第四章 存储器管理

引言

4.1 程序的装入和链接

4.2 连续分配方式

4.3 基本分页存储管理方式

4.4 基本分段存储管理方式

存储组织

- 存储器的功能是保存数据，存储器的发展方向是高速、大容量和小体积。
  - 内存在访问速度方面的发展：DRAM、SDRAM、DDR、DRDRAM、DDR2、XDR、SRAM等；
  - 硬盘技术在大容量方面的发展：接口标准、存储密度等；

- 存储组织是指在存储技术和CPU寻址技术许可的范围内组织合理的存储结构。
  - 其依据是访问速度匹配关系、容量要求和价格。
  - “寄存器-内存-外存”结构
  - “寄存器-缓存-内存-外存”结构；

存储层次结构

- 快速缓存：SRAM
- 内存：DRAM,SDRAM,DDR,DRDRAM、 DDR2、XDR等；
- 外存：软盘、硬盘、光盘、磁带等；
- 微机中的存储层次组织：
  - 访问速度越慢，容量越大，价格越便宜；
  - 最佳状态应是各层次的存储器都处于均衡的繁忙状态；

存储管理的功能

- 存储分配和回收：分配和回收算法及相应的数据结构。
- 地址变换：
  - 可执行文件生成中的链接技术
  - 程序加载(装入)时的重定位技术
  - 进程运行时硬件和软件的地址变换技术和机构
- 存储共享和保护：
  - 代码和数据共享
  - 地址空间访问权限（读、写、执行）
- 存储器扩充：

重定位概念

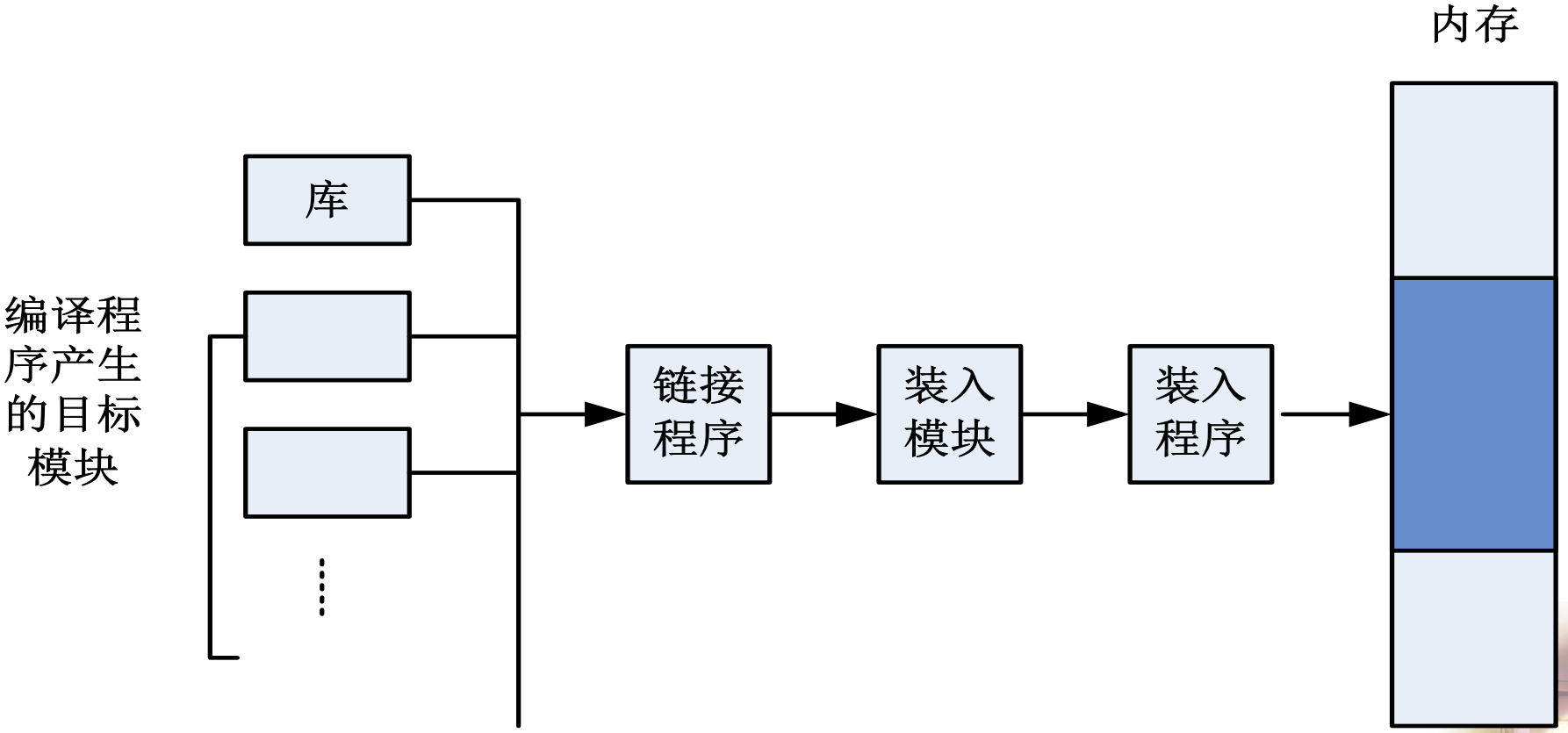
❖重定位：实现逻辑地址（相对地址）到物理地址（绝对地址）的映射。

❖逻辑地址：应用程序经编译后形成目标程序，再经过链接后形成可装入程序，这些程序的地址都是从0开始，程序中的其他地址都是相对于起始地址计算的，这些地址为相对地址。

❖物理地址：主存中一系列存储信息的物理单元的地址。

4.1 程序的装入和链接

- 编辑----编译----链接----装入----运行



4.1.1 程序的装入

1、绝对装入：

* 编译后，装入前已产生了绝对地址（内存地址），装入时不再作地址重定位。
* 绝对地址的产生：（1）由编译器完成，（2）由程序员编程完成。
* 对（1）而言，编程用符号地址。

2、可重定位装入；

* 静态重定位：地址转换在装入时一次完成，由软件实现（重定位装入程序完成）。

  缺点：不允许程序运行时在内存中移动位置。

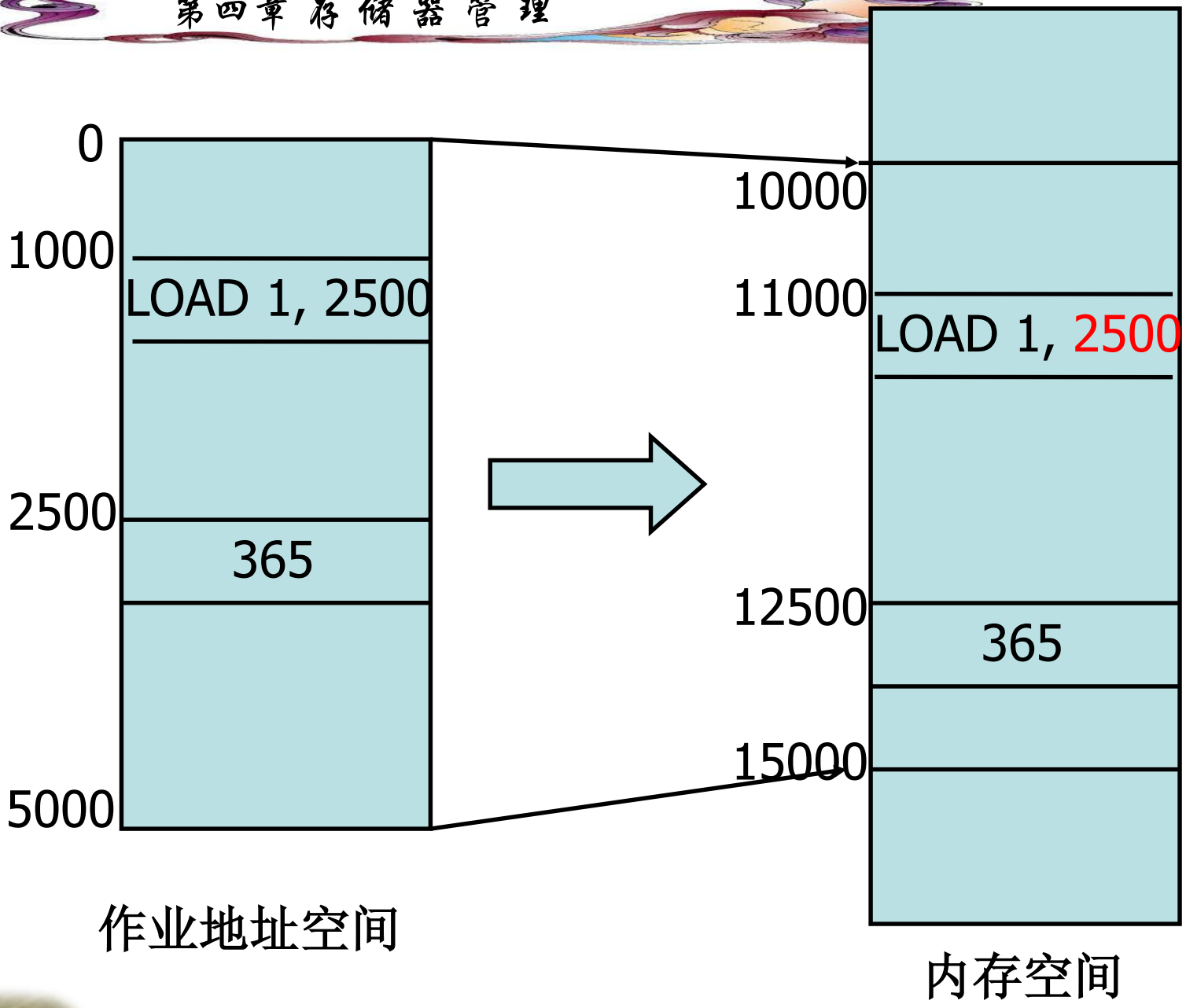

0
1000
LOAD 1, 2500
2500
365
5000
作业地址空间
10000
11000
LOAD 1, 2500
12500
365
15000
内存空间

