第四章 存储器管理

4.1 存储器的层次结构

4.2 程序的装入和链接

4.3 连续分配存储管理方式

4.4 对换（Swapping）

4.5 分页存储管理方式

4.6 分段存储管理方式

本章学习目标

- 本章首先介绍了存储管理的研究对象和目的，明确了存储管理的基本功能和原理；然后从连续、离散（实存、虚存）两个角度，分别介绍了常用的几种存储管理方案；最后介绍了当前主流操作系统中存储管理实例。

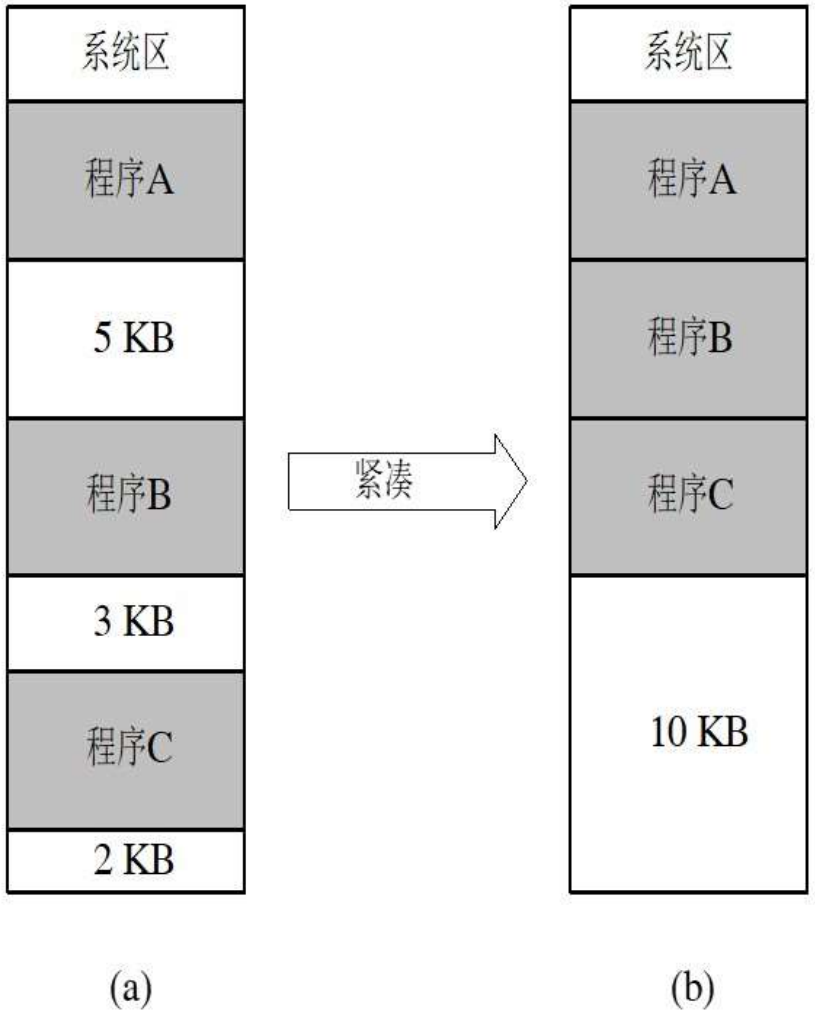
存储管理方式一览表

4.3.6 动态可重定位分区分配

1.紧凑

在动态分区分配中，消除了固定分区管理造成的“内碎片”，但是不可避免的在内存空间造成“外碎片”。多个无法利用的小分区所形成的“碎片”是一种很大的资源浪费。

如图（a）所示，空闲分区之和为10 KB，但是无法装入一个8 KB的程序。



4.3.6 动态可重定位分区分配

2.动态重定位

动态地址重定位是在程序执行期间进行的。在CPU访问内存之前,将要访问的程序或数据地址转换成内存地址。

通过基地址寄存器、变址寄存器计算出指令的有效地址，再利用硬件机构实现地址映射，这样的硬件设备称为存储管理单元MMU(Memory-Management Unit)。

