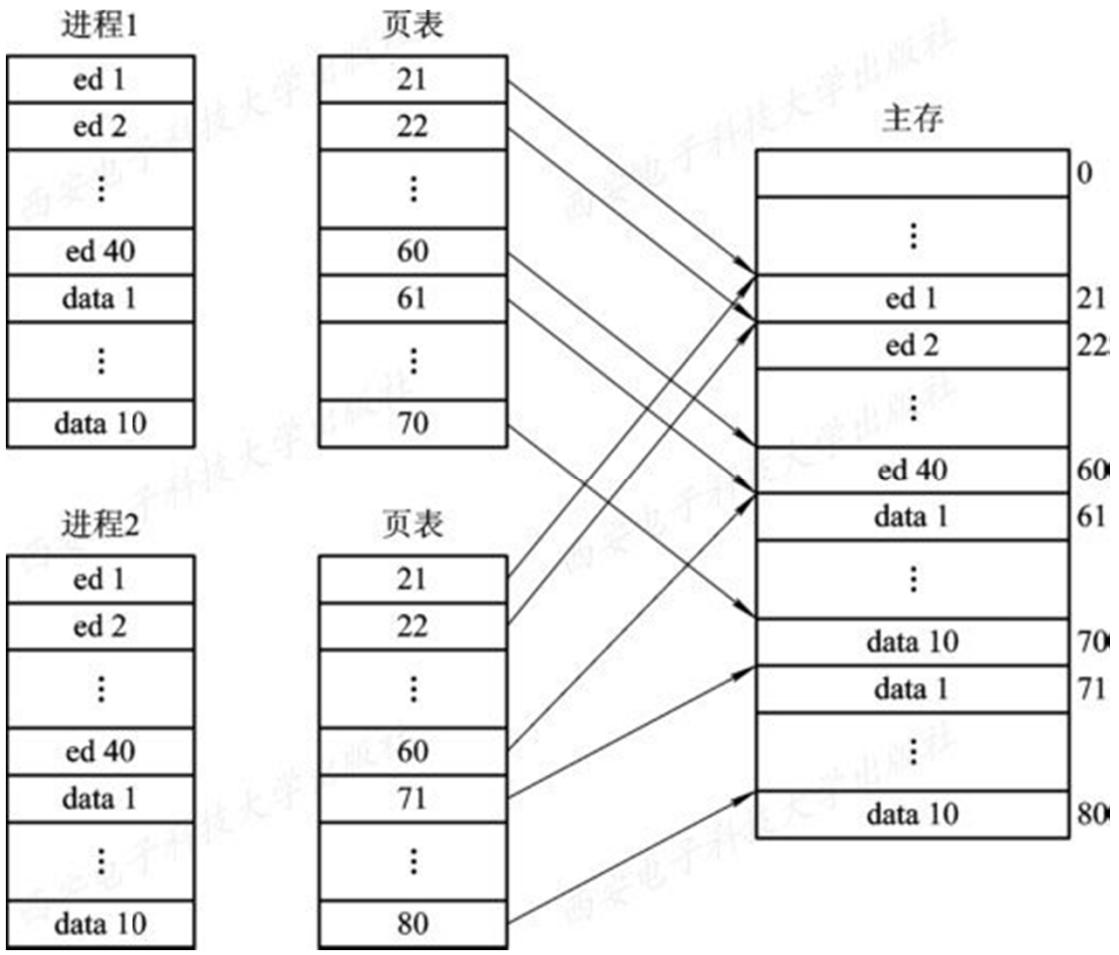


图4-21 分页系统中共享editor的示意图

文本编辑程序Text Editor，程序有160KB的代码和另外40KB的数据区。
可同时接纳40个用户