图4-3

3. 动态运行时装入

* 在装入后不能移动，
* 该情况一般在执行时才完成相对地址向绝对地址的转换，需要硬件地址变换“重定位寄存器”的支持，才能保证进程的可移动性。

4.1.2 程序的链接

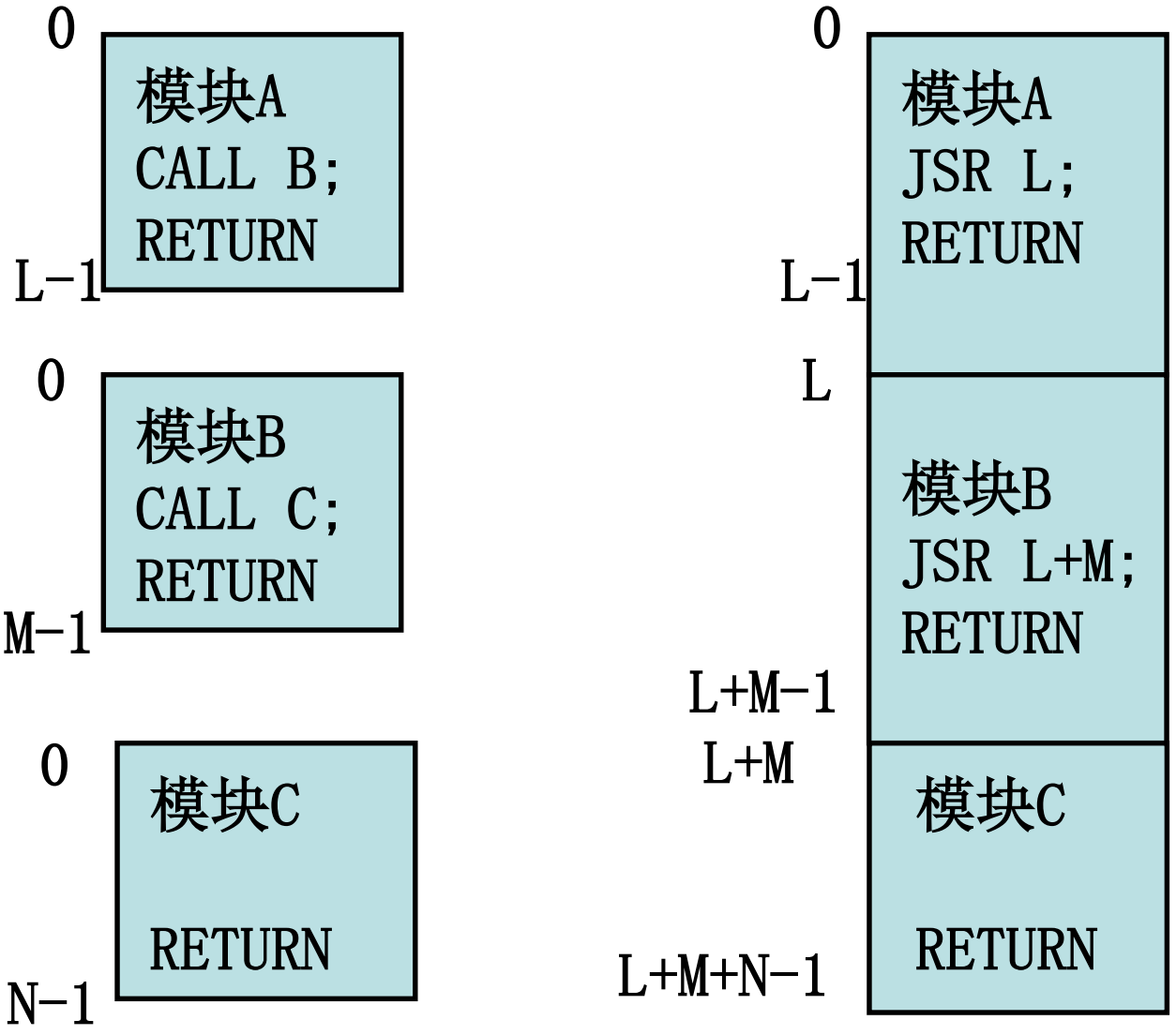
1、静态链接

* a. 对相对地址的修改
* b. 变换外部调用符号

2、装入时动态链接

* a. 便于修改和更新
* b. 便于实现对目标模块的共享

3、运行时动态链接


0
模块A
CALL B;
RETURN
L-1
0
模块B
CALL C;
RETURN
M-1
0
模块C
RETURN
N-1
0
模块A
JSR L;
RETURN
L-1
L
模块B
JSR L+M;
RETURN
L+M-1
L+M
模块C
RETURN
L+M+N-1

(a)目标模块　　(b)装入模块

4.2连续分配方式

- 单一连续分配
  - 用于单用户，单任务中
- 分区式连续分配
  - 固定式
  - 可变式
  - 可重定位分区分配

4.2.1 单一连续分区

- 内存分为两个区域：系统区，用户区。应用程序装入到用户区，可使用用户区全部空间。
- 最简单，适用于单用户、单任务的OS。
- 优点：易于管理。
- 缺点：对要求内存空间少的程序，造成内存浪费；程序全部装入，很少使用的程序部分也占用内存。

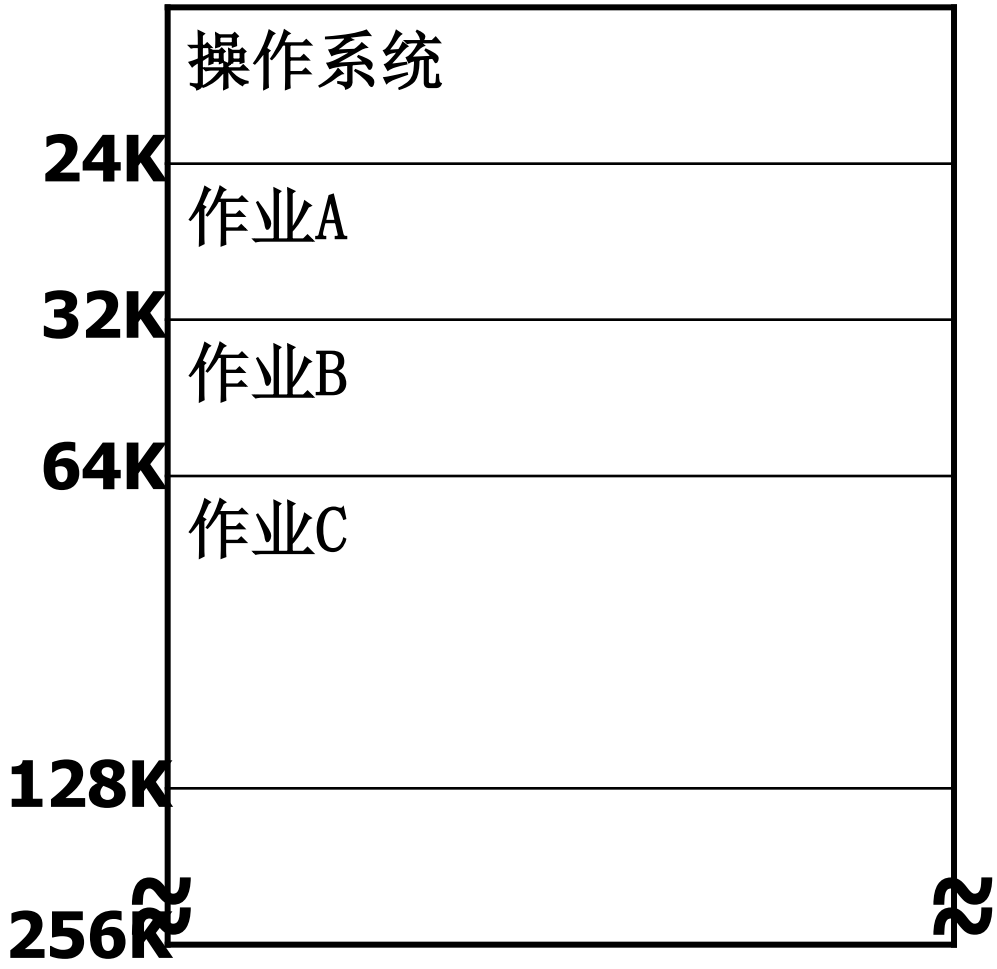
4.2.2 固定分区

- 基本思想：将内存划分成若干个连续区域，称为分区。每个分区只能存储一个程序，而且程序也只能在它所驻留的分区中运行。有n个分区，则可同时装入n个作业/任务。
- 一、分区大小：
  - 相等：
  - 不相等：不相等利用率更高。
- 二、内存分配：
  - 数据结构
    - 将分区按大小排序，并将其地址、分配标识作记录
  - 例：dos的MCB
- 三、优缺点：
  - 优点：简单；缺点：有碎片（内零头）。

分区说明表

分区号	大小（K)	起址（K）	状态
1	12	20	已分配
2	32	32	已分配
3	64	64	已分配
4	128	128	已分配

分配情况

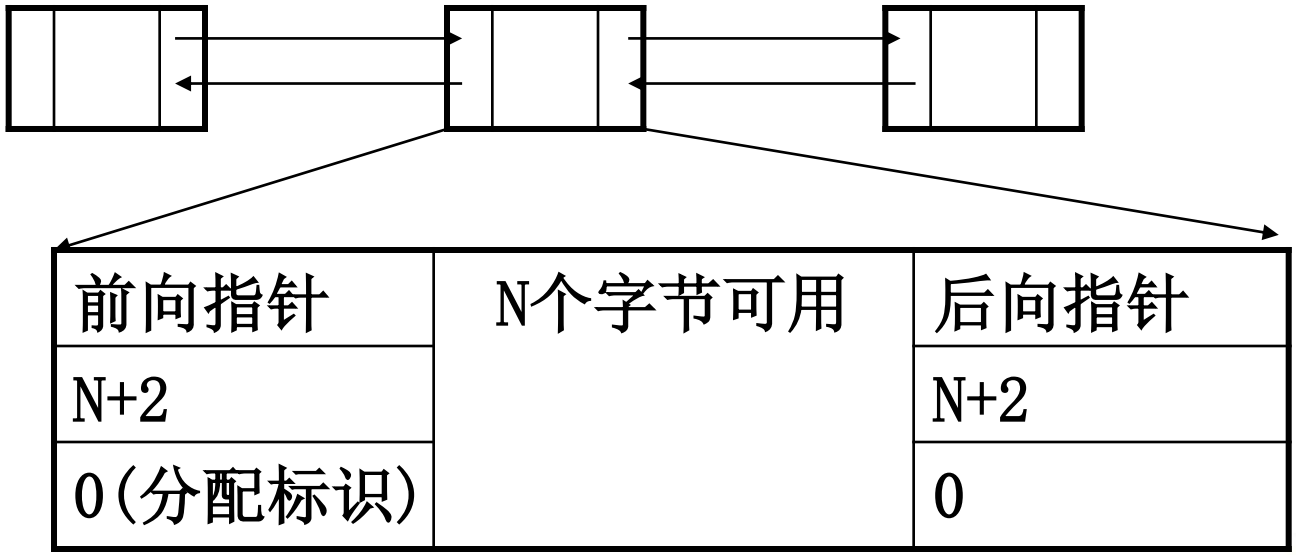


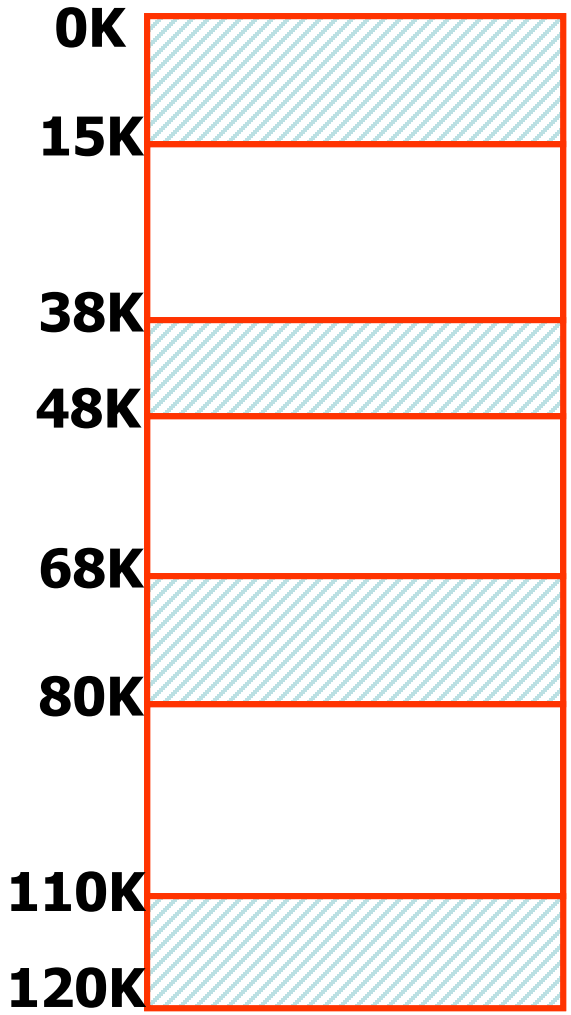
4.2.3 可变式分区

基本思想：内存不预先划分好，当作业装入时，根据作业的需求和内存空间的使用情况决定是否分配。若有足够的空间，则按需分割一部分分区给该进程。

一、数据结构

* 1．空闲分区表
* 2．空闲分区链





分区分配表

空闲分区表：

区号	大小	始址地址
1	23K	15K
2	20K	48K
3	30K	80K
		空
		空

二、分配算法

* 1. 首次适应算法。
  - 要求：分区按低址---高址链接
  - 特点：找到第一个大小满足的分区，划分。有外零头，低址内存使用频繁。
* 2. 循环首次适应算法。
  - 从1中上次找到的空闲分区的下一个开始查找。
  - 特点：空闲分区分布均匀，提高了查找速度；缺乏大的空闲分区。
* 3. 最佳适应算法。（和最坏适应算法）
  - 分区按大小递增排序；分区释放时需插入到适当位置。