通常采用的办法是利用一个重定位寄存器，对每一个有效地址都要加上重定位寄存器中的内容，以形成绝对地址。

4.3.6 动态可重定位分区分配

动态地址重定位的优点是程序在内存中的搬移不会对程序的正确执行造成影响，使内存得以被充分利用，其缺点是需要附加的硬件支持，实现存储管理的软件算法比较复杂。

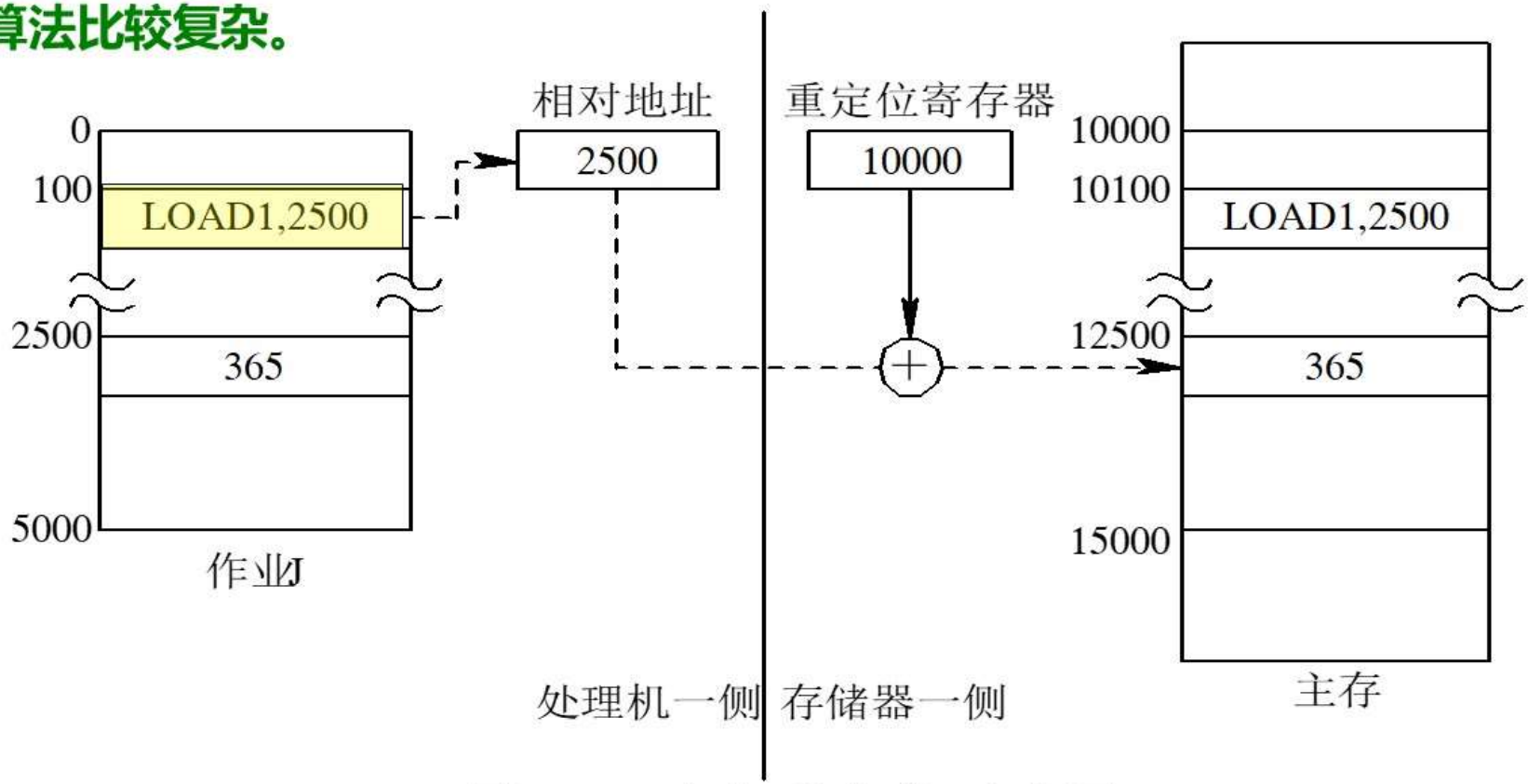


图 4-12 动态重定位示意图

4.3.6 动态可重定位分区分配

3. 动态重定位分区分配算法

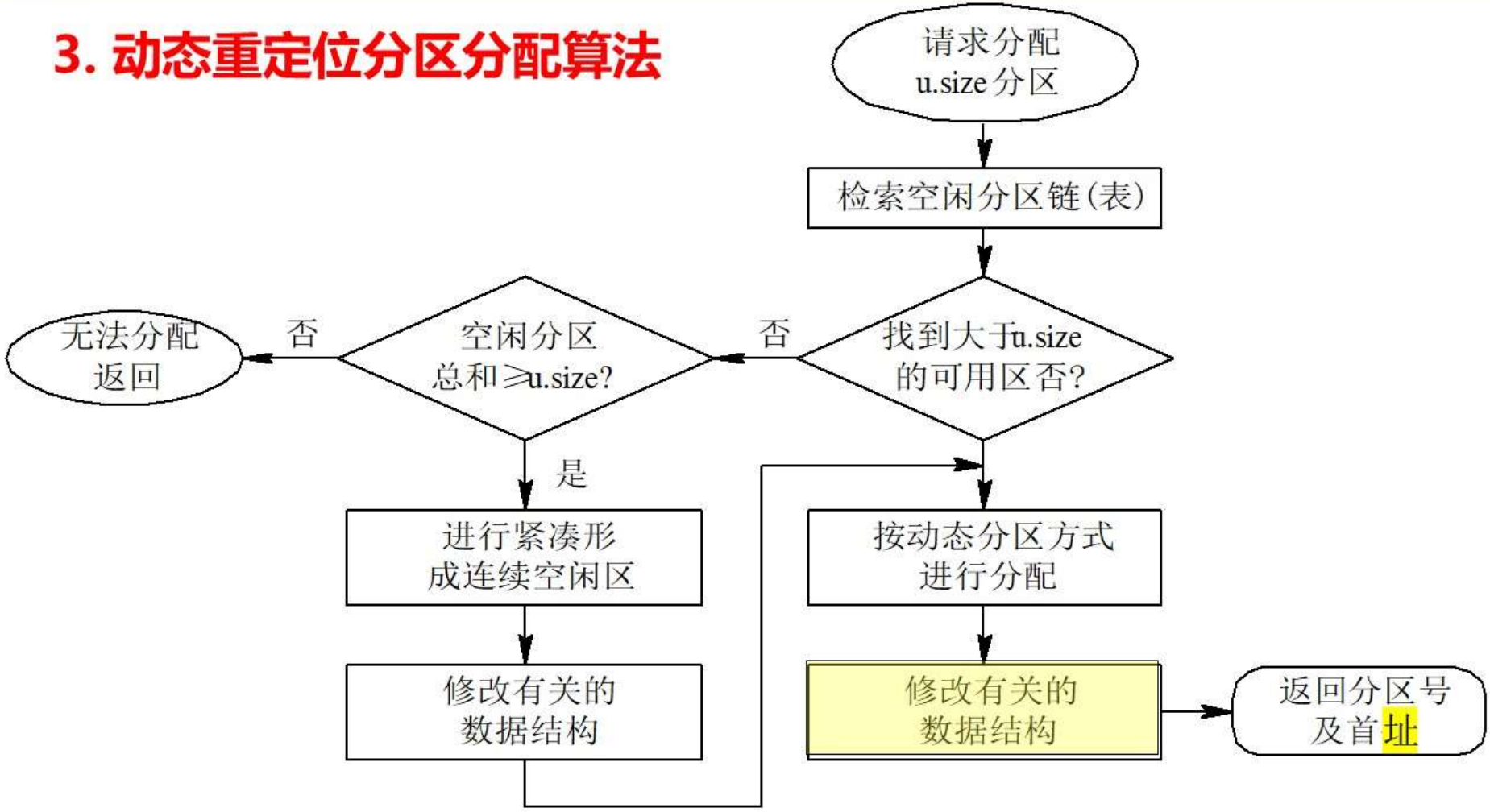


图 4-13 动态分区分配算法流程图

4.4.1 多道程序环境下的对换技术

1.对换的引入

所谓交换，就是系统根据需要把主存中暂时不运行的某个(或某些)作业部分或全部移到外存，而把外存中的某个(或某些)作业移到相应的主存区，并使其投入运行。所以，交换技术也叫对换或滚进滚出（roll-in,roll-out）。也有的系统叫挂起调度或中级调度。交换是一种用时间换空间的技术。

曾被广泛地运用于早期的小型分时系统的存贮管理中(以进程为单位)。

4.4.2 对换空间的管理

1.对换空间管理的主要目标

在引入对换技术的存储管理系统中，外存被划分为两个部分，一部分是文件区，另外一部分是对换区。

(1)对文件区管理的主要目标

通常文件都较长时间驻留在外存，访问频率较低，因此文件区管理的主要目标是提高文件存储空间的利用率，然后才是提高对文件的访问速度。

4.5 分页存储管理方式

动态重定位是解决存储器零头问题的一种途径，但要移动大量信息花去不少处理机时间,代价比较高,这是因为这种分配要求把作业必须安置在一连续存储区内的缘故,若作业不要求连续存放，即作业空间分割后离散存放在物理空间中，就可避免紧凑，又能充分利用存储空间，解决存储零头问题。

离散分配方式有三种：

分页存储管理

分段存储管理

段页式存储管理

4.5.1 分页存储管理的基本方法

1. 页面和物理块

①物理块：将整个系统的内存空间划分成一系列大小相等的块，每一块称为一个物理块、物理页或实页，页架或页帧（frame），可简称为块（block）。所有的块按物理地址递增顺序连续编号为0、1、2、……。

