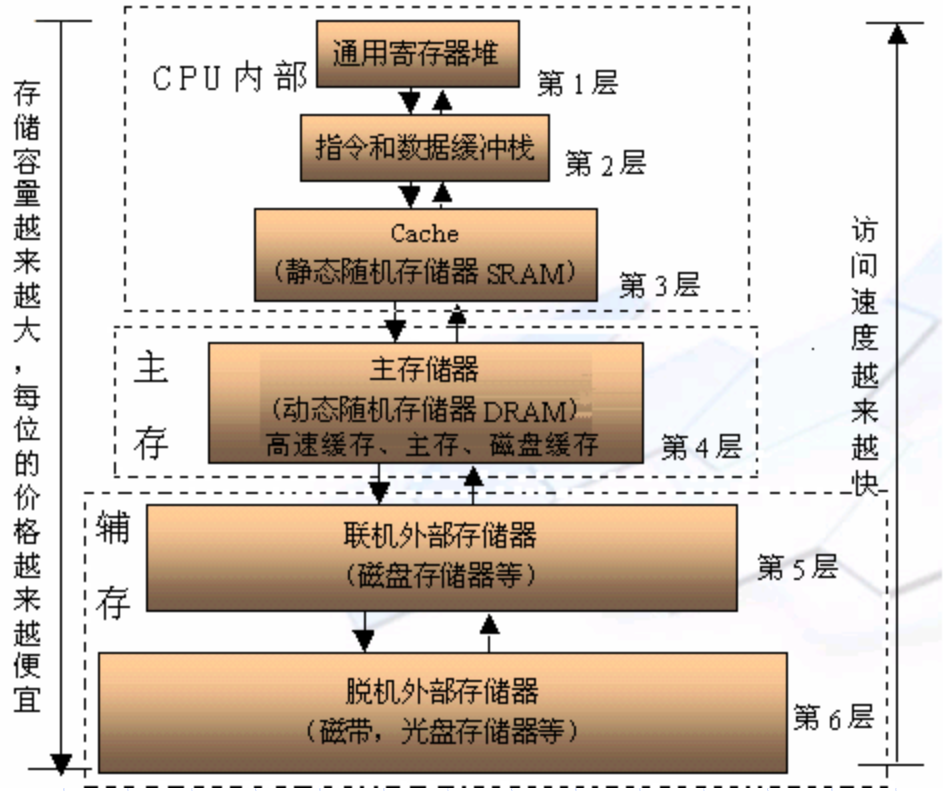
第四章　存储器管理

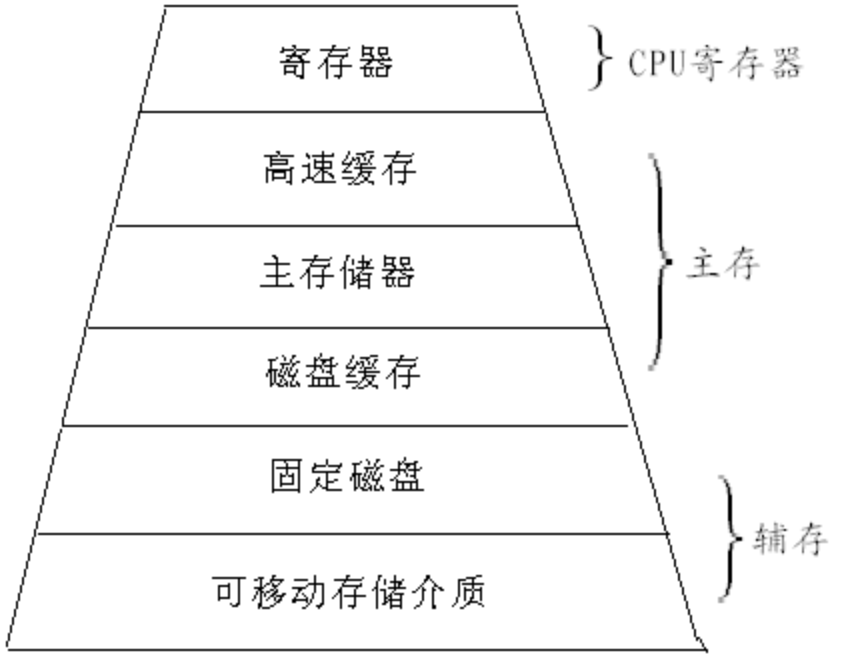
§4.1 存储器的层次结构

- 在现代计算机系统中，存储器是信息处理的来源与归宿，占据重要位置。
- 但是，在现有技术条件下，任何一种存储装置，都无法同时从速度与容量两方面，满足用户的需求。
- 实际上它们组成了一个**速度由快到慢，容量由小到大**的存储装置层次。