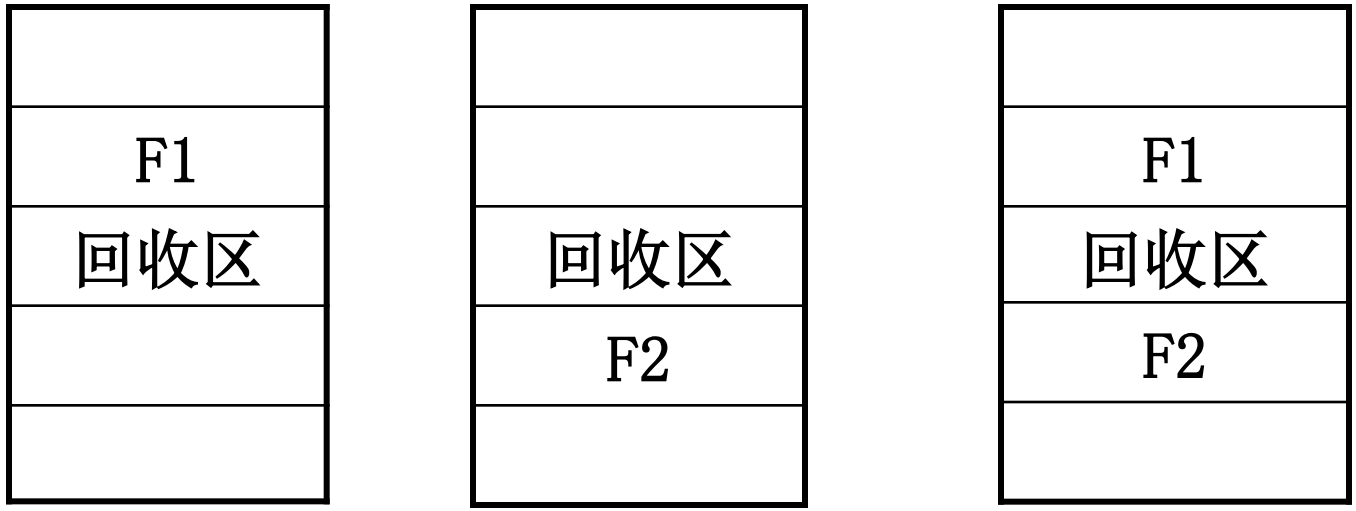
总结：没有完美的算法，只有最合适的算法，根据算法的特点和用户的需要选择。

*回收：（1）上邻空闲区：合并，改大小

（2）下邻空闲区：合并，改大小，首址。

（3）上、下邻空闲区：合并，改大小。

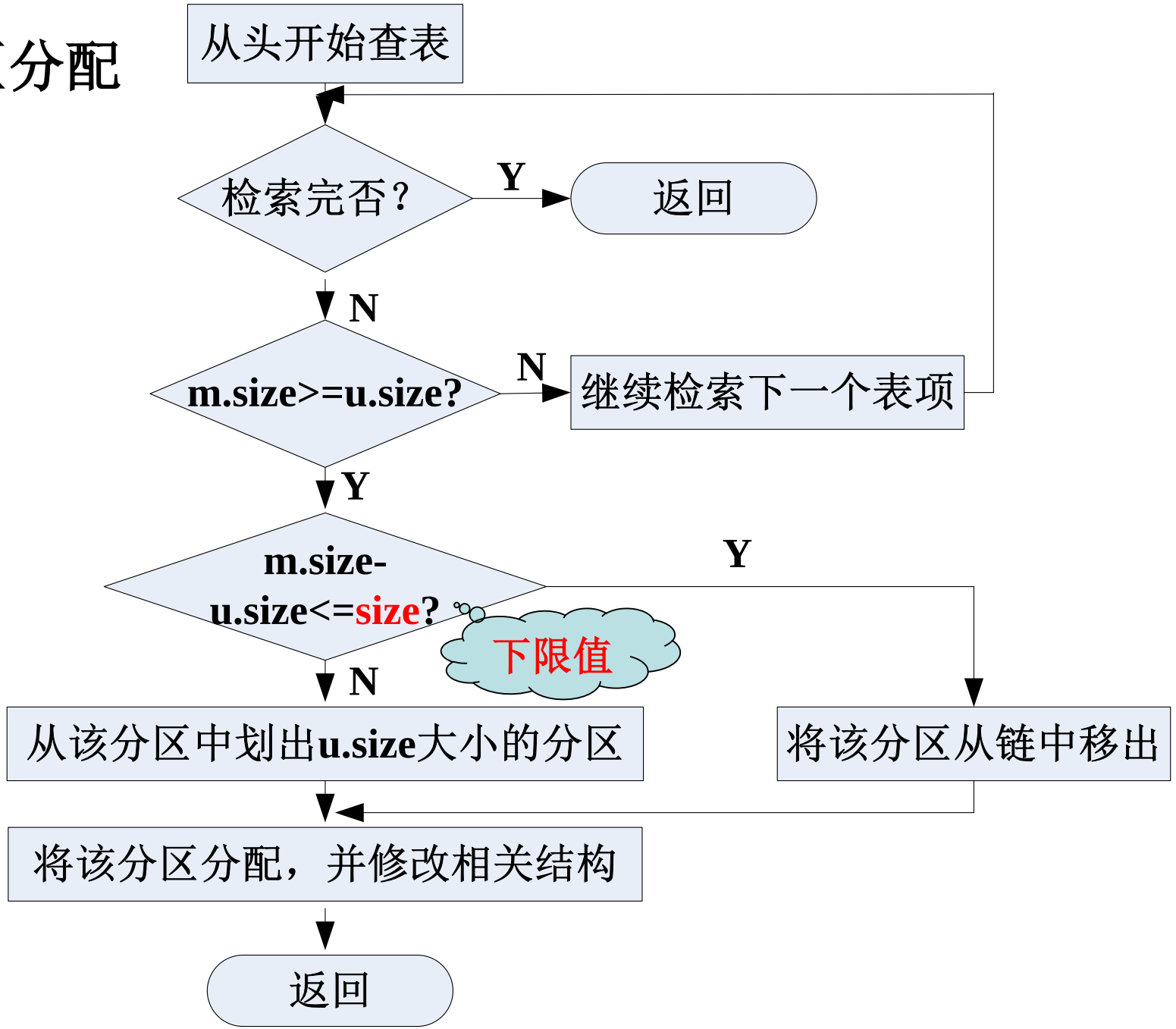
（4）不邻接，则建立一新表项。



4-9 内存回收时的情况

三、分区分配

分配算法


从头开始查表
检索完否？
Y
返回
N
m.size>=u.size?
N
继续检索下一个表项
Y
m.size-
u.size<=size?
Y
下限值
N
从该分区中划出u.size大小的分区
将该分区从链中移出
将该分区分配，并修改相关结构
返回

例：在计算机系统中,按地址排列的内存中的空闲区大小是:10K,4K,20K,18K,7K,9K,12K,15K,对于连续的段请求:12K,10K,9K.使用循环适应算法和最佳适应算法将找出哪些空闲区?

解：循环适应算法：20K,18K,9K

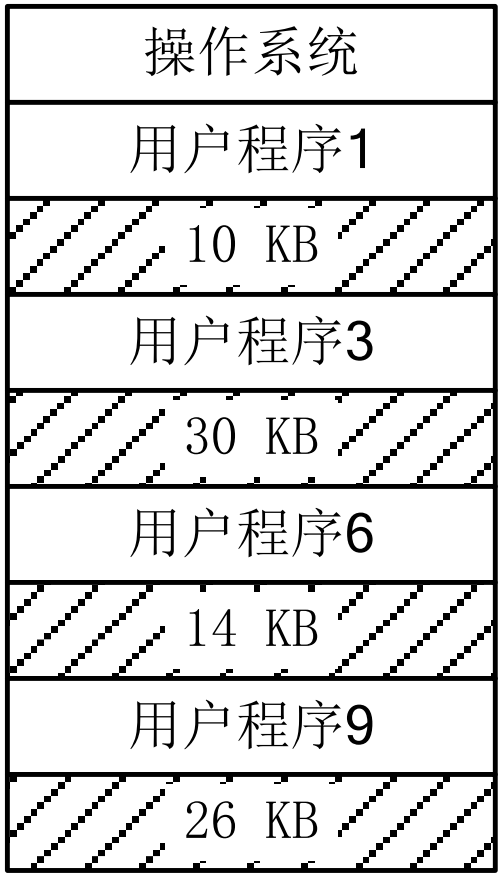
最佳适应算法：12K,10K,9K

4.2.4 可重定位分区分配

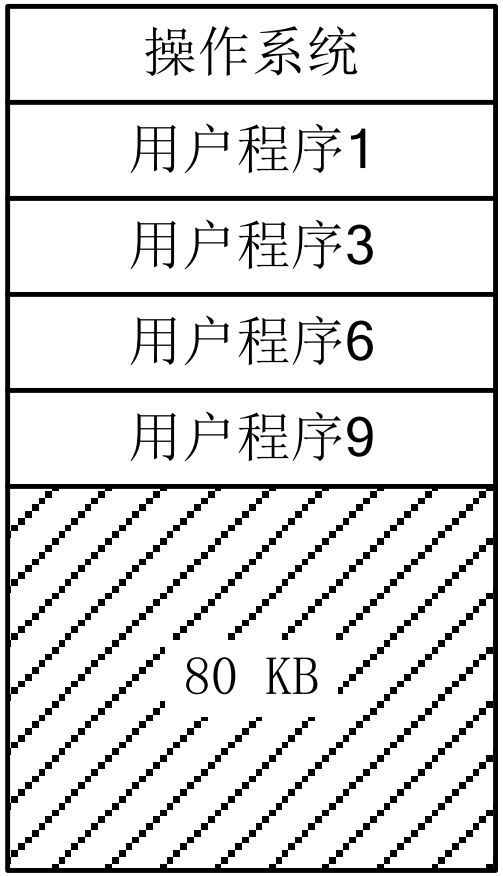
1. 动态重定位的引入

* 连续式分配中，总量大于作业大小的多个小分区不能容纳作业。
* 紧凑
  - 通过作业移动将原来分散的小分区拼接成一个大分区。
  - 作业的移动需重定位。是动态（因作业已经装入）