②页面：每个作业的地址空间也划分成一系列与内存块一样大小的块，每一块称为一个逻辑页或虚页，也有人叫页面，可简称为页（page）。所有的页按照逻辑地址递增顺序连续编号为0、1、2、……。

③一个作业，只要它的总页数不大于内存中的可用块数，系统就可以对它实施分配。系统装入作业时，以页为单位分配内存，一页分配一个块，作业所有的页所占的块可以不连续。

3. 页表

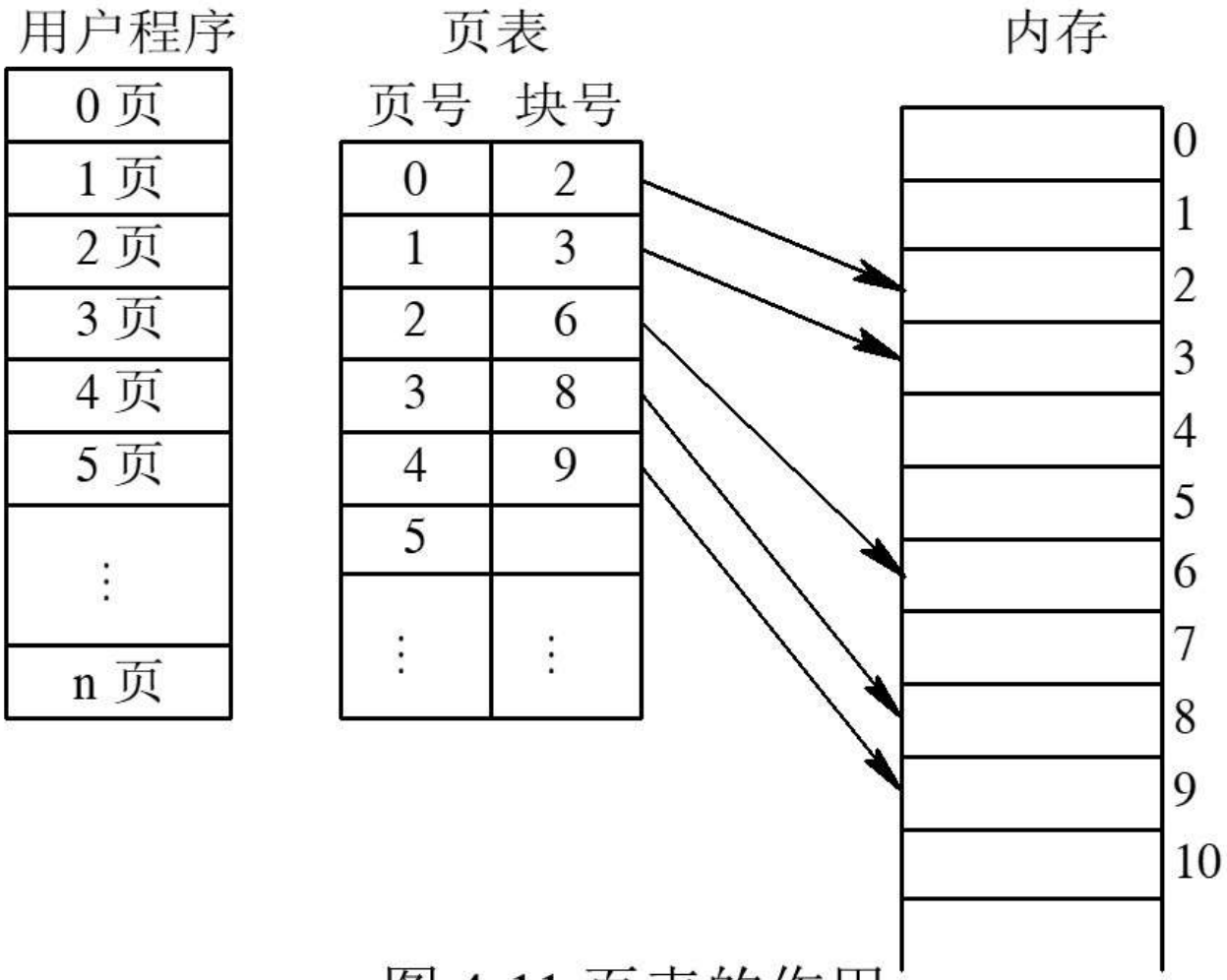
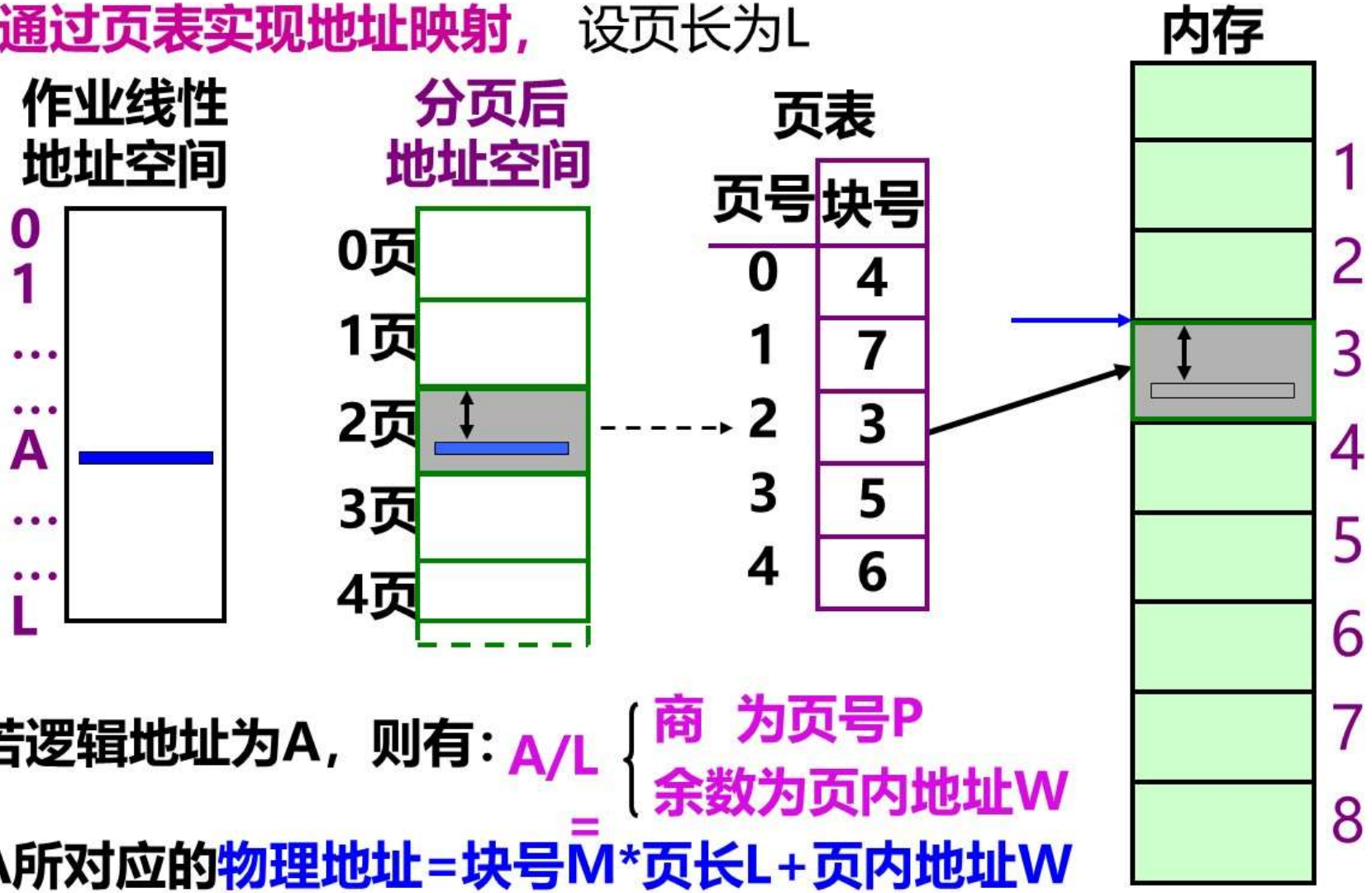