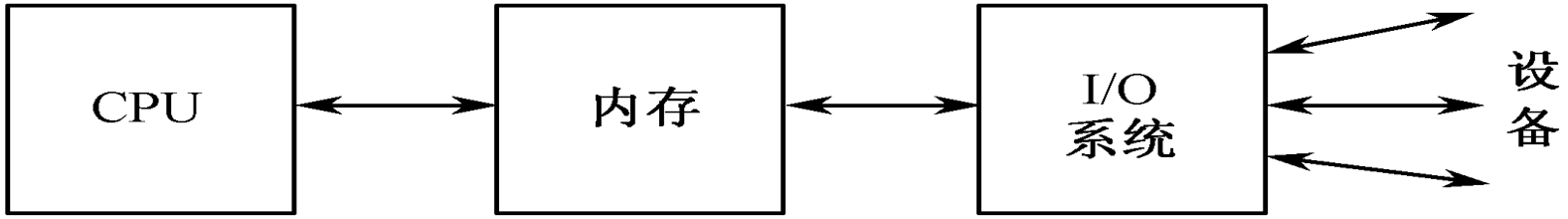
存储器的层次结构



存储器的层次结构



- 内存是现代计算机系统的中心，是指CPU能直接存取指令和数据的存储器，CPU和I/O设备都要和内存打交道。



- 内存由很大的一组字或字节所组成，每个字或字节都有它们自己的编号，称为内存地址。
- 对内存的访问是通过一系列对指定地址单元进行读写来实现的。

§4.2 程序的装入和链接

程序的装入：

- 绝对装入
- 可重定位装入
- 动态运行时装入

程序的链接

- 静态链接
- 装入时动态链接
- 运行时动态链接

§4.3 连续分配存储管理方式

4.3.1 单一连续分配

整个用户区仅装入一道用户程序。

4.3.2 分区管理的基本原理

- 分区管理的基本原理是给每一个内存中的进程划分一块适当大小的存储区(分区),以连续存储各进程的程序和数据，使得各进程在自己的分区中和其他进程并发的执行。
- 按分区的时机，分区管理可以分为固定分区和动态分区两种方法。

1、固定分区法：

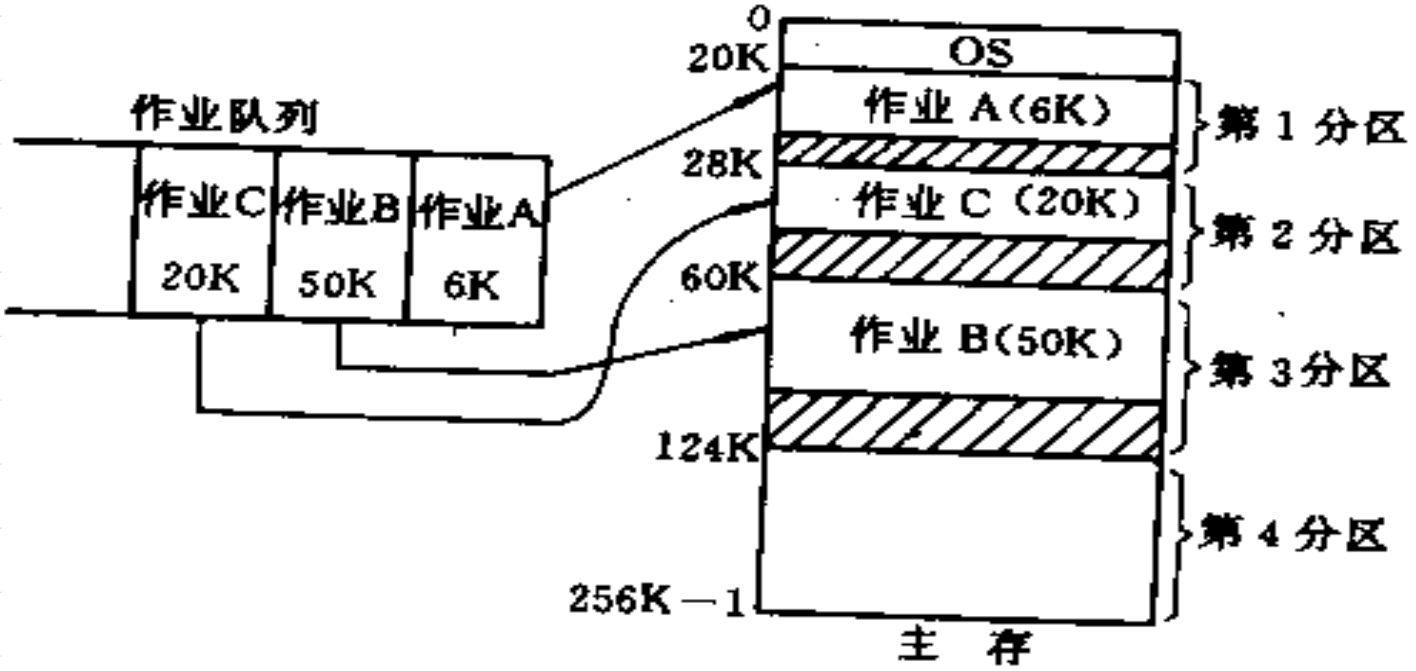
- 把内存区固定地划分为若干个大小不等的区域。划分的原则由系统操作员或操作系统决定。
- 分区一旦划分结束，在整个执行过程中每个分区的长度和内存的总分区个数将保持不变。