紧凑


操作系统
用户程序1
10 KB
用户程序3
30 KB
用户程序6
14 KB
用户程序9
26 KB

(a) 紧凑前


操作系统
用户程序1
用户程序3
用户程序6
用户程序9
80 KB

(b) 紧凑后

2、动态重定位的实现

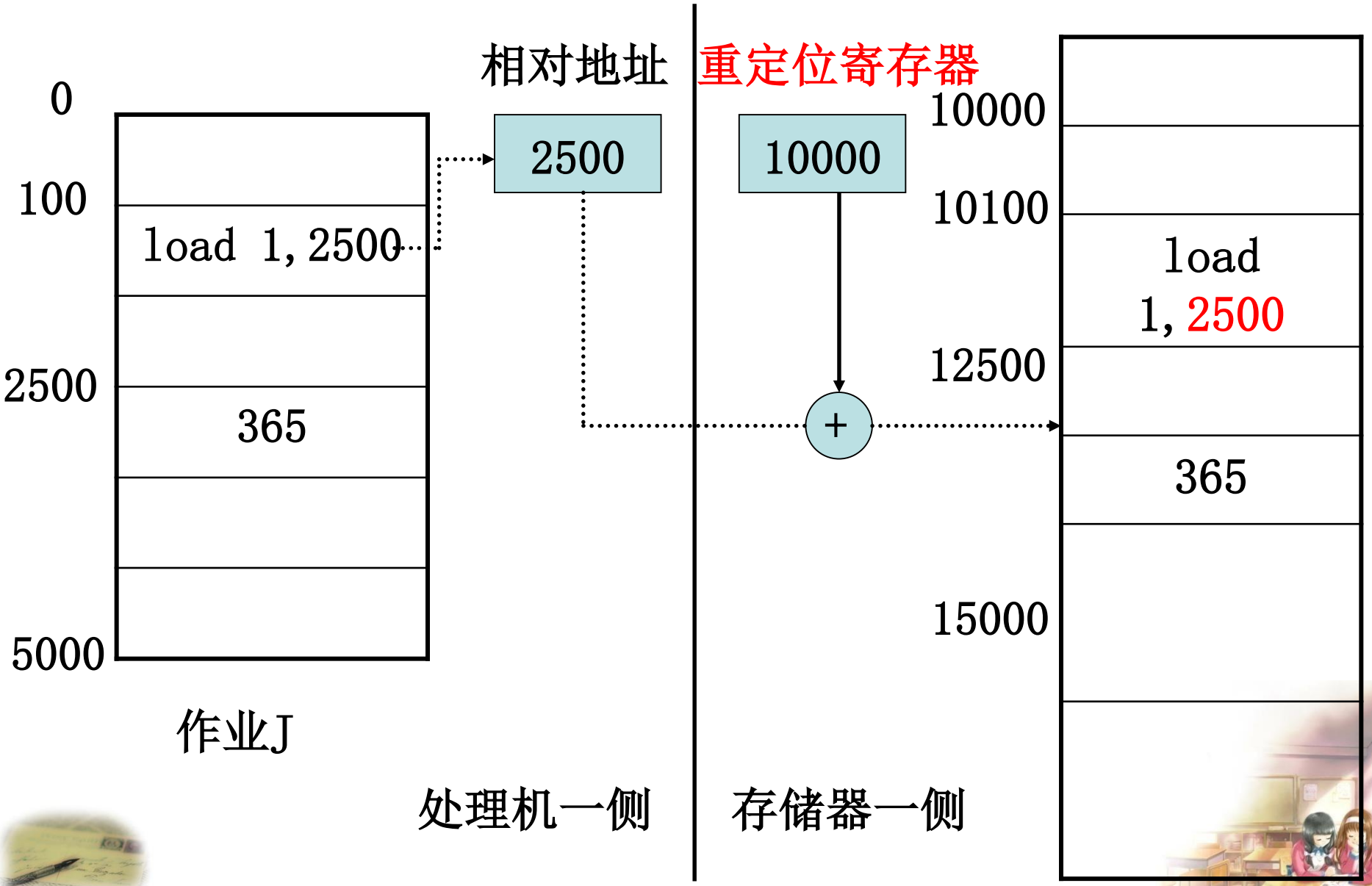
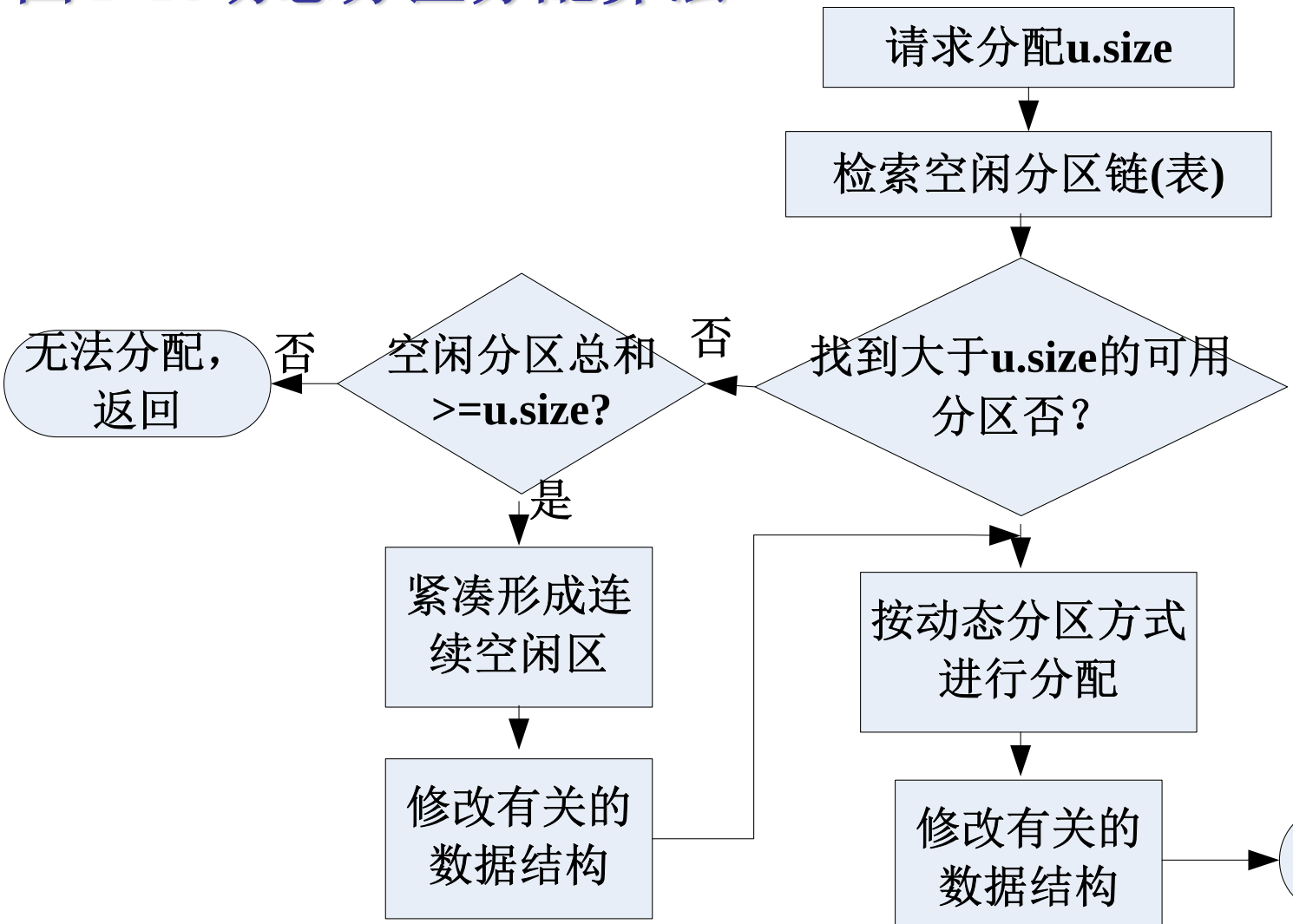

相对地址
重定位寄存器
0
100
2500
5000
load 1, 2500
365
2500
10000
+
10000
10100
12500
15000
load
1, 2500
365
作业J
处理机一侧
存储器一侧

图4-13动态分区分配算法



4.2.5 对换

1 对换的引入

* 将阻塞进程，暂时不用的程序，数据换出。
* 将具备运行条件的进程换入。
* 类型：
  - 整体对换：进程对换，解决内存紧张
  - 部分对换：页面对换/分段对换：提供虚存支持

2 对换空间的管理

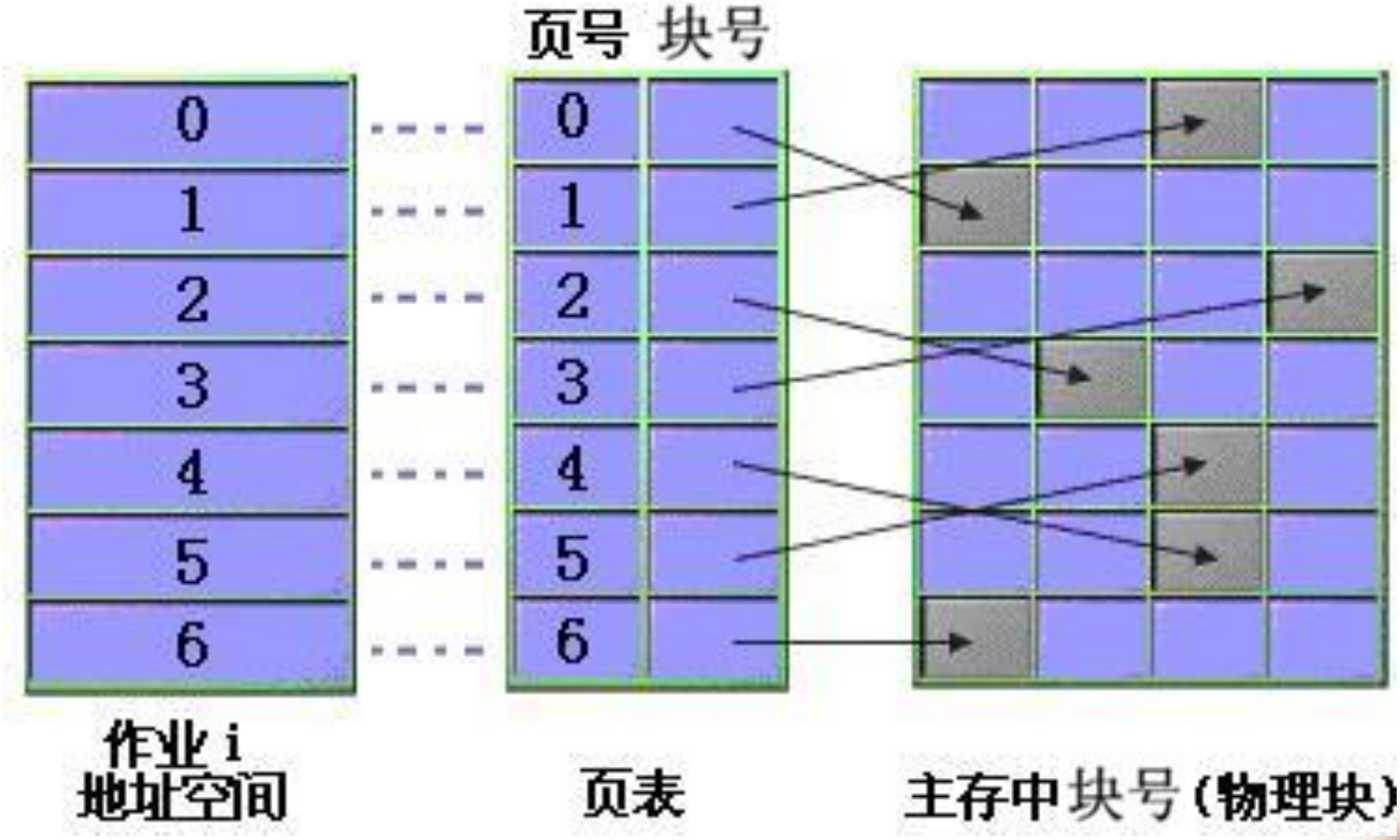
* 外存
  - 对换区比文件区侧重于对换速度。
* 因此，对换区一般采用连续分配。采用数据结构和分配回收类似于可变化分区分配。

3 换出与换入

* 换出
  - 1．选出被换出进程：
    因素：优先级，驻留时间，进程状态
  - 2．换出过程：
    对于共享段：计数减1， 是0则换出，否则不换
    修改PCB和MCB（或内存分配表）
* 换入：
  - 1．选择换入进程：优先级，换出时间等。
  - 2．申请内存。
  - 3．换入

4.3基本分页存储管理

- 连续分配引起:碎片
- 碎片问题的解决：紧凑方式消耗系统开销。
- 离散分配
  - 分页、分段、段页


页号 块号
0
1
2
3
4
5
6
0
1
2
3
4
5
6
作业 i
地址空间
页表
主存中块号(物理块)

4.3.1页面与页表

- 1.页面
  - 页面和物理块：逻辑空间和内存空间
  - 页面大小
    - 页太大，页内碎片大。
    - 页太小：页表可能很长，换入/出效率低
- 2.地址结构
  - 31 12 11 0