图 4-11 页表的作用

4.5.1 分页存储管理的基本方法

通过页表实现地址映射， 设页长为L



若逻辑地址为A，则有：A/L { 商 为页号P；余数为页内地址W

A所对应的物理地址=块号M*页长L+页内地址W

4.5.1 分页存储管理的基本方法

地址映射：　即将逻辑地址中页号替换为块号。

由于页面和物理块的大小相等，故页内偏移地址和块内偏移地址是相同的。将逻辑地址中的页号转换为内存中的物理块号可通过查表完成。

举例

设页长为4096

逻辑地址28024(28024/4096=6...3448)

0000 0000 0000 0000 0110 | 1101 0111 1000

页号6　　页内地址3448

设该页对应块号20，则物理地址为：
20*4096+3448=81920+3448=85368

块号20　　块内地址3448

0000 0000 0000 0001 0100 | 1101 0111 1000

物理地址85368

1. 基本的地址变换机构

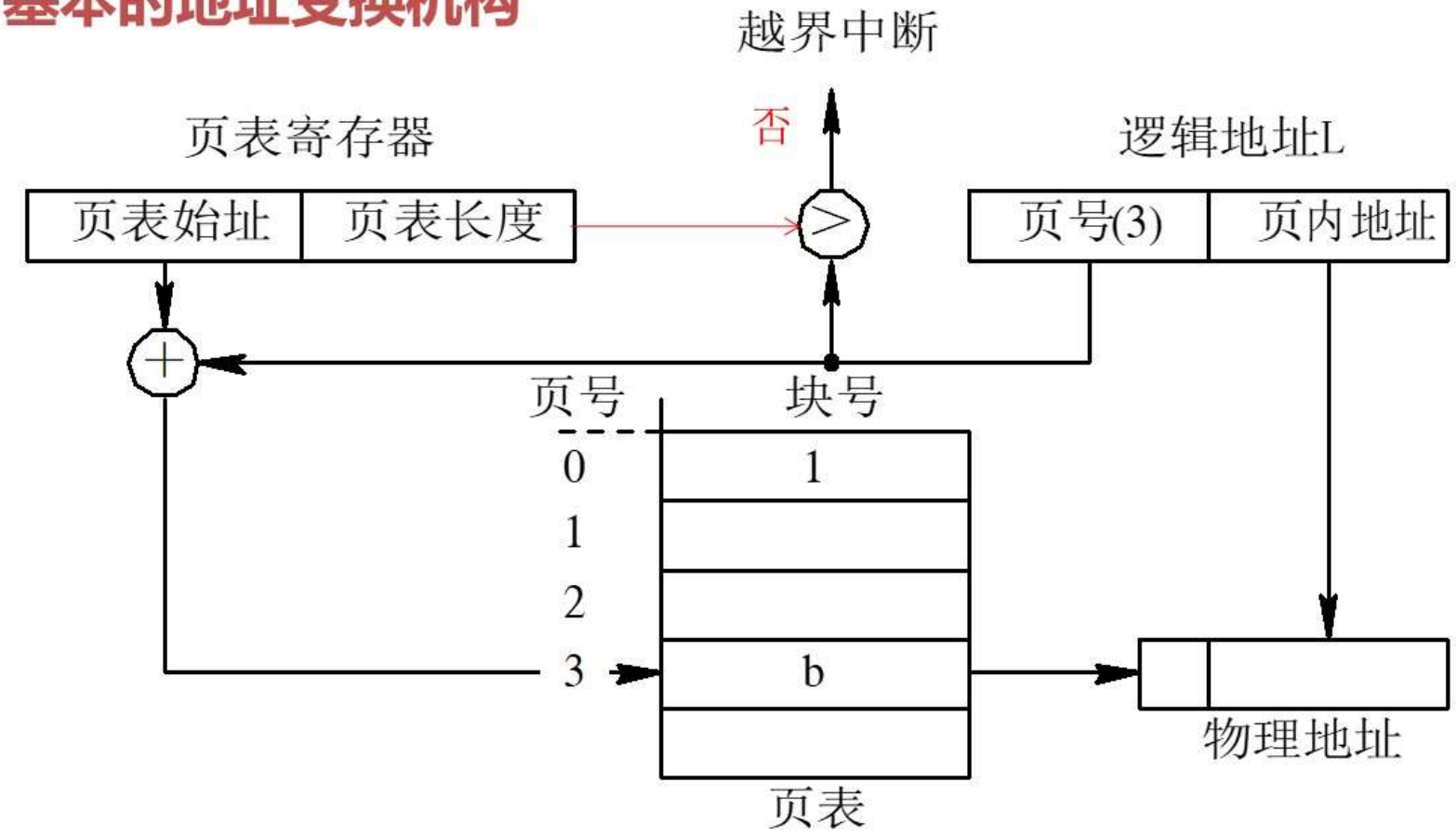


图 4-15 分页系统的地址变换机构

2. 具有快表的地址变换机构

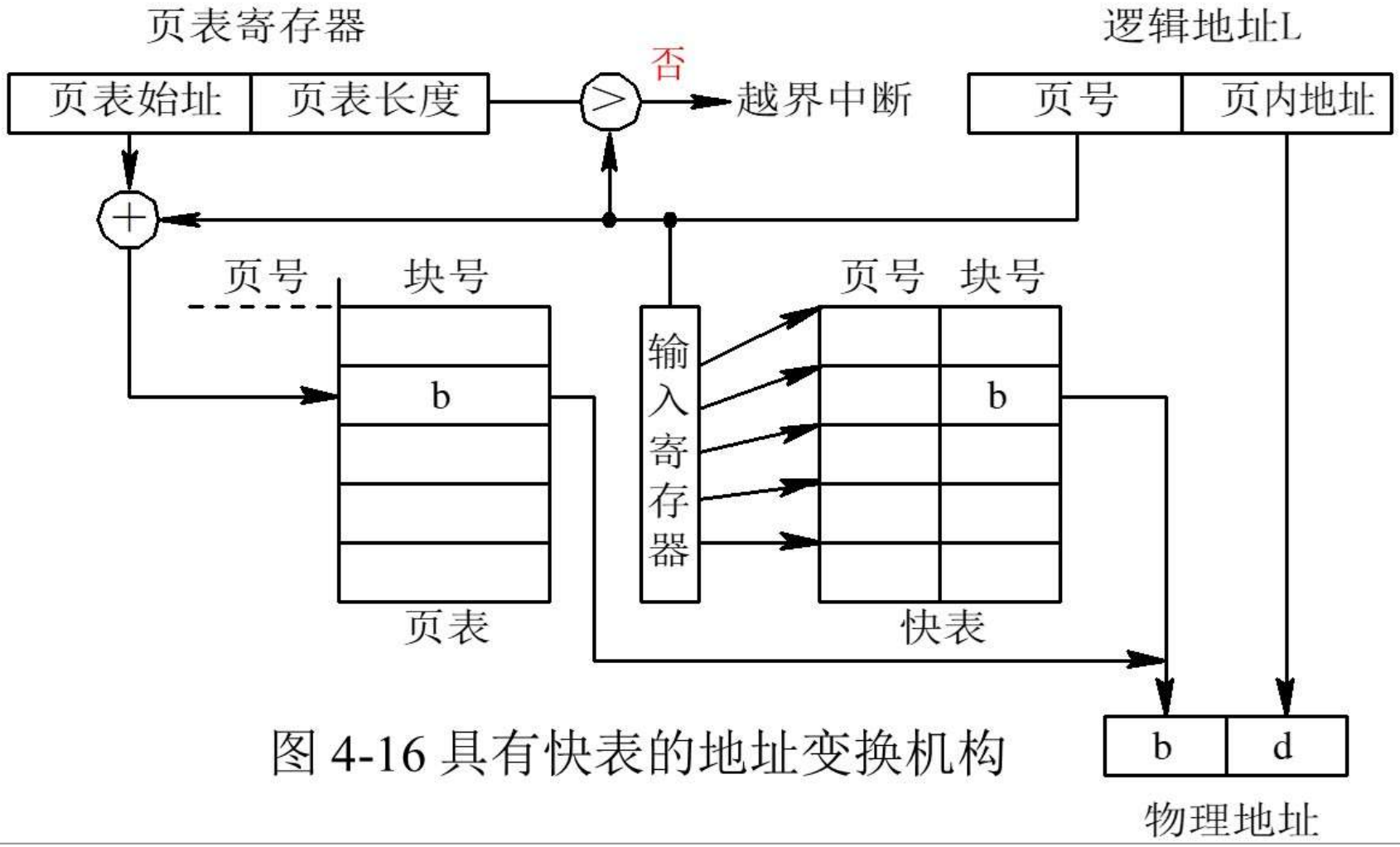


图 4-16 具有快表的地址变换机构

4.5.4 两级和多级页表

现代的大多数计算机系统，都支持非常大的逻辑地址空间(2^{32}-2^{64})。在这样的环境下，页表就变得非常大，要占用相当大的内存空间。例如，对于一个具有32位逻辑地址空间的分页系统，规定页面大小为4 KB即2^{12} B，则在每个进程页表中的页表项可达1兆个之多。设每个页表项占用一个字节，则每个进程的页表就要占用1M的内存空间，而且还要求是连续的。 可以采用这样两个方法来解决这一问题：