§4.2 连续内存分配

- 在固定分区管理方式下，每个用户作业占用一个**连续**的分区区域，作业的程序和数据一旦装入分区后就不能再移动，所以它通常采用**静态地址重定位**。
- 在系统运行期间，对于要装入系统的用户作业，由操作系统根据用户作业的大小，选择一个尺寸合适的分区分配给该程序使用。
- 固定分区管理使用一个称为**分区说明表**的数据结构对用户作业进行存储分配。分区说明表中每一个表项对应一个分区，它记载着这个分区的**序号、空间大小、起始地址和使用状况**。
- 当用户作业要求装入系统时，操作系统通过分区说明表查找内存的空闲区域，然后根据作业的大小，按照一定的分配策略，选择一个空闲分区分配给该作业，并把分区说明表中该分区标明已占用。

§4.3 连续分配存储管理方式

- 某系统的内存容量为256K，操作系统占用低地址的20K，其余空间划分成4个固定大小的分区。如下图：



固定式分区

§4.3 连续分配存储管理方式

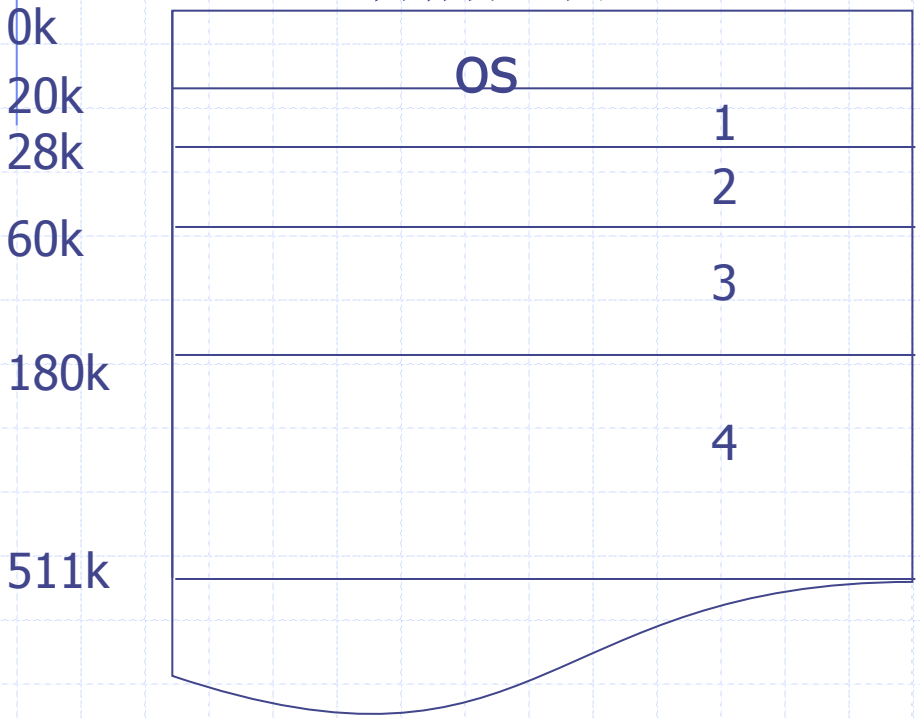
分区说明表

分区号	大小(KB)	始址	状态
1	8	20	已分配
2	32	28	已分配
3	64	60	已分配
4	132	124	未分配

§4.3 连续分配存储管理方式

例：在某系统中，采用固定分区分配管理方式，内存分区（单位字节）情况如图所示，现有大小为1K、9K、33K、121K的多个作业要求进入内存，试画出它们进入内存后的空间分配情况，并说明主存浪费多大？

（1）内存分区图

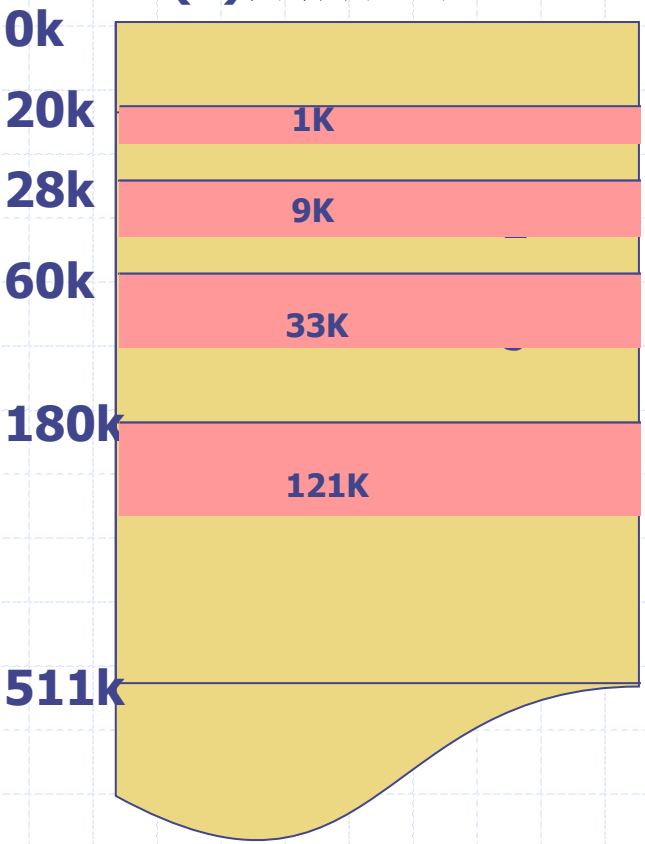


（2）分区说明表

区号	大小	起址	状态
1	8k	20k	未分配
2	32k	28k	未分配
3	120k	60k	未分配
4	331k	180k	未分配

解：根据分区说明表，将4个分区依次分配给4个作业，同时修改分区说明表，其内存分配和分区说明表如下所示：

(1)内存分配图



（2）分区说明表

区号	大小	起址	状态
1	8k	20k	已分配
2	32k	28k	已分配
3	120k	60k	已分配
4	331k	180k	已分配

(3)主存浪费空间=(8-1)+(32-9)+(120-33)+(331-121)