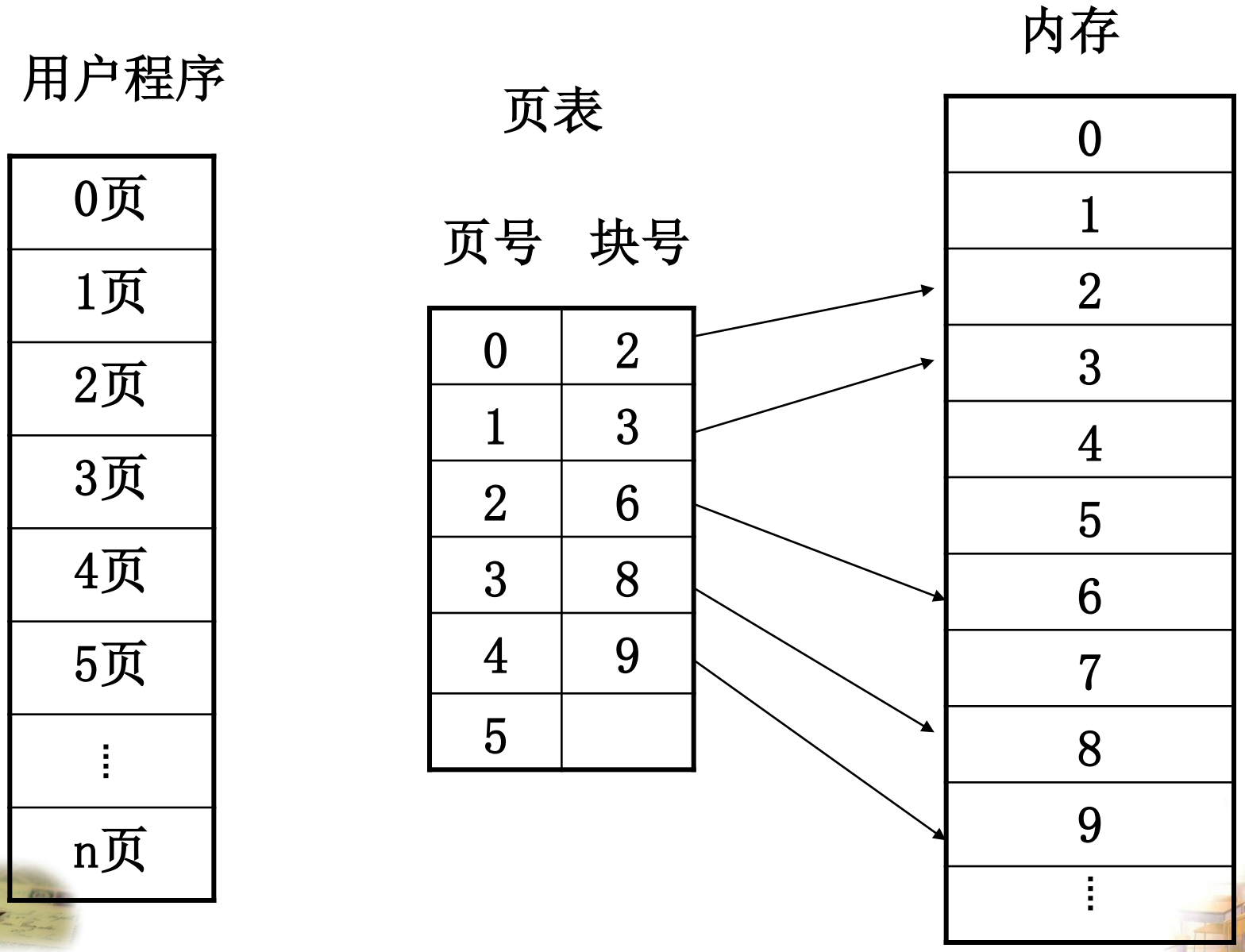
页号P	位移W

  - 逻辑地址A；页大小L；页内偏移d

$$P = INT\left[\frac{A}{L}\right]$$

$$d = [A]MODL$$

页表



例：L=1000B，则第0页对应0-999，第1页对应1000-1999。

设A=3456，则P=INT[3456/1000]=3，d=[3456] mod 1000=456

故 A=3456→（3，456）

一般来说，页面尺寸应该是2的幂。这样的优点是可以省去除法，由硬件自动把地址场中的数拆成两部分来决定对应的页号和页内地址。

例：页的大小为1KB，则逻辑地址4101的页号、页内地址可这样定：

1K=1024=2^{10} (页内地址位数为10)