若代码不可重入（不可共享）——8000KB

代码可重入（可共享）——1760KB

假定每个页面大小为4KB

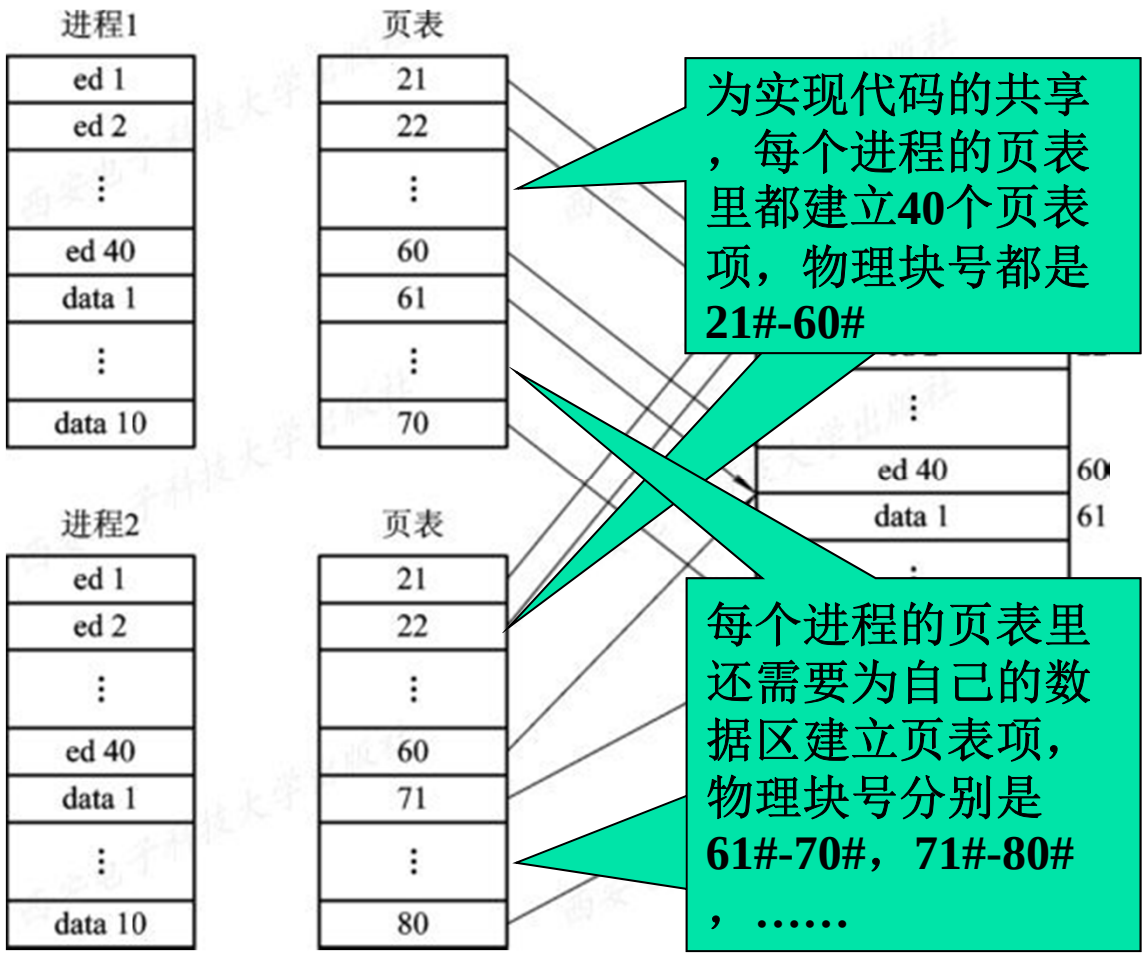
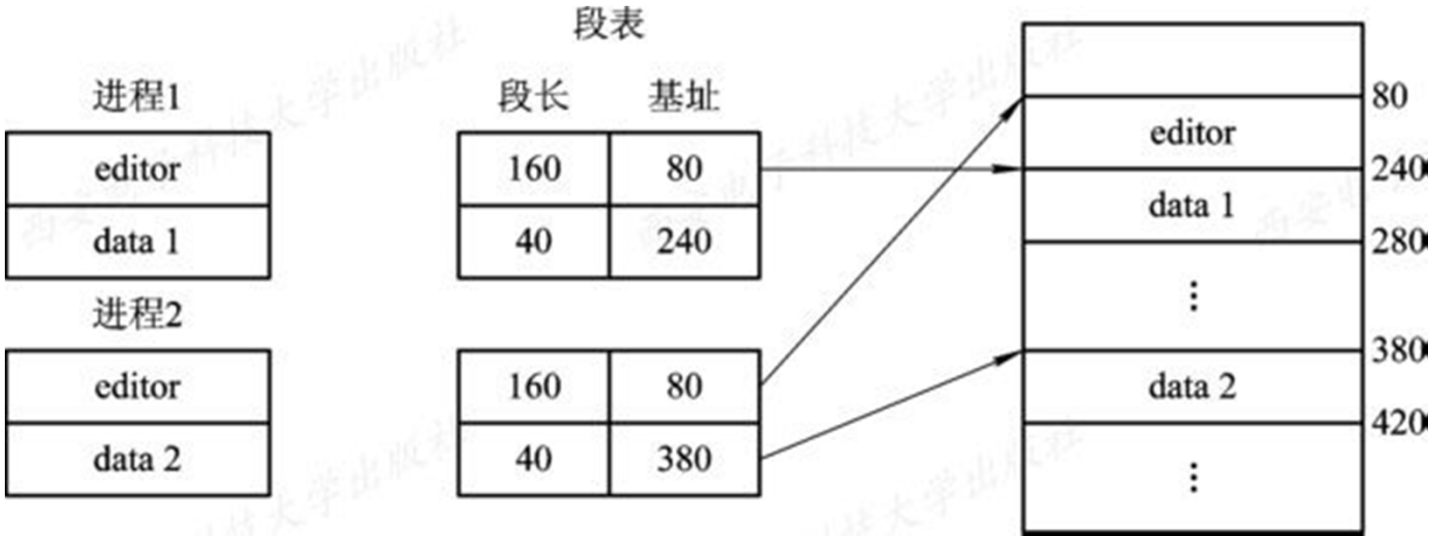