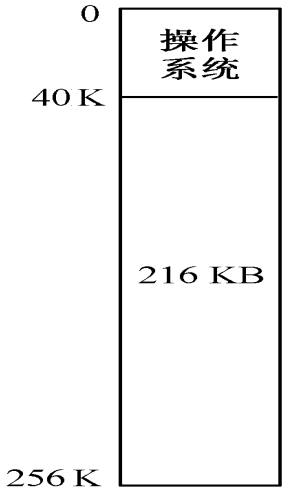
=7+23+87+210=327(k)

§4.3 连续分配存储管理方式

2、动态分区法

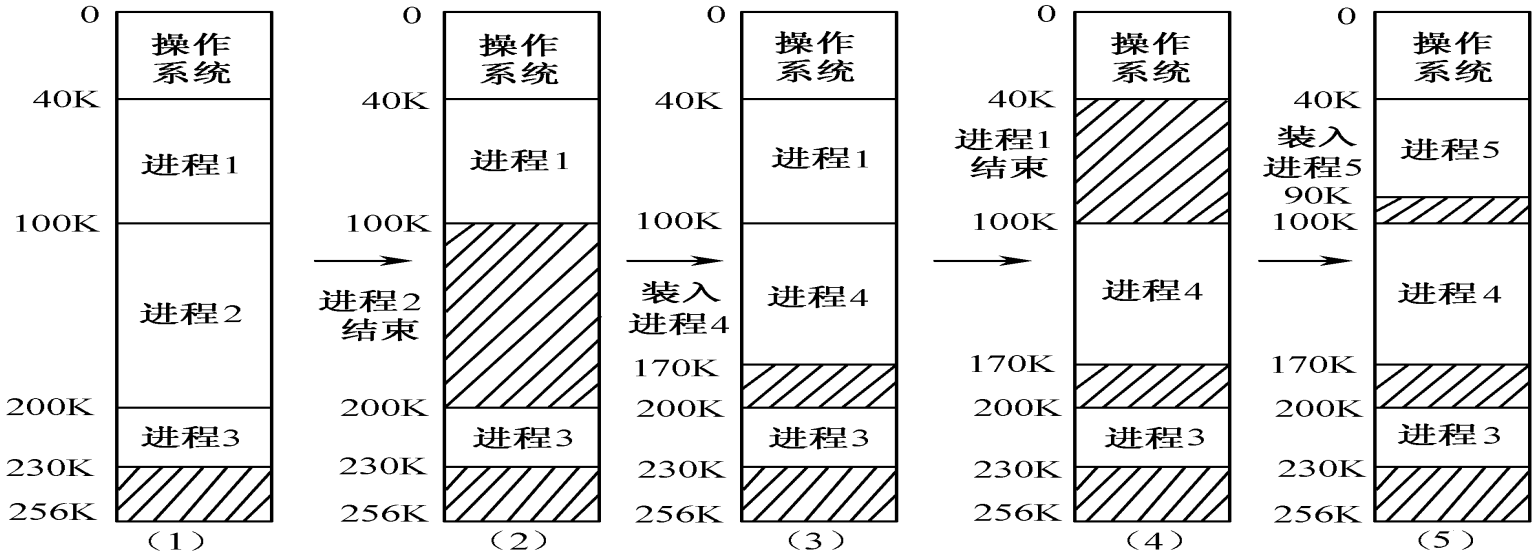
- 动态分区法在作业执行前并不建立分区，分区的建立是在作业的处理过程中进行的，且其大小可随作业或进程对内存的要求而改变。这就改变了固定分区法中那种即使是小作业也要占据大分区的浪费现象，从而提高了内存的利用率。
- 采用动态分区法，在系统初启时，除了操作系统中常驻内存部分之外，只有一个空闲分区。随后，分配程序将该区依次划分给调度选中的作业或进程。