4101=$2^{12}+2^{2}+2^{0}$ ，逻辑地址字如下：

000100 0000000101

页号 页内地址

故 A=4101→（4，5）

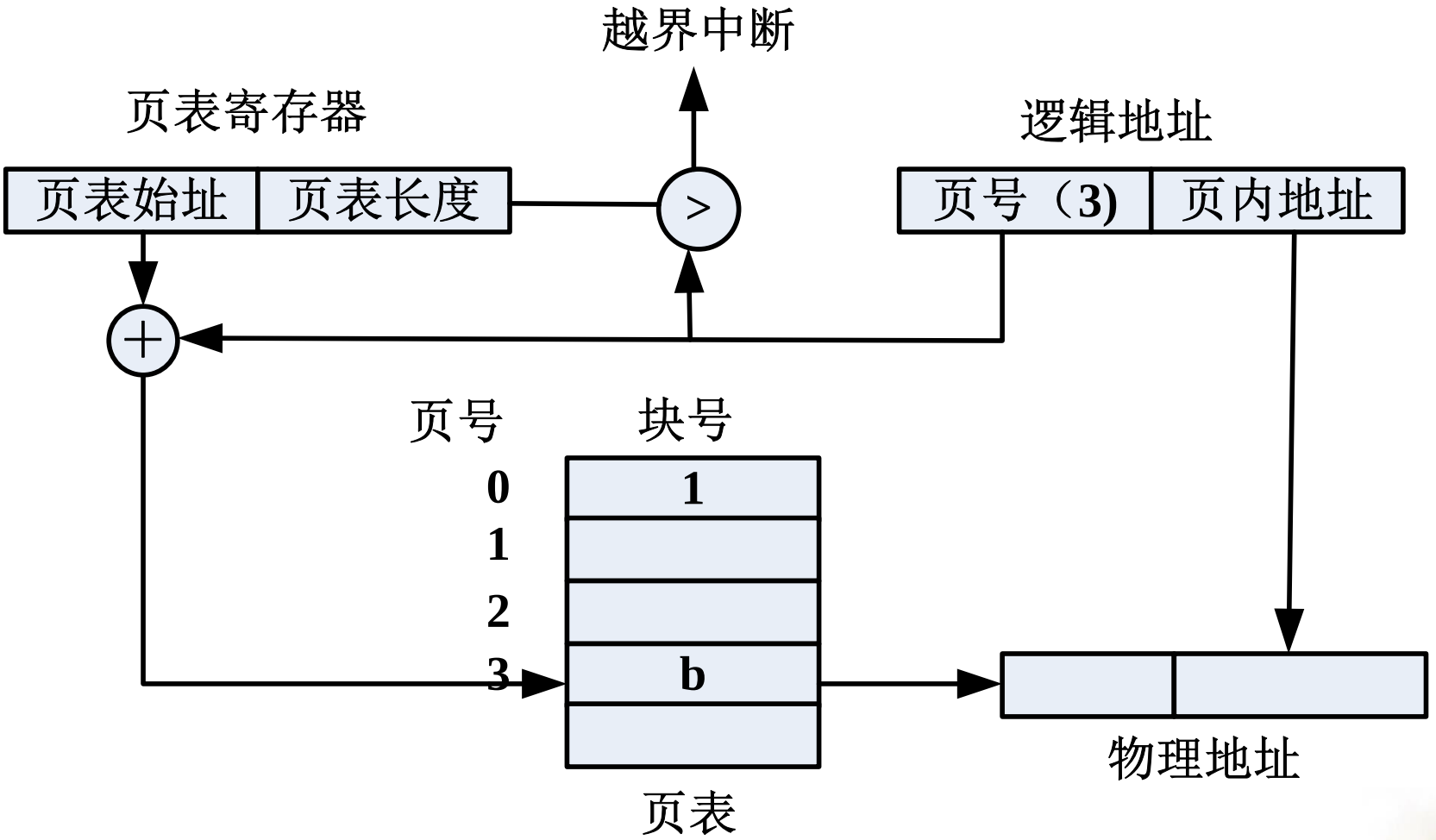
4.2 地址变换机构

- 基本任务：逻辑地址——物理地址的映射。

  页号→块号　　通过页表来完成

  页内地址→块内地址　　无需转换

- 一、基本地址变换机构：
  - 越界保护
  - 每个进程对应一页表，其信息（如长度、始址）放在PCB中，执行时将其首地址装入页表寄存器。



分页系统地址变换机构

需要考虑的问题：

❖页表放在哪里？整个系统的页表空间有多大？

❖直接映像的分页系统对系统效能的不利影响？（影响执行速度，因为CPU至少要访问两次主存才能存取到所要数据）

❖基本的地址变换机构

①页表驻留在内存中。

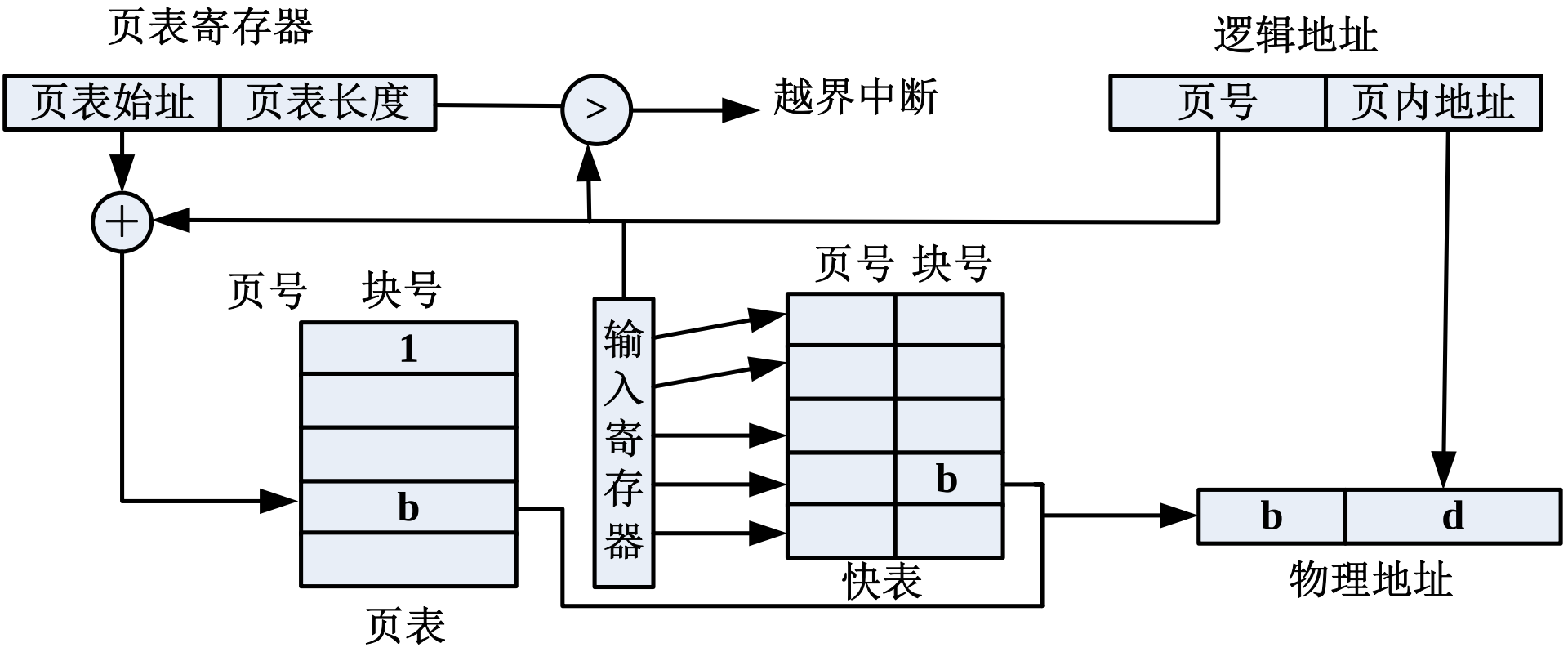
②系统中设置一个页表寄存器存放页表在内存中的始址和页表的长度。

③缺点：两次访问主存，速度降低近1/2

2. 具有快表的地址变换机构

- 不具快表，则需两次访问内存。
  - （1）访页表
  - （2）得到绝对地址内容
- 有快表，速度提高。
- 快表贵，不能太多。

2. 具有快表的地址变换机构



具有快表的地址变换机构

例：有一页式系统，其页表存放在主存中：

①如果对主存的一次存取需要1.5 μs，试问实现一次页面访问的存取时间是多少？

②如果系统加有快表，平均命中率为85%，当页表项在快表中时，其查找时间忽略为0， 试问此时的存取时间是多少？

答：若页表存放在主存中，则要实现一次页面访问需两次访问主存：一次是访问页表，确定所存取页面的物理地址（称为定位）。第二次才根据该地址存取页面数据。

■页表在主存的存取访问时间

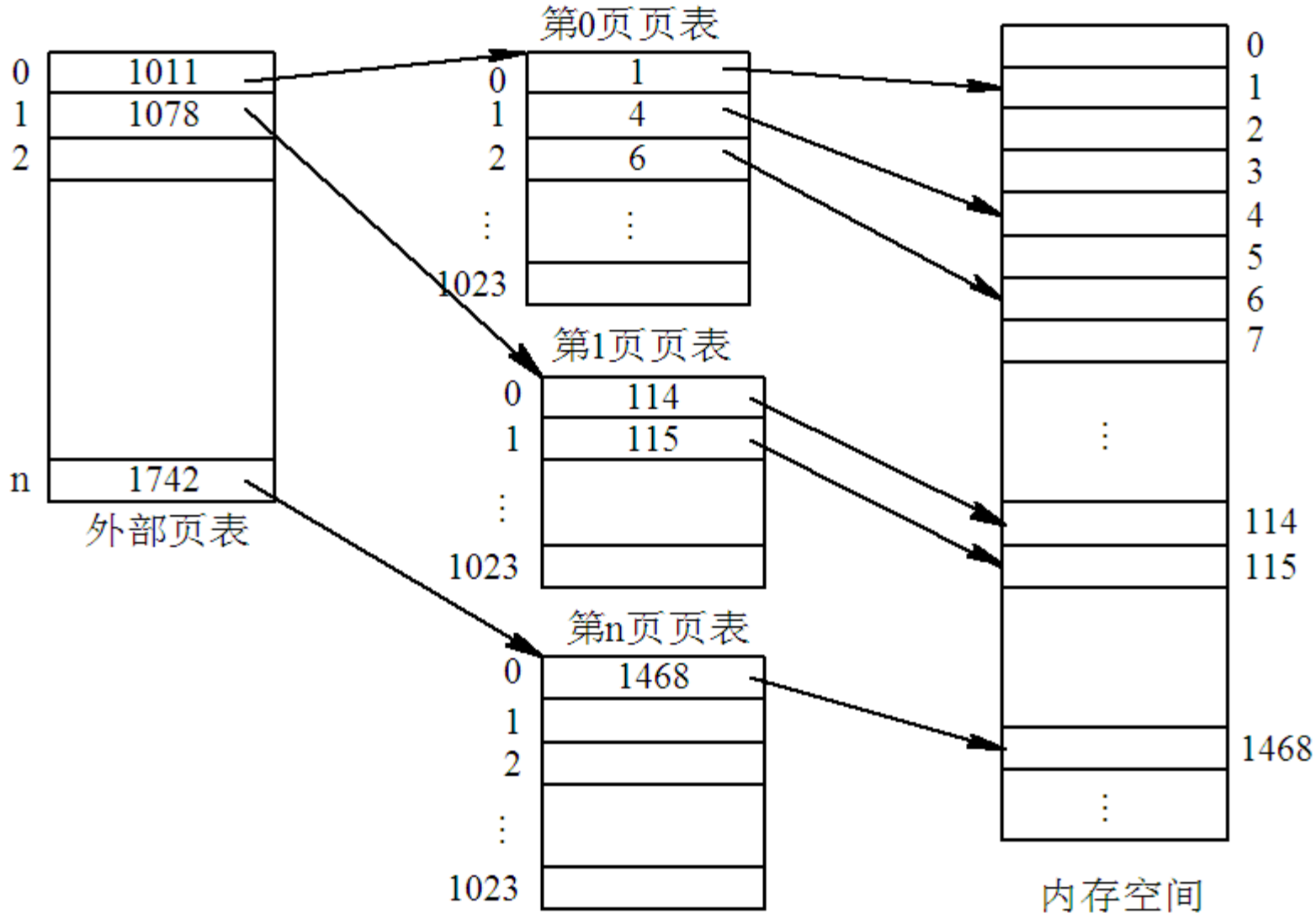
=1.5*2=3(μs)

■增加快表后的存取访问时间

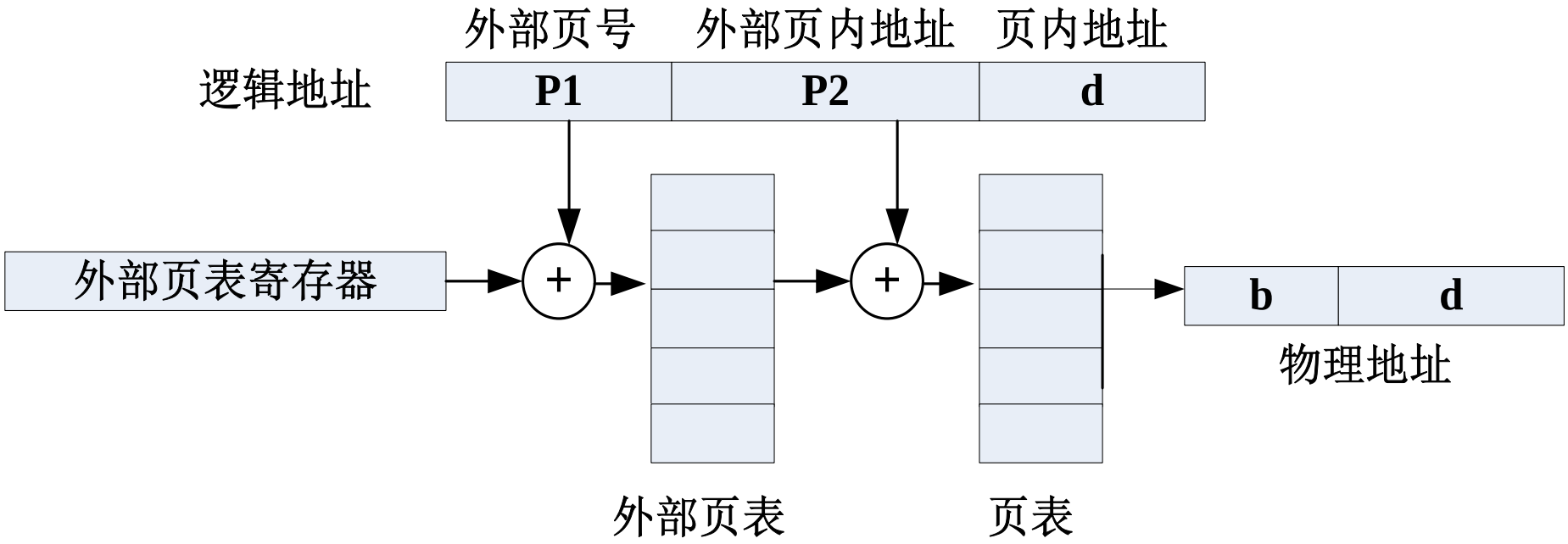
=0.85*1.5+(1-0.85)*2*1.5=1.725(μs)

4.3.3 两级和多级页表

- 页表可能很大，将其离散存放在不同页块中。
- 建一“外部页表”来管理这些离散页表块。
  - 相当于单级页表中的页表寄存器，一般应常驻内存。每项记录页表始址，且增加存在位。
- 64位机器页表一般>3级，最外层页表常驻。


第0页页表
第1页页表
第n页页表
外部页表
内存空间
0
1
2
n
1011
1078
1742
1
4
6
114
115
1468
1023

两级页表示意图



具有两级页表的地址变换机构

练习：

1.某系统采用页式存储管理策略，拥有逻辑空间32页，每页2K，拥有物理空间1M。

（1）写出逻辑地址的格式。

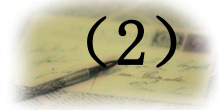
（2）若不考虑访问权限等，进程的页表有多少项？每项至少有多少位？

（3）如果物理空间减少一半，页表结构应相应作怎样的改变？

2. 已知某分页系统，主存容量为64K，页面大小为1K，对一个4页大的作业，其0、1、2、3页分别被分配到主存的2、4、6、7块中。

（1）将十进制的逻辑地址1023、2500、3500、4500转换成物理地址。

（2）以十进制的逻辑地址1023为例画出地址变换过程图。



1.

(1)系统拥有逻辑地址空间32页，则逻辑地址中页号需用5位描述；每页2K，则页内地址用11位描述。

（2）进程页表项数为32，另外页表项中只给出页所对应的物理块号，1M的物理空间可分为2^9个内存块，故每个页表项至少有9位。

（3）如果物理空间减少一半，则页表中页表项数不变，每项的长度可减少1位。

2.(1)

逻辑地址1023：1023/1024，得页号0，页内地址1023，查页表的相应块号2，故物理地址为2*1024+1023=3071

逻辑地址2500： 2500/1024，得页号2，页内地址452，查页表的相应块号6，故物理地址为6*1024+452=6596

逻辑地址3500： 3500/1024，得页号3，页内地址428，查页表的相应块号7，故物理地址为7*1024+428=7596

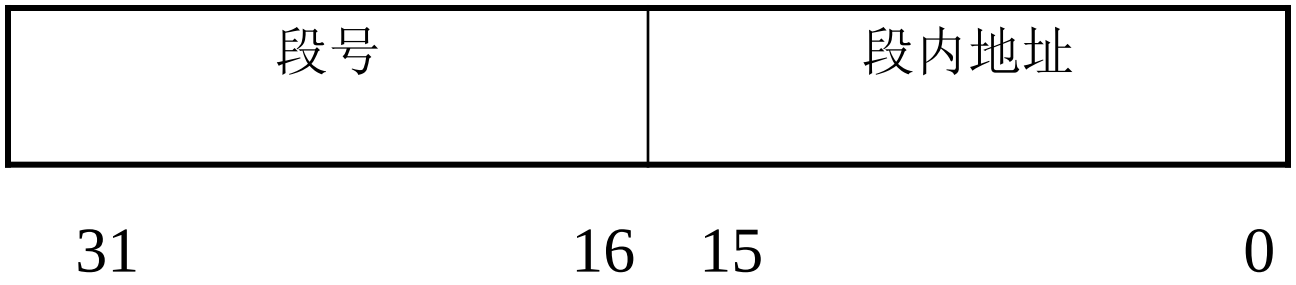
逻辑地址4500： 4500/1024，得页号4，页内地址404，因页号大于页表长度产生越界中断。