① 采用离散分配方式来解决难以找到一块连续的大内存空间的问题：

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

4.5.5 反置页表

1. 反置页表的引入

为了减少页表占用的内存空间而引入了反置页表。它是为每个物理块设置一个页表项，并将它们按物理块的编号进行排序，其内容是页号和所隶属的进程标识符。

2. 地址变换

根据进程标识符和页号去检索反置页表，若检索成功则该页表项的序号就是该页所对应的物理块号；若检索失败则表示地址出错，若为请求调页方式，则表明该页尚未装入内存。

若为请求调页方式，每个进程还应有一个外部页表，在页表中包含了各个页在外存中的物理地址。

若反置页表很大时可以采用Hash算法检索。

4.6.1 分段存储管理方式的引入

分段思想符合编程理念，能满足用户和程序员的下述需求：

1. 方便编程

用户作业按逻辑关系划分为若干个段，段有自己的名字和长度，独立编址（从0开始）。
如指令　load 1,[A]|<D>；store 1,[B]|<C>

2. 信息共享

段是信息的逻辑单位，符合程序和数据的共享需求。分段存储管理与用户程序分段组织方式相适应。

3. 信息保护

信息保护同样是以信息的逻辑单位（段）为基础的。

4. 动态增长

实际应用中，某些段尤其是数据段会呈现动态增长的情况，相应的存储空间也会动态增加。

5. 动态链接

动态链接要求的是以目标模块（段）作为链接的基本单位。

4.6.2 分段系统的基本原理

2. 段表