§4.3 连续分配存储管理方式



进程队列		
进程	需要内存大小	运行时间
1	60 KB	10
2	100 KB	5
3	30 KB	20
4	70 KB	8
5	50 KB	15

（a）内存初始情况和进程队列

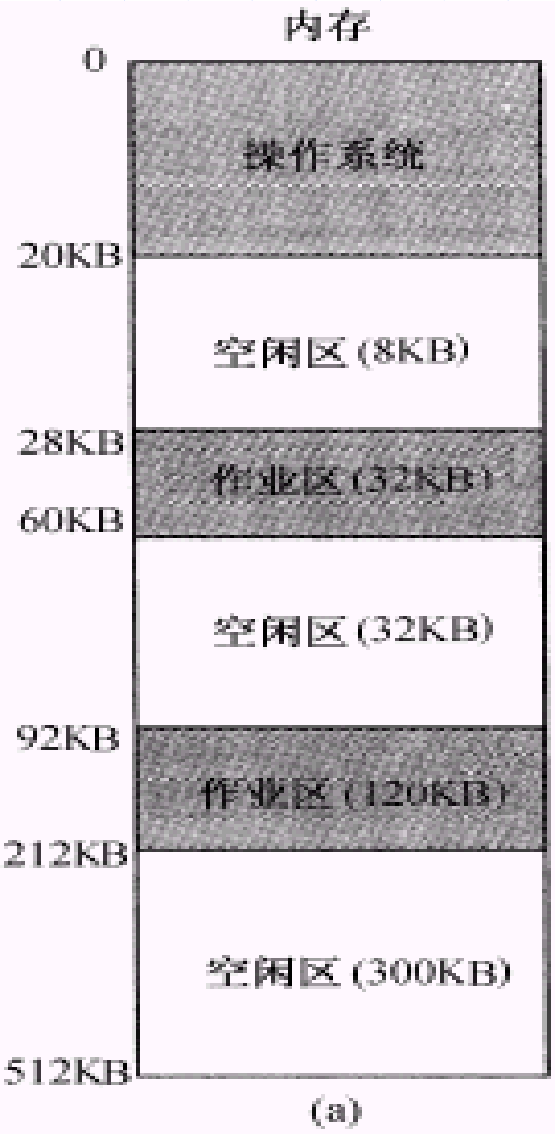


（b）内存分配和进程调度情况

§4.3 连续分配存储管理方式

- 在动态分区存储管理中，要有相应的数据结构来登记空闲区的说明信息，它包括空闲区的大小和位置。
- 不同系统根据设计要求采用不同的结构。常用的有表结构和队列结构。
- 空闲区表的每个表目记录一个空闲区，主要参数包括区号、长度和起始地址。空闲区队列则是利用每个内存空闲区的头几个单元存放本空闲区的大小及下个空闲区的起始地址，从而把所有的空闲区链接起来。