4.4.2 分段系统的基本原理

1. 分段

- 基本思想：按程序的逻辑结构，将程序的地址空间划分为若干段，各段大小可不相同。在进行存储分配时，以段为单位，这些段在内存中可以不相邻接。

分段地址中的地址具有如下结构：



2. 段表

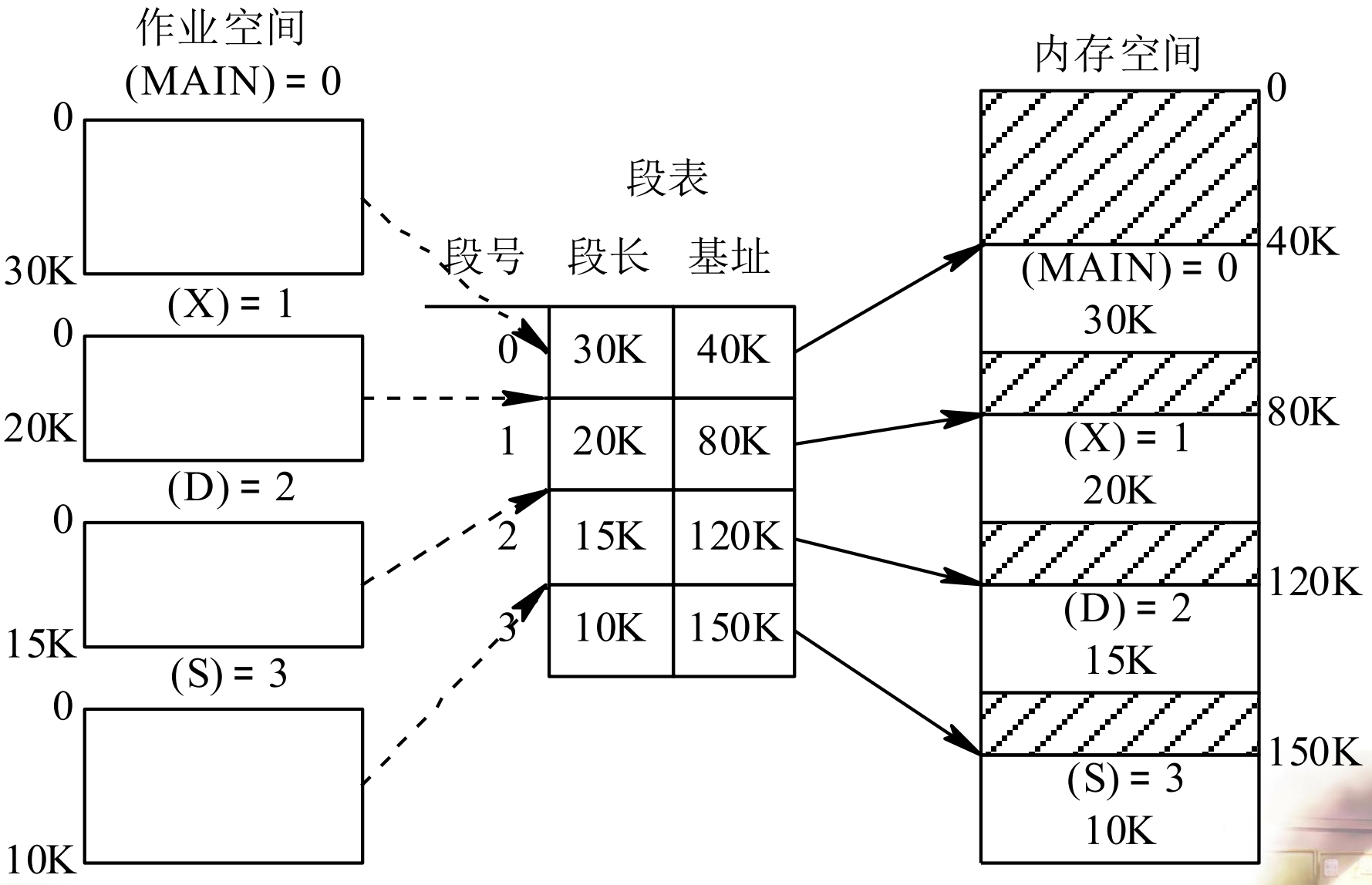


图 4-16 利用段表实现地址映射

3. 地址变换机构

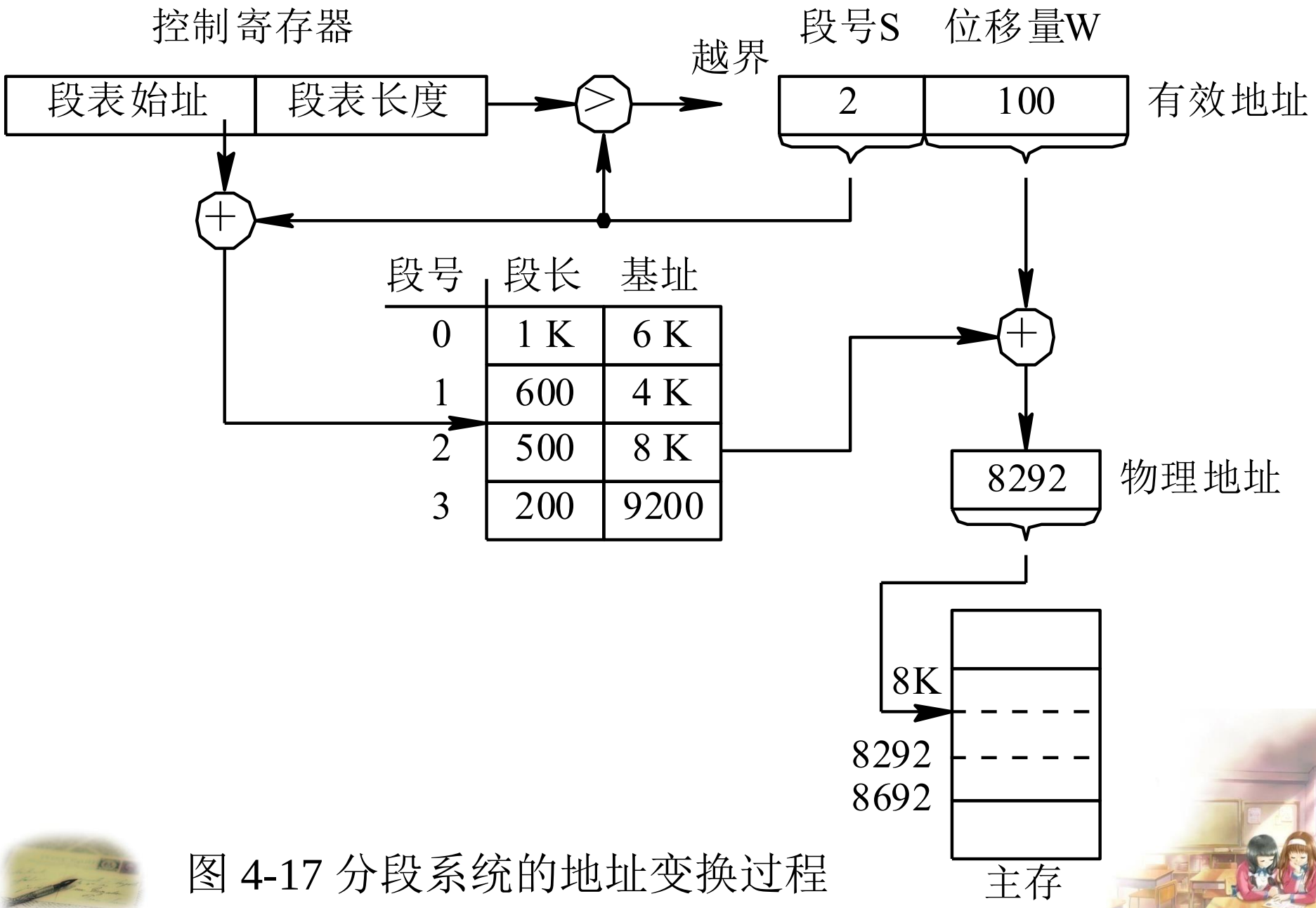


图 4-17 分段系统的地址变换过程

4. 分页和分段的主要区别

（1）页是信息的物理单位，段是逻辑单位

（2）页长度固定，段长度不固定（由用户指定）

（3）一维与二维

4.4.3 信息共享

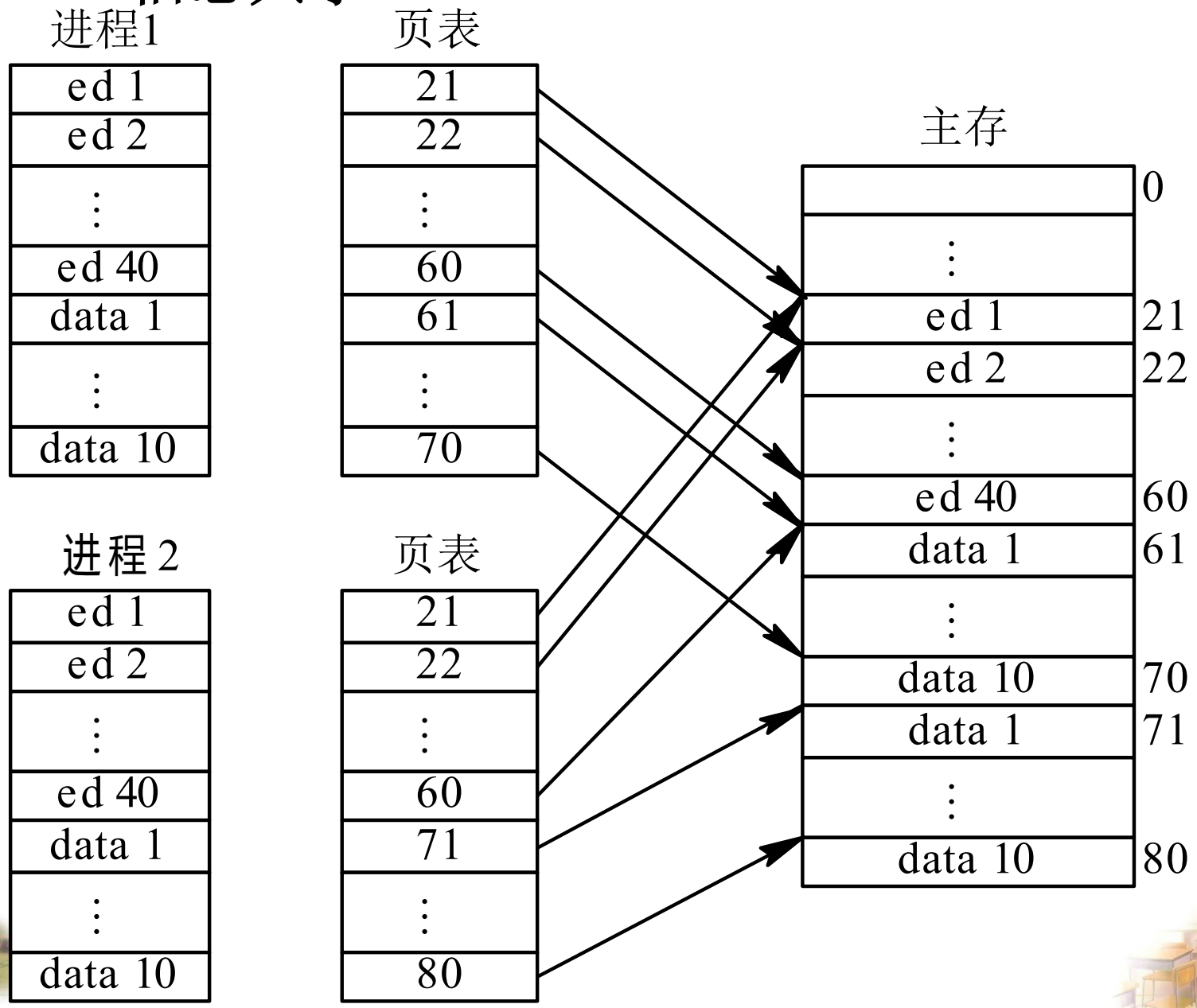


图 4-18 分页系统中共享editor的示意图

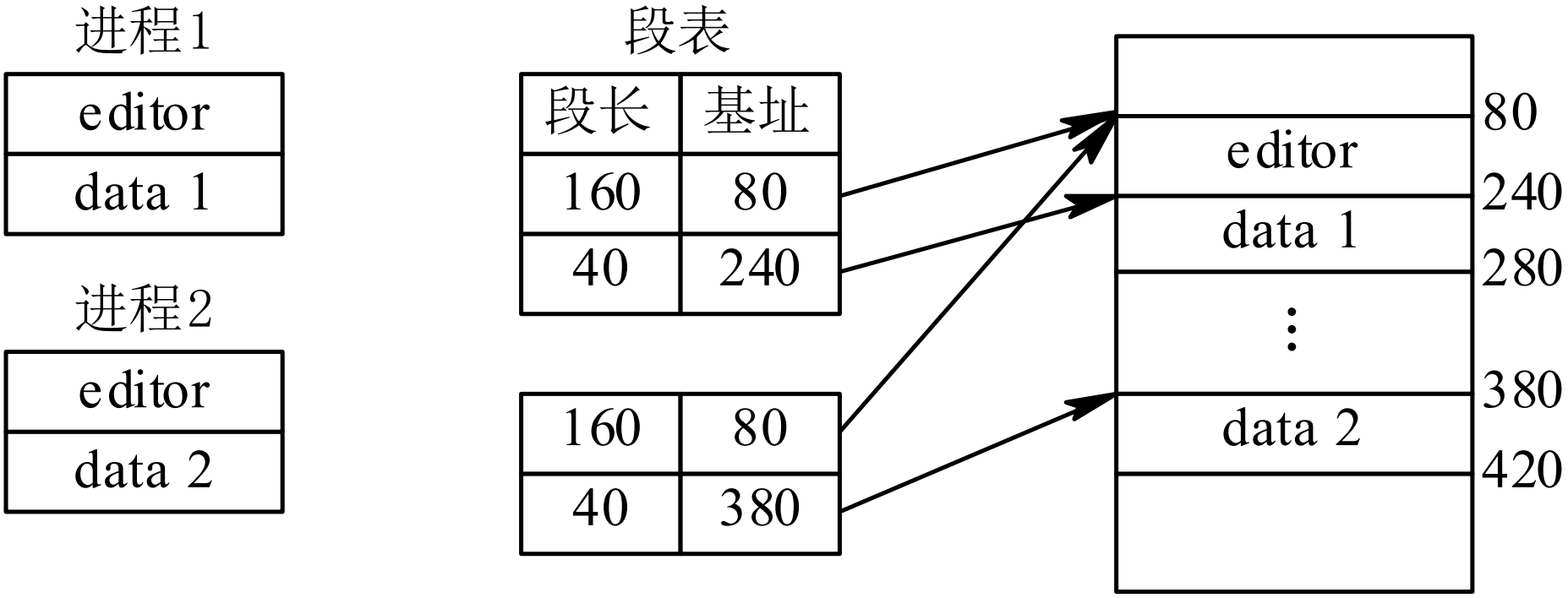


图 4-19 分段系统中共享editor的示意图

段式管理的优缺点

优点：

1. 程序的各段可独立编译（修改一个过程不会影响其它无关过程）
2. 可采用不同的保护措施（段只包含一种类型的对象，可以有针对这种特定类型的合适的保护）
3. 便于共享某些段（常见的例子是共享库，如图形库）