图4-21 分页系统中共享editor的示意图

2. 分段系统中程序和数据的共享

在分段系统中，由于是以段为基本单位的，不管该段有多大，我们都只需为该段设置一个段表项，因此使实现共享变得非常容易。我们仍以共享editor为例，此时只需在(每个)进程1和进程2的段表中，为文本编辑程序设置一个段表项，让段表项中的基址(80)指向editor程序在内存的起始地址。图4-22是分段系统中共享editor的示意图。



4.4.4 段页式存储管理方式

分页系统能有效地提高内存利用率
分段系统能很好地满足用户的需要

取长补短
段页式存储管理

- 分段可共享、可动态链接
- 解决内存的外部碎片问题

1. 基本原理

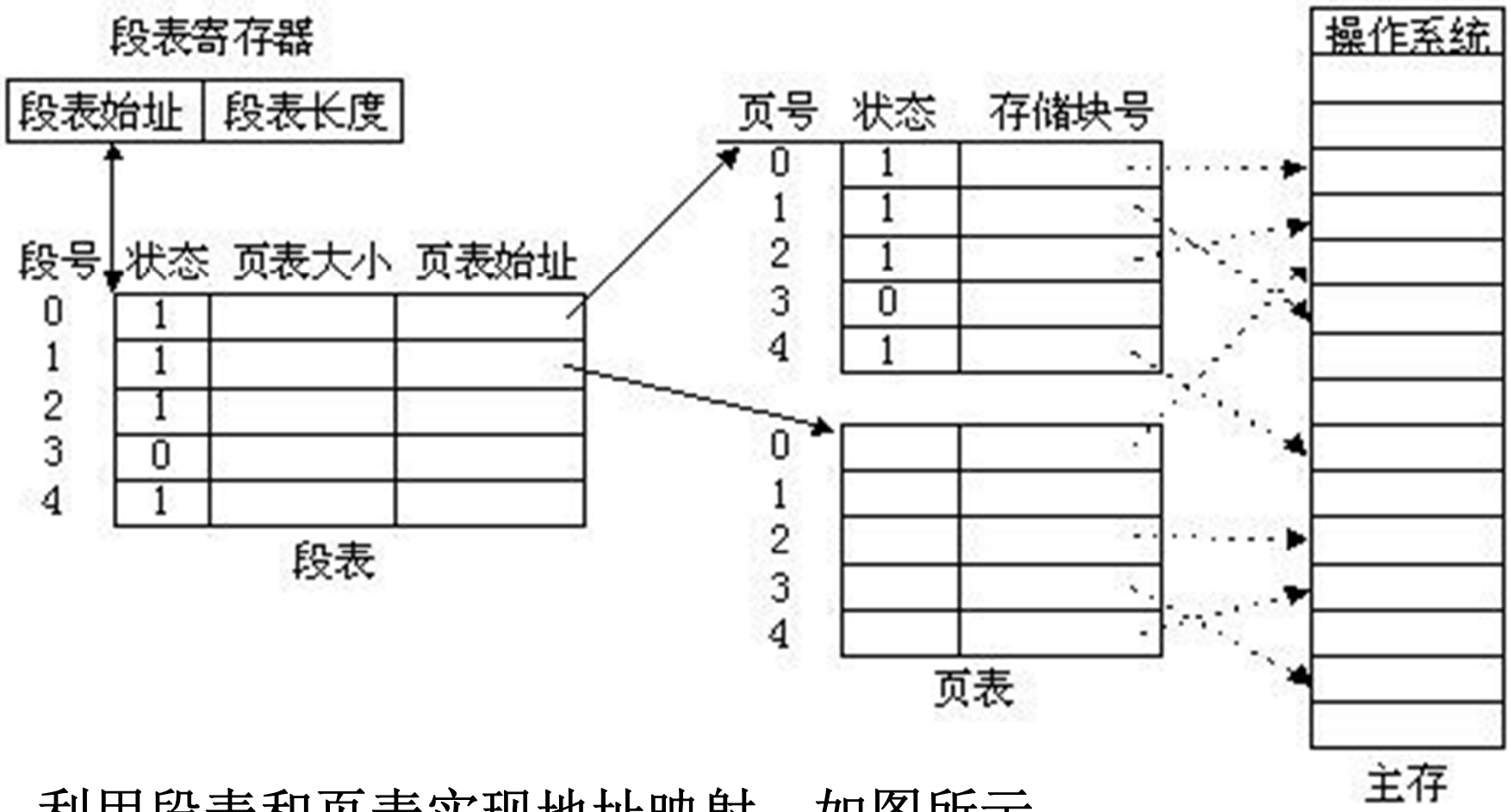
先将用户程序分成若干段，再把每段分成若干页。

逻辑地址由3部分组成：段号、段内页号、页内地址。
如下所示：

段号S	段内页号P	页内地址W

利用段表和页表实现地址映射，如图所示。

4.4.4 段页式存储管理方式



利用段表和页表实现地址映射，如图所示。

2．地址变换过程

需配置一个**段表寄存器**，其中存放段表始址、段表长度。原理见图**4-25**。

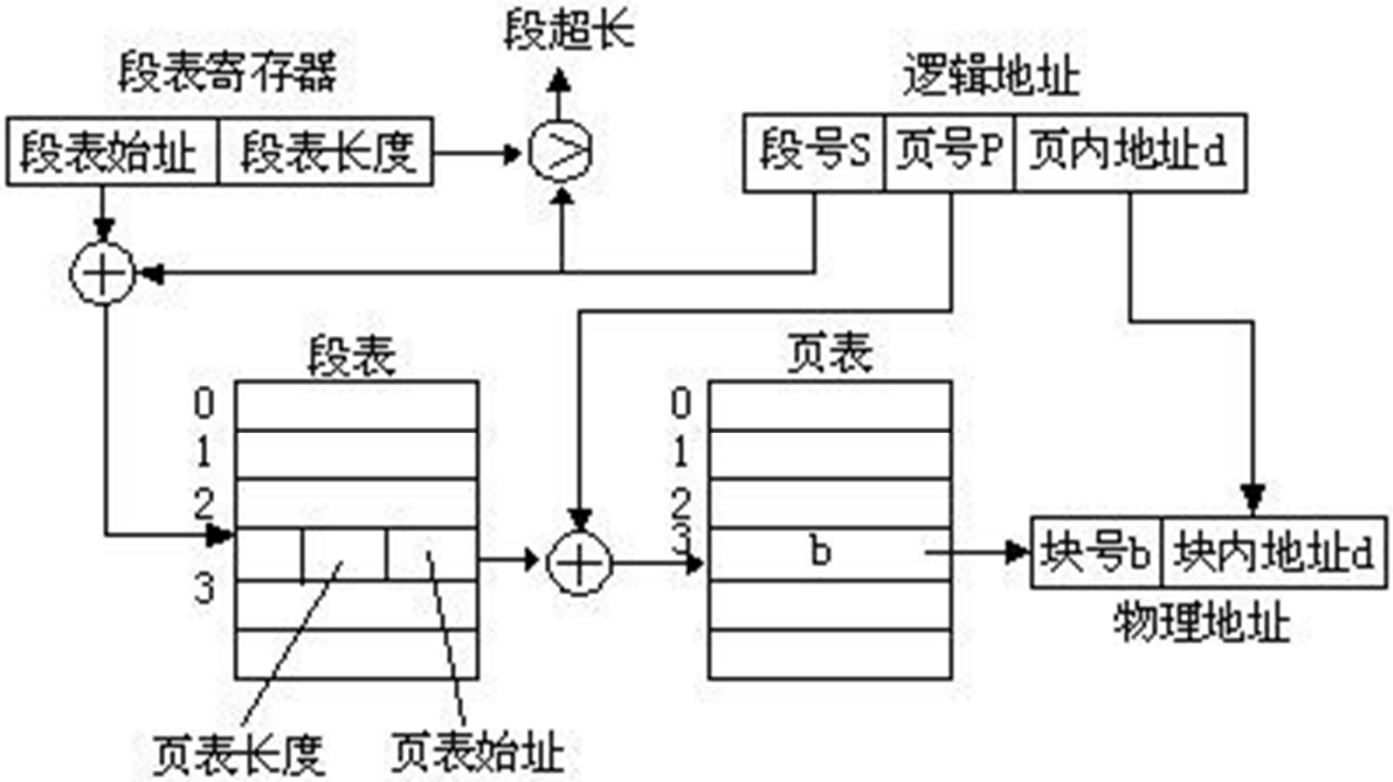
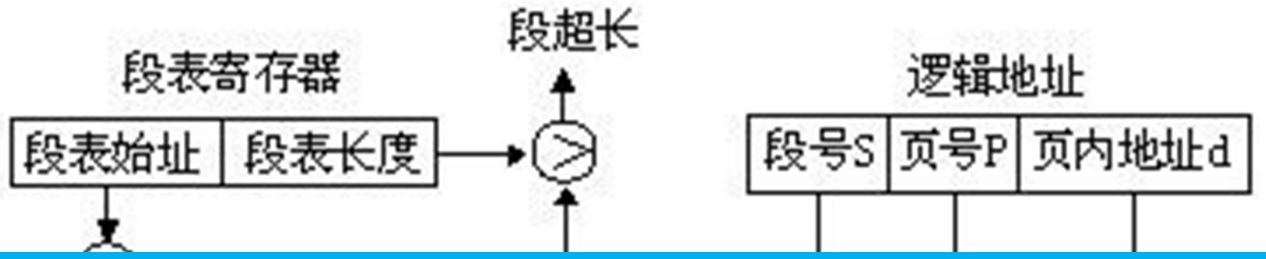


图**4-25** 段页式系统的地址变换机构

访问**3**次内存：第一次，段表；第二次，页表；第三次，取出指令或数据。

2．地址变换过程

需配置一个**段表寄存器**，其中存放段表始址、段表长度。原理见图**4-25**。



三次访存使访存次数增加了近两倍，为了提升执行速度，在地址变换机构中增设一个高速缓冲寄存器。

- 每次访问时，先同时利用段号和页号去检索高速缓存，若找到匹配的表项，便可从中得到相应页的物理块号，用来与页内地址一起形成物理地址；
- 若未找到匹配表项，则仍需要三次访问内存。

访问3次内存：第一次，段表；第二次，页表；第三次，取出指令或数据。