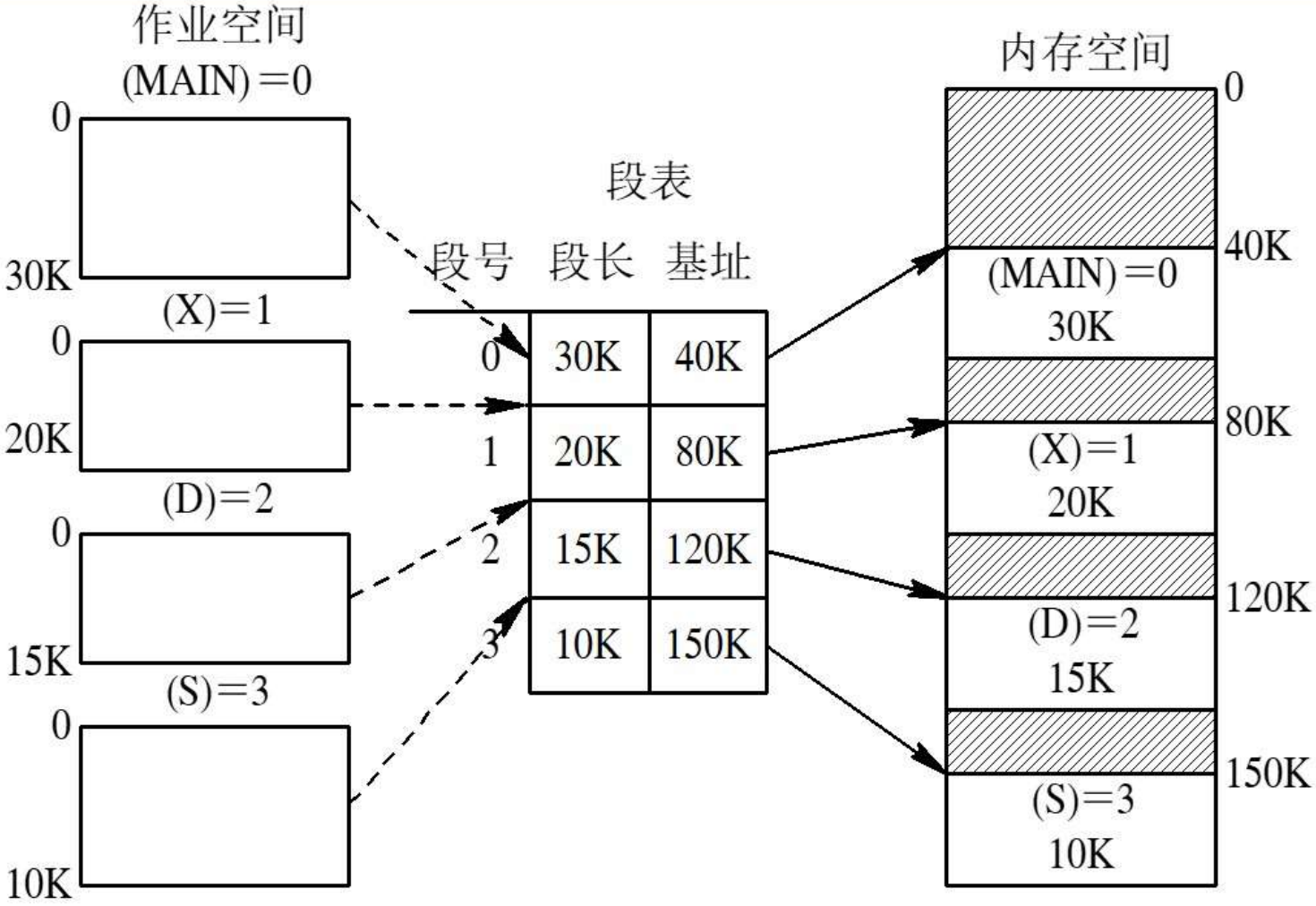


图 4-19 利用段表实现地址映射

4.6.2 分段系统的基本原理

4. 分页和分段的主要区别

(1) 页是信息的物理单位，段则是信息的逻辑单位。

分页是为实现离散分配方式，以消减内存的外零头， 提高内存的利用率。或者说， 分页仅仅是由于系统管理的需要而不是用户的需要。而段含有一组其意义相对完整的信息。 分段的目的是为了能更好地满足用户的需要。

(2) 页的大小固定且由系统决定，而段的长度却不固定，决定于用户所编写的程序。

分页是由系统把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而在系统中只能有一种大小的页面；而分段通常由编译程序在对源程序进行编译时，根据信息的性质来划分。

4.6.2 分段系统的基本原理

4. 分页和分段的主要区别

(1) 页是信息的物理单位，段则是信息的逻辑单位。

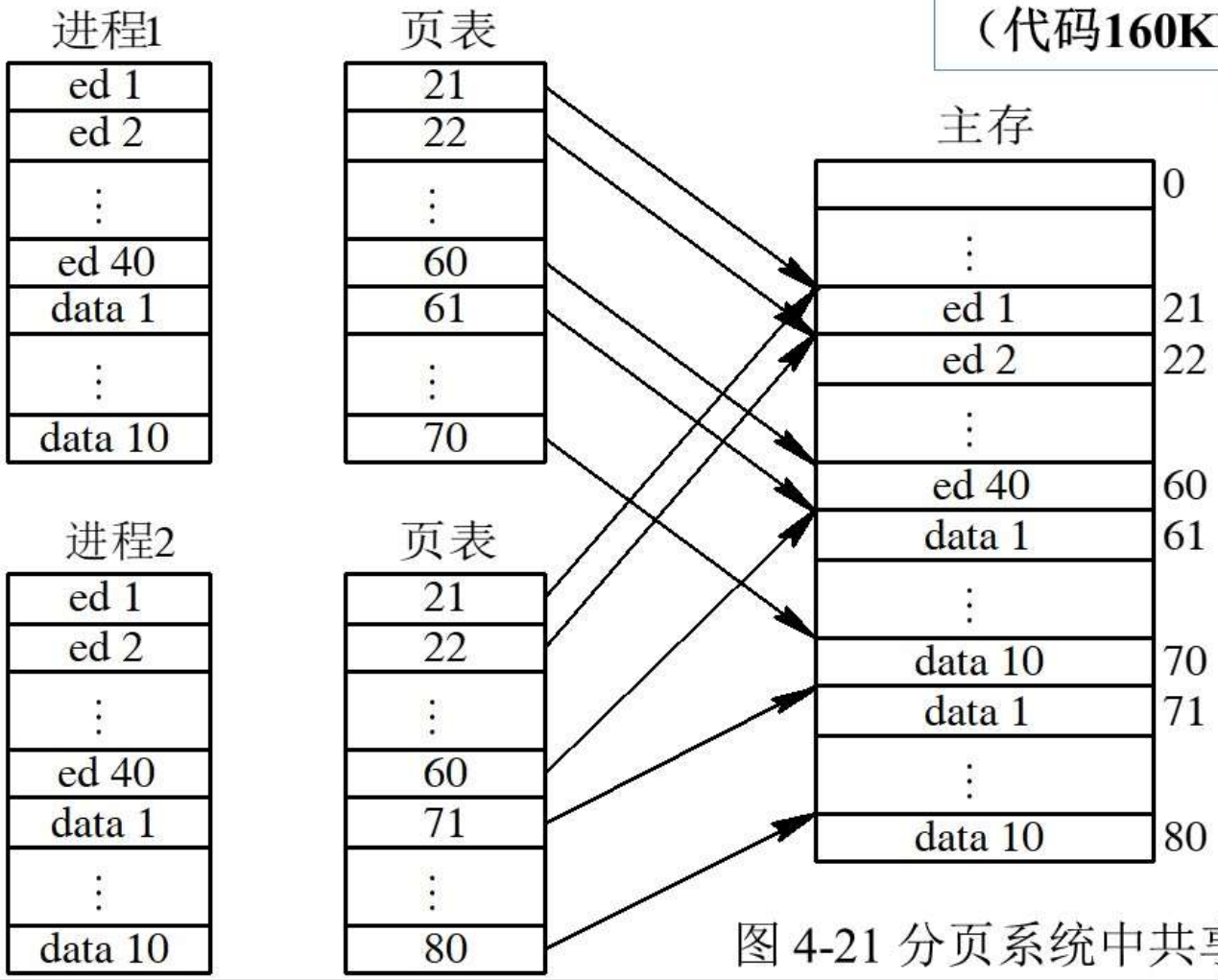
(2) 页的大小固定且由系统决定，而段的长度却不固定， 决定于用户所编写的程序。

(3) 分页的作业地址空间是一维的，而分段的作业地址空间则是二维的。

分页即单一的线性地址空间，程序员只需利用一个记忆符，即可表示一个地址； 而分段的作业地址空间则是二维的，程序员在标识一个地址时，既需给出段名，又需给出段内地址。

1. 分页系统中对程序和数据的共享

设有40个用户都运行编辑程序（代码160KB+数据区40KB）。



非共享需要空间
(160+40)*40=8M.