§4.3 连续分配存储管理方式



序号	起始地址	尺寸	状态
1	—	—	空
2	28KB	32KB	作业B
3	—	—	空
4	92KB	120KB	作业D
5	—	—	空

(b) 已分配表

序号	起始地址	尺寸	状态
1	20KB	8KB	空闲
2	60KB	32KB	空闲
3	212KB	300KB	空闲
4	—		空
5	—		空

(c) 空闲区表

图 3-11　内存分区的管理表格

§4.3 连续分配存储管理方式

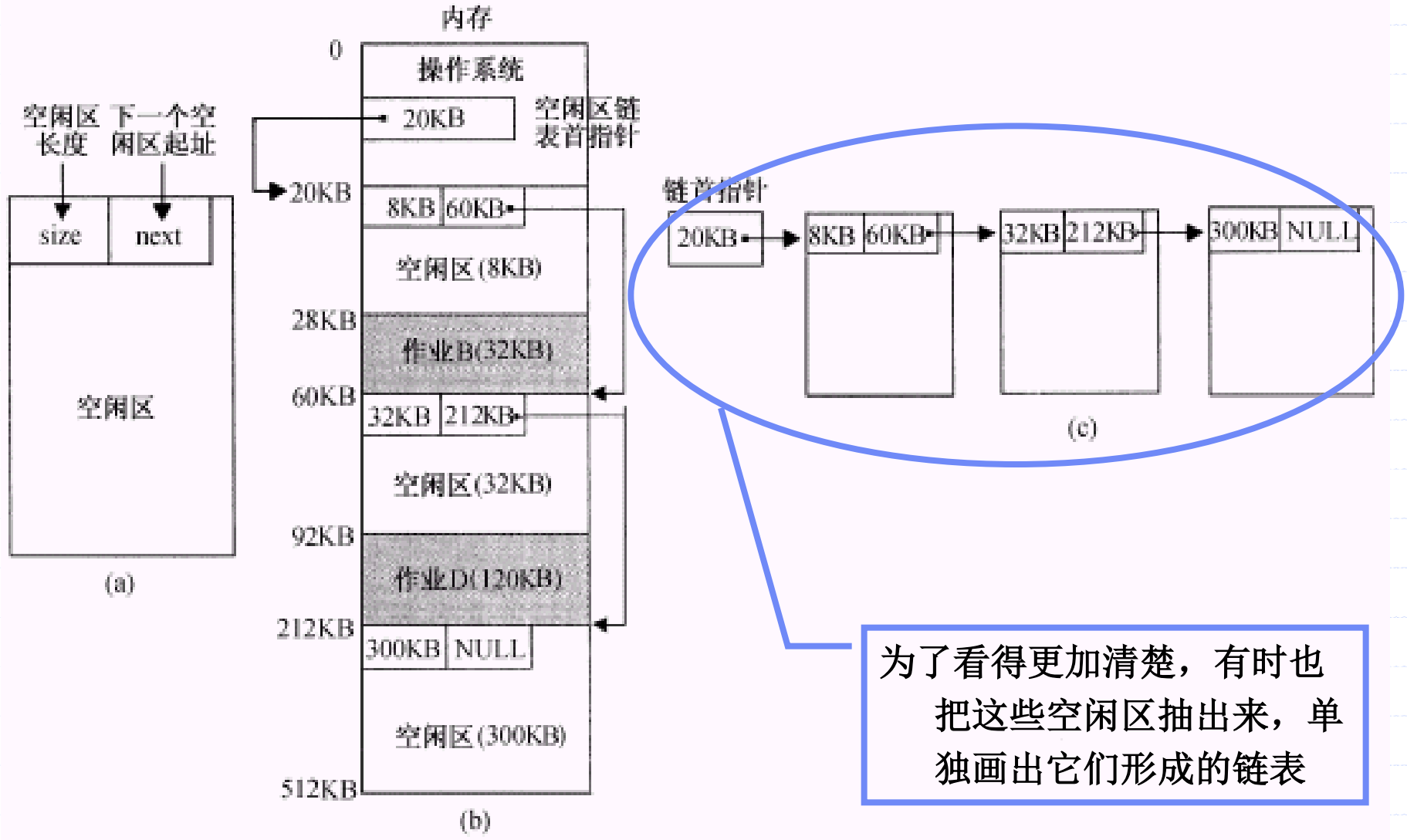


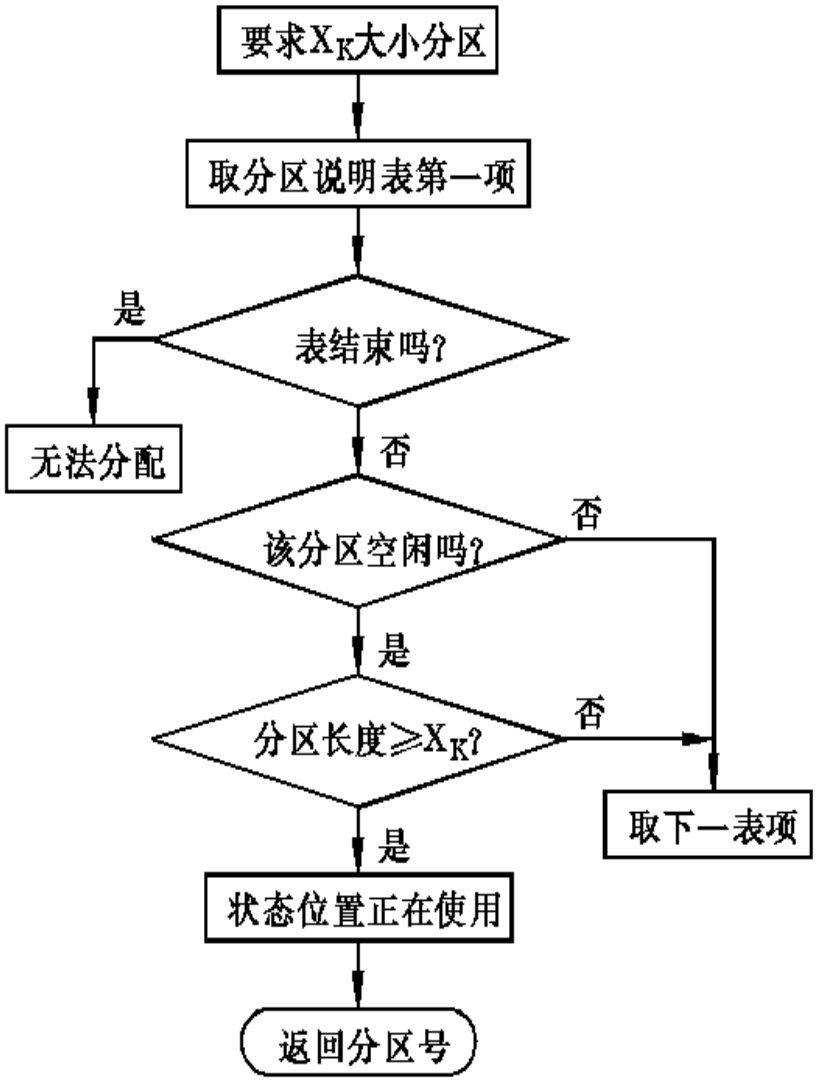
图 3-14 空闲分区组成的单链表

§4.3 连续分配存储管理方式

分区的分配和回收

1. 固定分区的分配和回收

当用户程序要装入执行时，存储管理程序根据用户程序的大小查询分区说明表，从中找出一个满足要求的空闲分区，并将其分配给申请者。

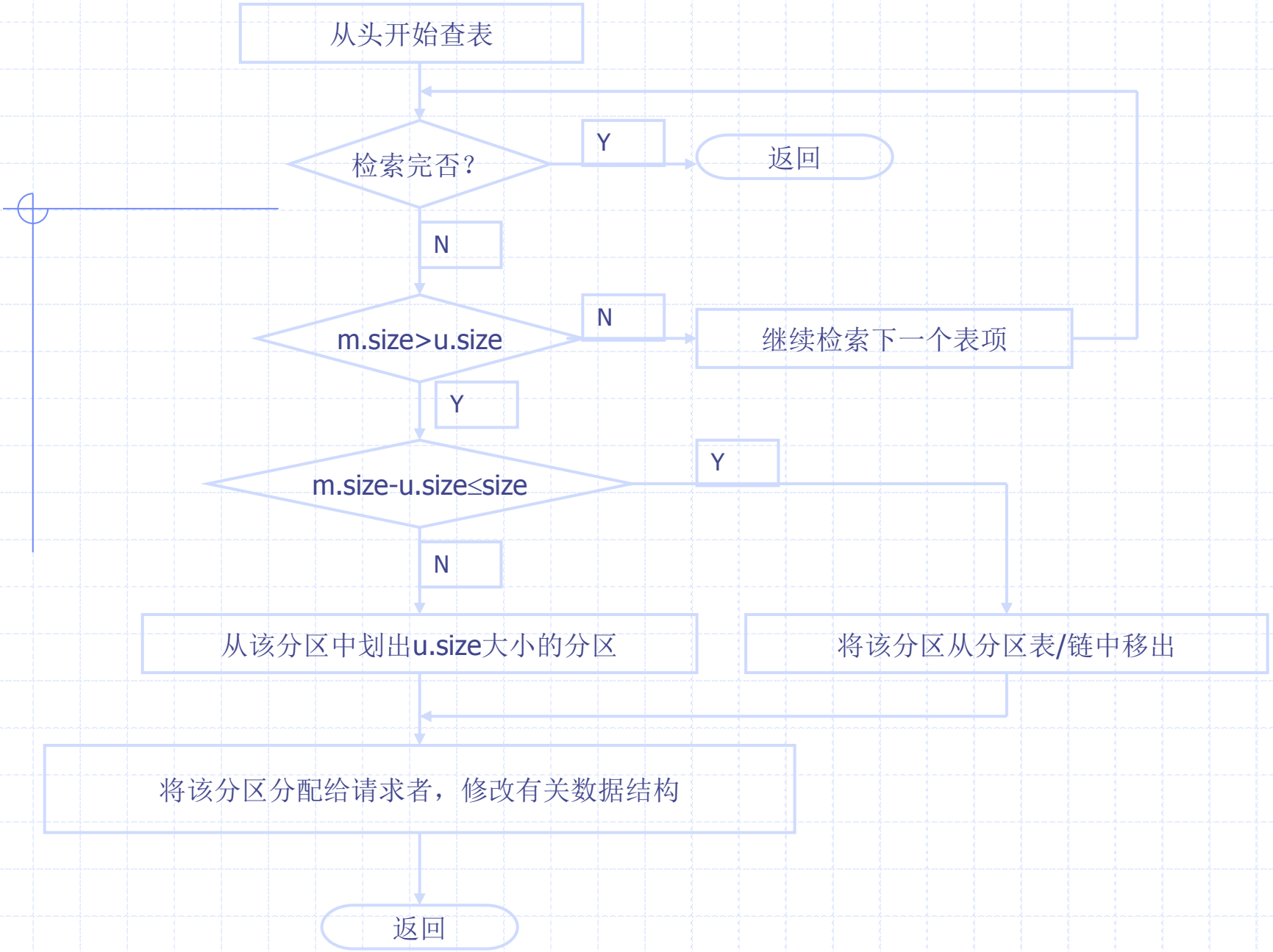


§4.3 连续分配存储管理方式

2、动态分区的分配和回收

（1）分区的分配

- 分区的分配是指系统根据用户的请求，在空闲区表或空闲区队列中寻找一个满足用户要求的空闲区，把这个空闲区分配给用户。当用户要求一个大小为**SIZE**的存储空间时，系统查询空闲区表或空闲分区队列，找一个大于或等于**SIZE**的空闲区。
- 分配时会出现以下三种情况：一是系统中无满足要求的空闲区，则分配失败。二是空闲区大小与SIZE相等，则修改空闲区表相应表目，向用户返回该空闲区首址。三是空闲区大于SIZE，这时将空闲区一分为二。