缺点：

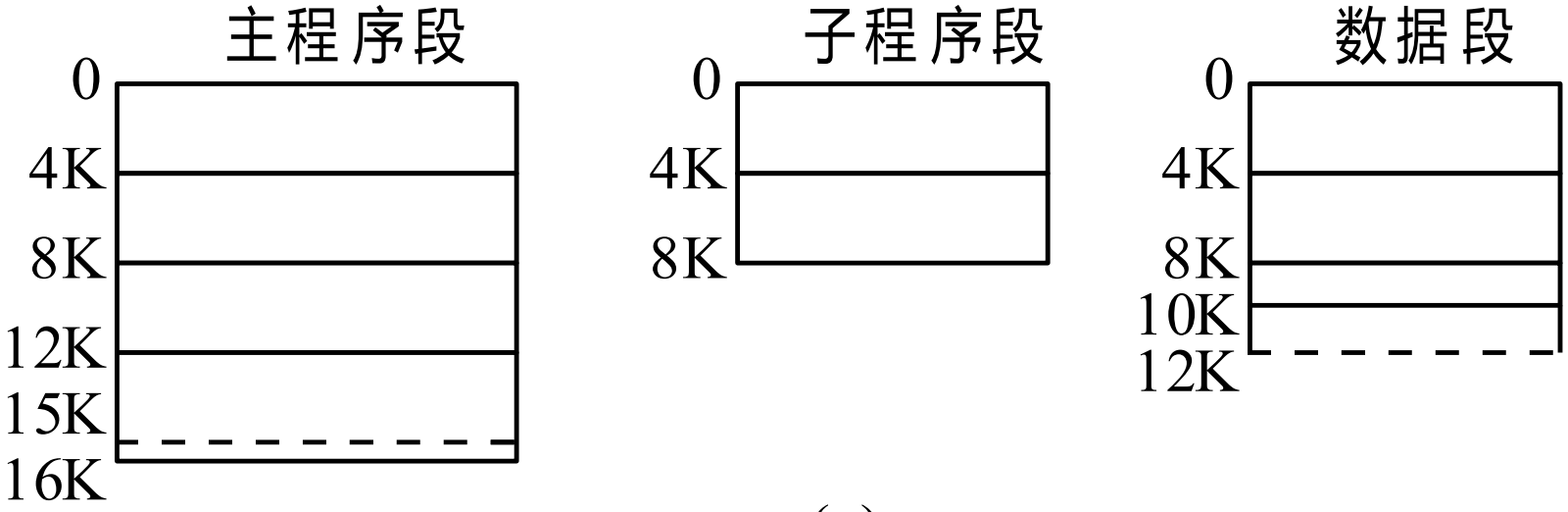
1. 段长受限制（段长不定会出现空闲区上内存的浪费）
2. 段是作为一个整体调入调出，操作时间长

4.4.4 段页式存储管理方式

- 分页优点：提高内存利用率
- 分段优点：方便用户，易于共享，保护，动态链接。

1. 基本原理

- 面对用户程序的地址空间，采用段式分割
- 内存分为长度相等的若干块
- 将每段划分为页，也常与内存块相等



(*a*)

(*b*)

图 4-20 作业地址空间和地址结构

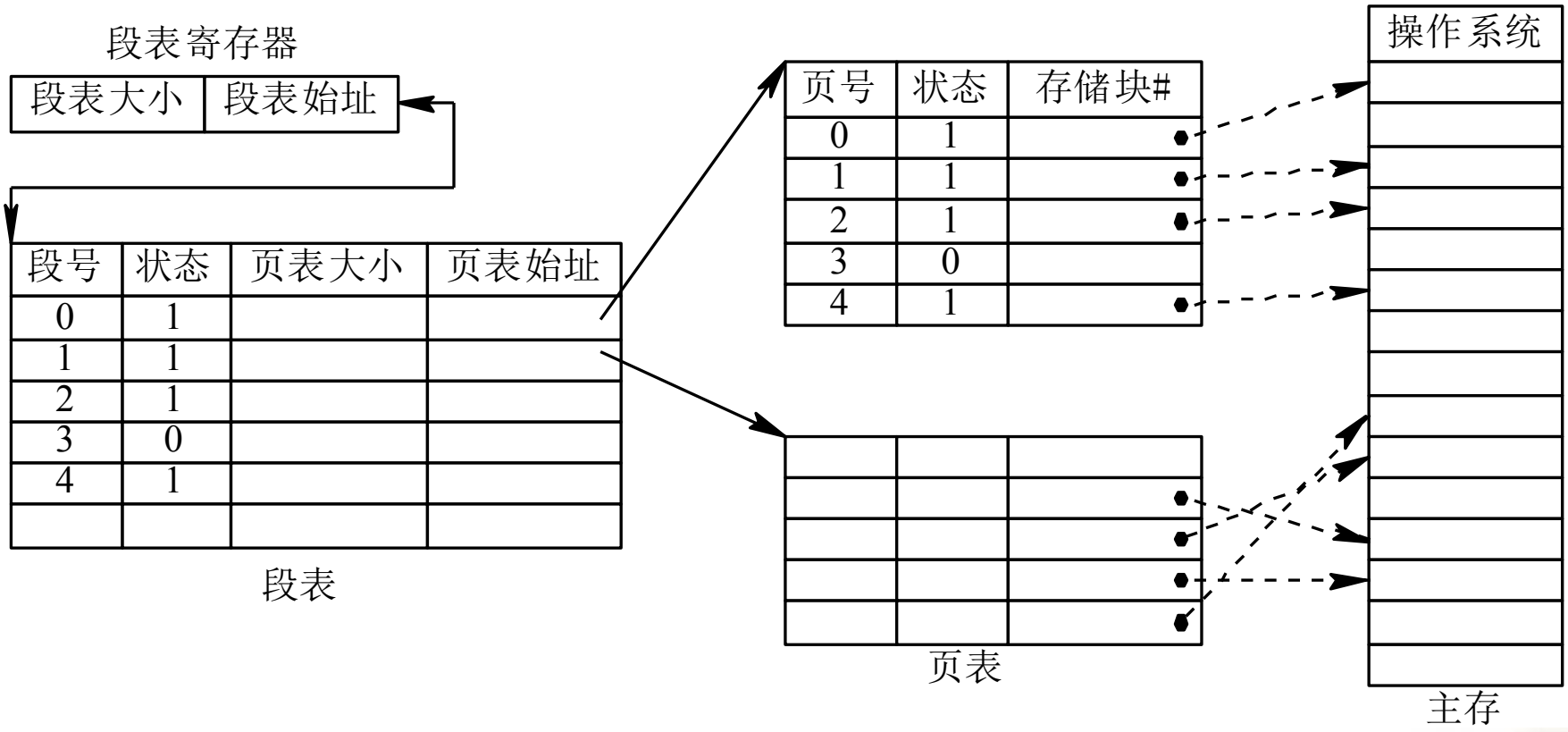


图 4-21 利用段表和页表实现地址映射

2. 地址变换过程

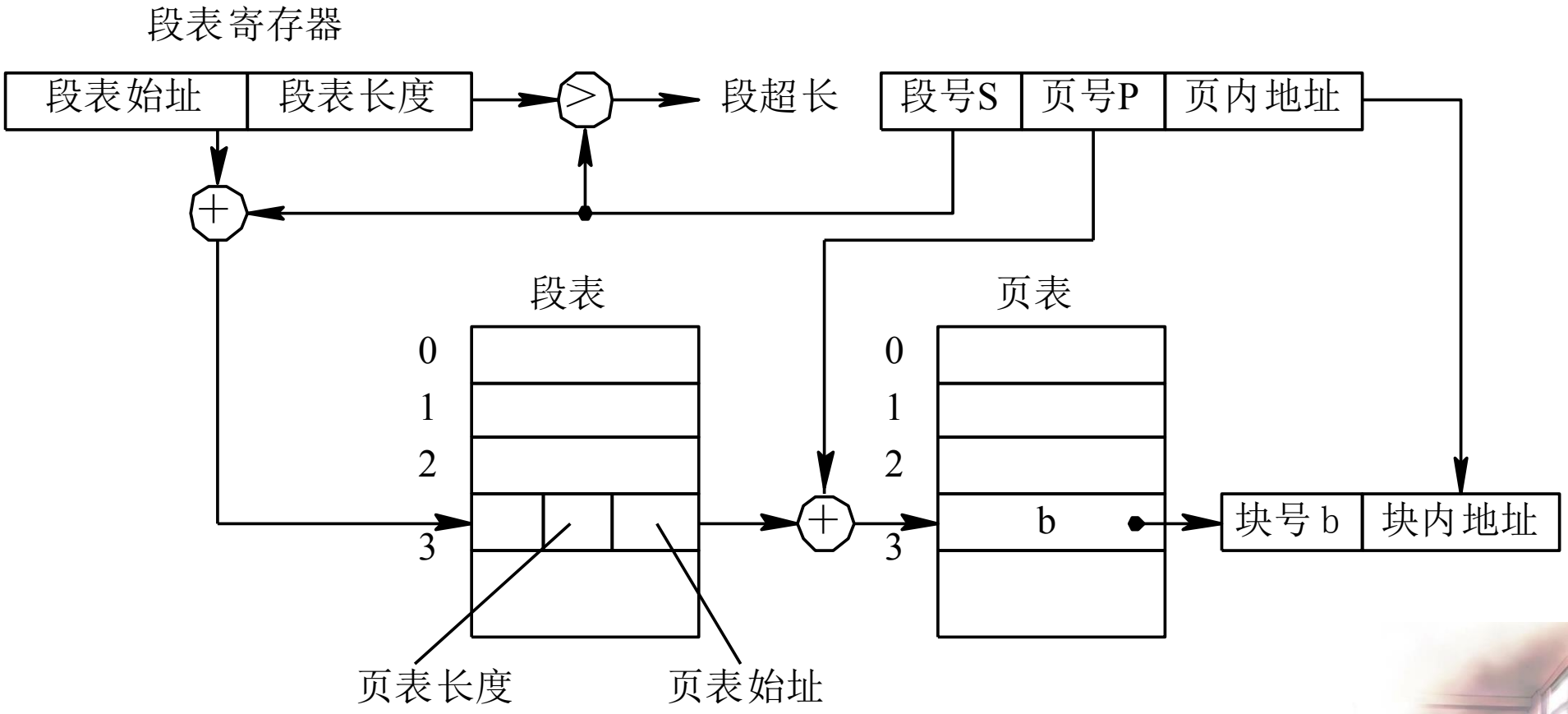


图 4-22 段页式系统中的地址变换机构

例：对于下表所示段表，请将逻辑地址（0，137），（1，4000），（2，3600），（5，230）转换成物理地址。

段号	基址	段长
0	50K	10K
1	60K	3K
2	70K	5K
3	120K	8K