设代码可重用，则共享需要空间
160+40*40=1760。

共享代码要一致
数据区不同。

图 4-21 分页系统中共享editor的示意图

2. 分段系统中对程序和数据的共享

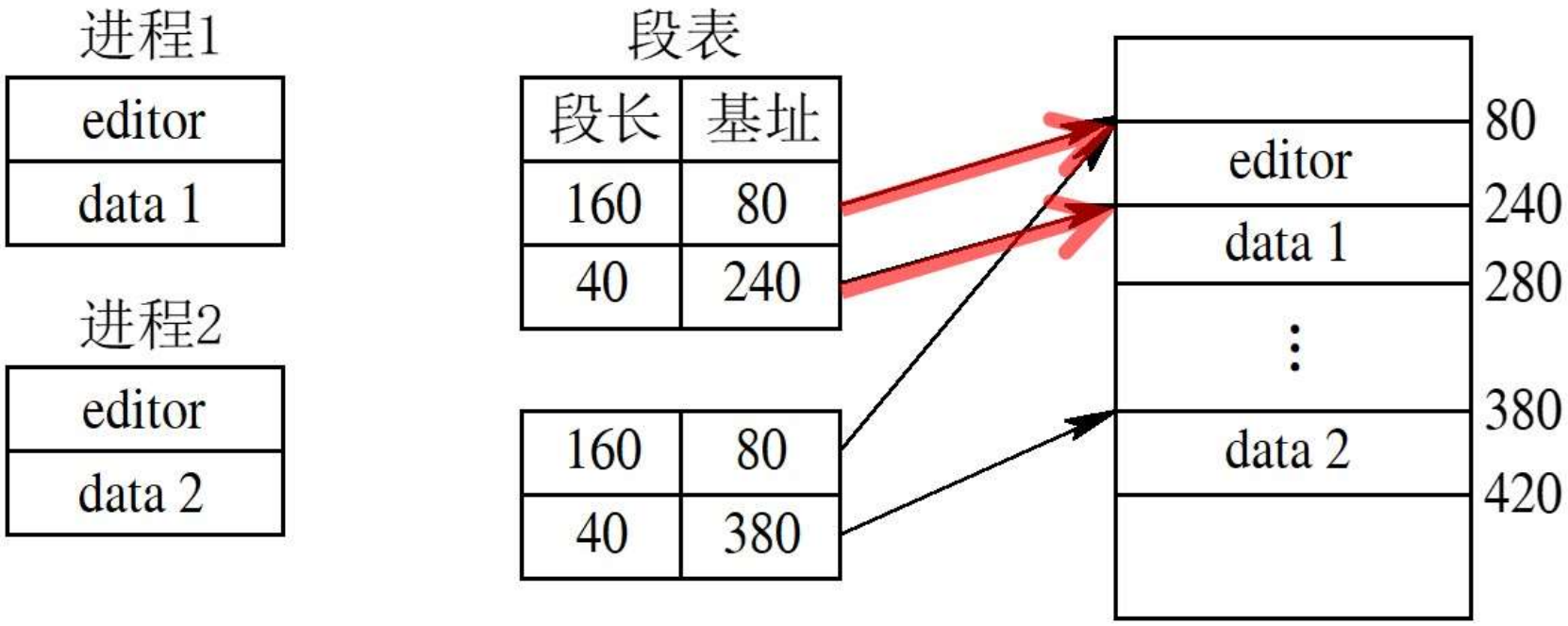
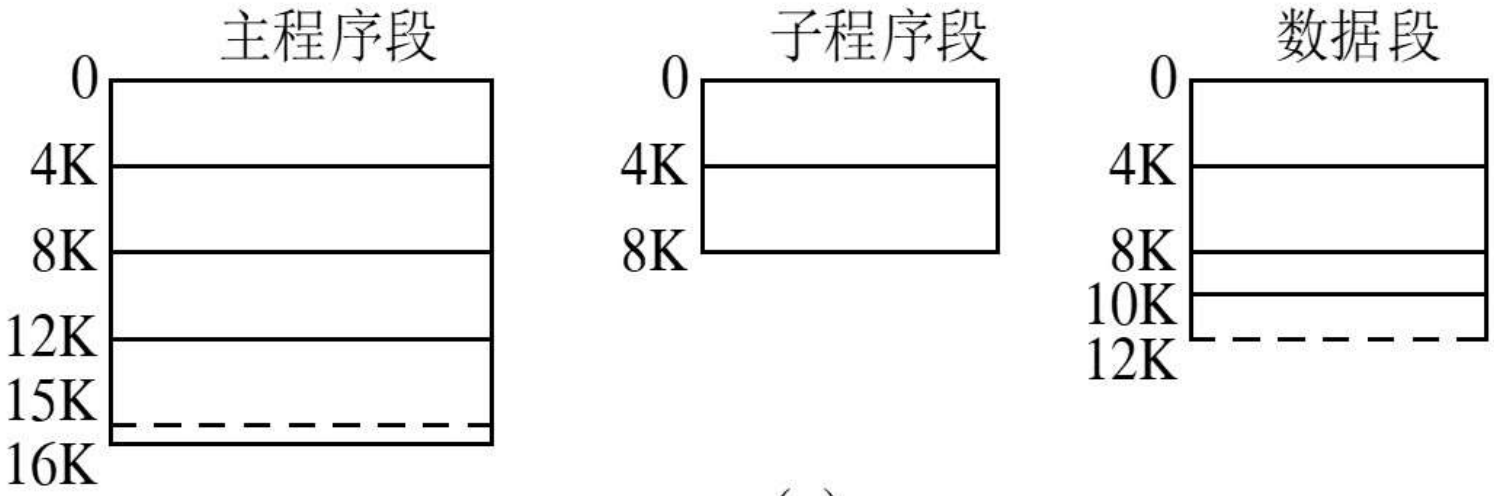


图 4-22 分段系统中共享editor的示意图

1. 基本原理



(*a*)

(*b*)