- 将一个空闲区分成二部分有两种办法：一是从空闲区的上部开始划出SIZE大小的空闲区给用户；二是从空闲区的底部开始向上划出SIZE大小的空闲区给用户。


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
N
从该分区中划出u.size大小的分区
将该分区从分区表/链中移出
将该分区分配给请求者，修改有关数据结构
返回

§4.3 连续分配存储管理方式

- **动态分区分配算法：**

（1）首次适应算法：

- 要求空闲区按首址递增的次序组织空闲区表（队列）。
- 申请：取最小可满足区域；
- 优点：尽量使用小空闲区， 保持大空闲区。缺点：可能形成碎片

（2）最佳适应算法：

- 要求按空闲区大小从小到大的次序组成空闲区表（队列）。
- 申请：取最小可满足区域；
- 优点：尽量使用小空闲区， 保持大空闲区。缺点：可能形成碎片

（3）最坏适应算法

- 要求空闲区按大小递减的顺序组织空闲区表（或队列）。
- 申请：取最大可满足区域；
- 优点：防止形成碎片。缺点：分割大空闲区域。

§4.3 连续分配存储管理方式

例题：（华中科技大学2002）某系统采用动态分区存储管理技术。某时刻在内存中有三个空闲区，它们的首地址和大小分别是：空闲区1（100KB，10KB）、空闲区2 （200KB，30KB）、空闲区3 （300KB，15KB）。现有如下作业序列：作业1要求15KB、作业2要求16KB、作业3要求10KB。要求：