图 4-23 作业地址空间和地址结构

4.6.4 段页式存储管理方式

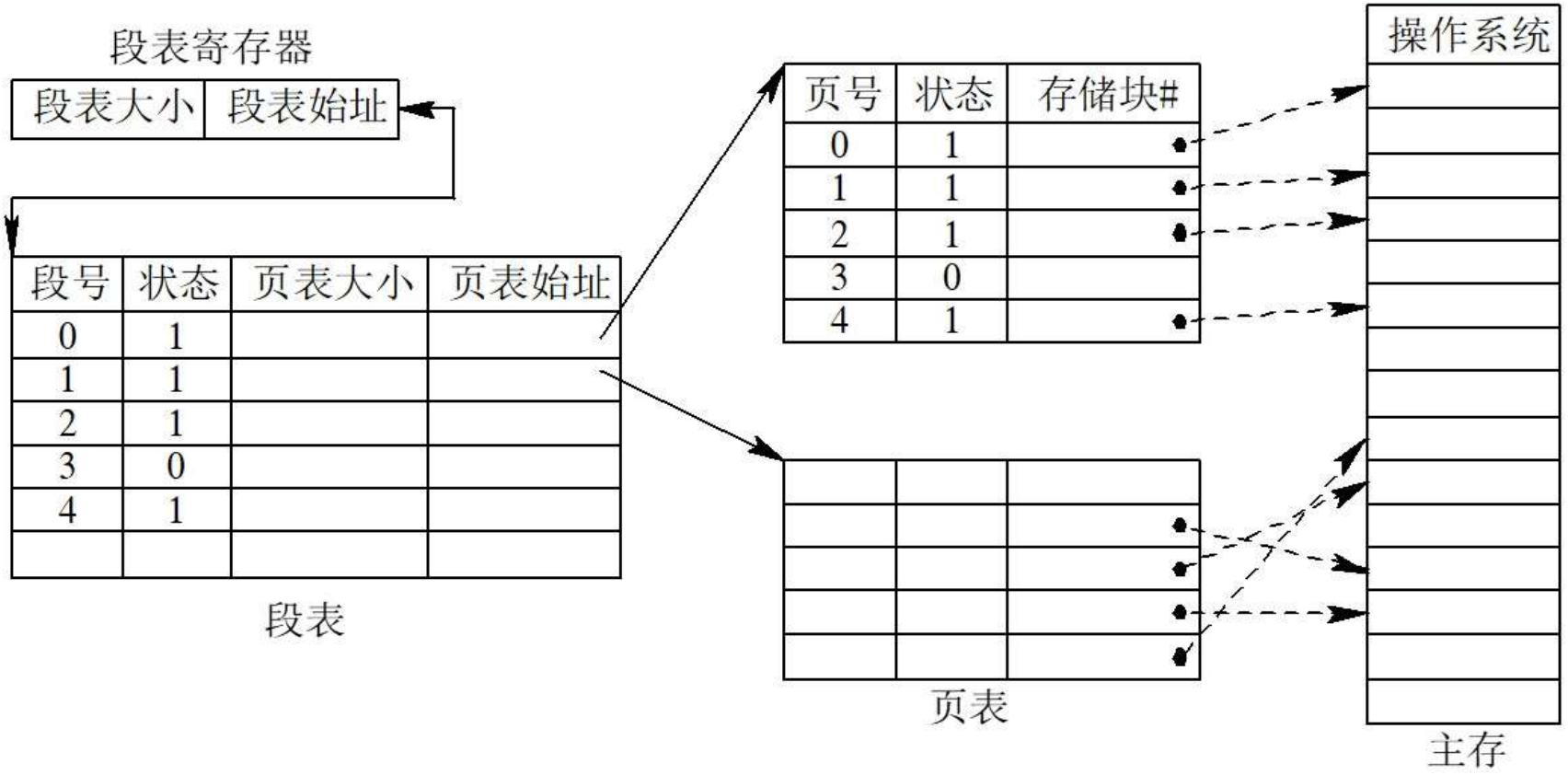


图 4-24 利用段表和页表实现地址映射

2. 地址变换过程

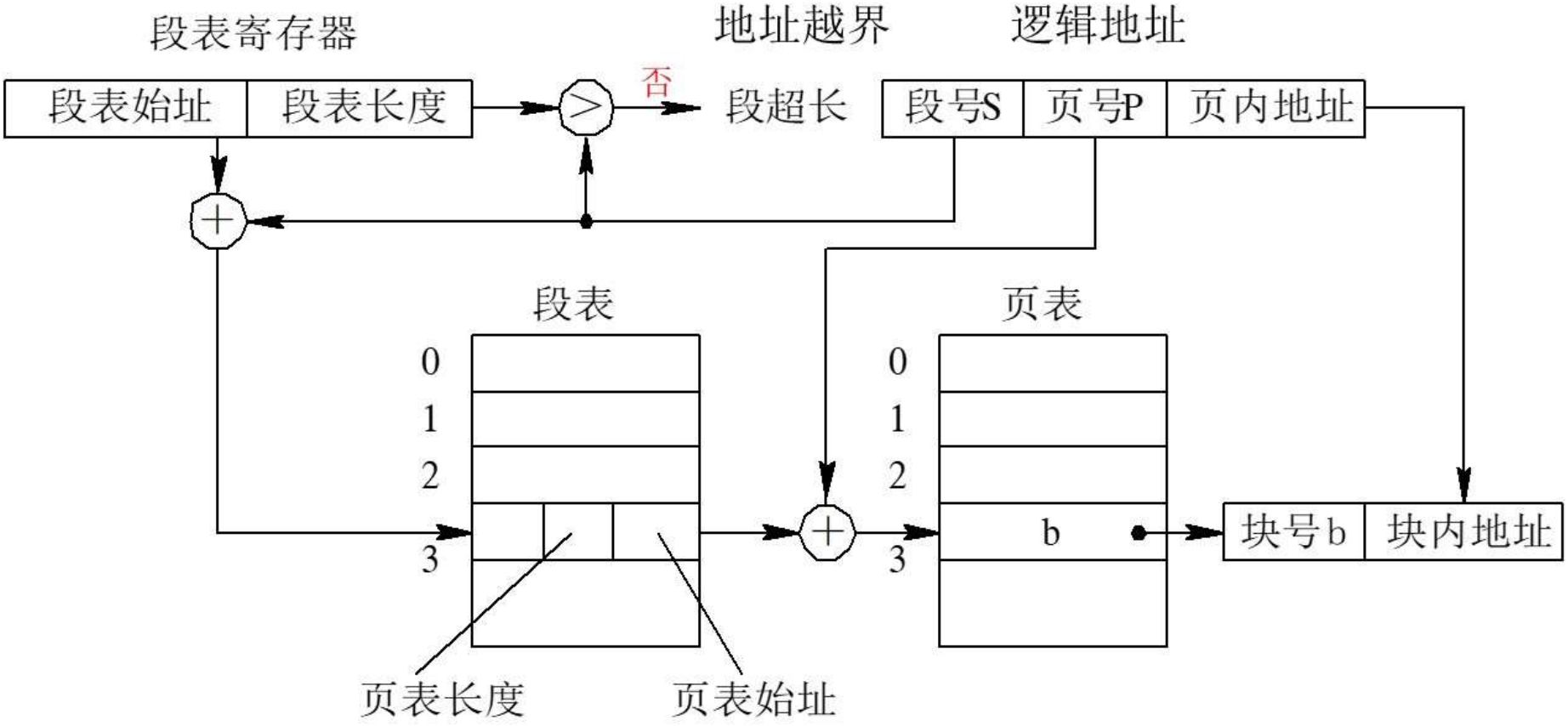


图 4-25 段页式系统中的地址变换机构