（1）画出该时刻内存分布图；

（2）用首次适应算法和最佳适应算法画出此时的自由主存队列结构；

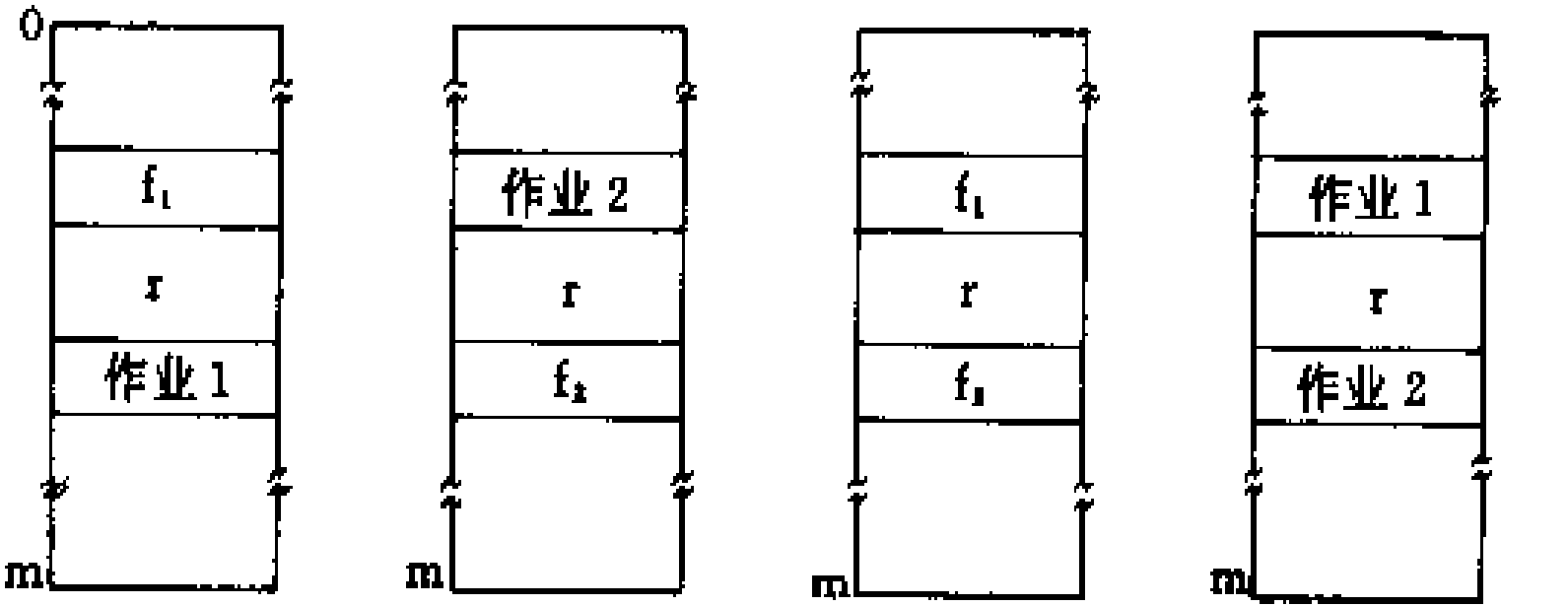
（3）哪种算法能将该作业装入内存（给出简要的分配过程）。

§4.3 连续分配存储管理方式

(2) 动态分区的内存回收

当某一个用户作业完成释放所占分区时，系统应进行回收。在可变式分区中，应该检查回收区与内存中前后空闲区是否相邻，若相邻，则应进行合并，形成一个较大的空闲区，并对相应的链表指针进行修改；若不相邻，应将空闲区插入到空闲区链表的适当位置。

§4.3 连续分配存储管理方式



(a)回收分区r上邻空白区　(b)回收分区r下邻空白区　(c)回收分区r上、下邻空白区　(d)回收分区r上、下邻已分配区

回收分区r与空白区邻接的几种情况

动态分区内存回收情况写小结

编号	情形	说明
1	回收区上邻空闲区	回收区与上邻空闲区合并为一个大的空闲区，该空闲区的首地址为上邻空闲区的首地址，大小为上邻空闲区和回收区大小的和，整个系统空闲区总数保持不变
2	回收区下邻空闲区	回收区与下邻空闲区合并为一个大的空闲区，该空闲区的首地址为回收区的首地址，大小为下邻空闲区和回收区大小的和，整个系统空闲区总数保持不变
3	回收区上、下均邻空闲区	回收区与上、下邻空闲区合并为一个大的空闲区，该空闲区的首地址为上邻区的首地址，大小为上、下邻空闲区和回收区大小的和，整个系统空闲区总数减少1个
4	回收区上、下邻已分配区	回收区成为新的空闲区，将其首地址和大小登记到空闲区表（队列）中相应的位置，整个系统的空闲区总加1

§4.3 连续分配存储管理方式

例题：某操作系统采用可变分区分配存储管理方法，用户区大小为512K且初始值为0，用空闲分区表管理空闲分区。若分配时采用分配空闲区低地址部分的方案，且初始时用户区的512K空间空闲，对于下列申请序列：

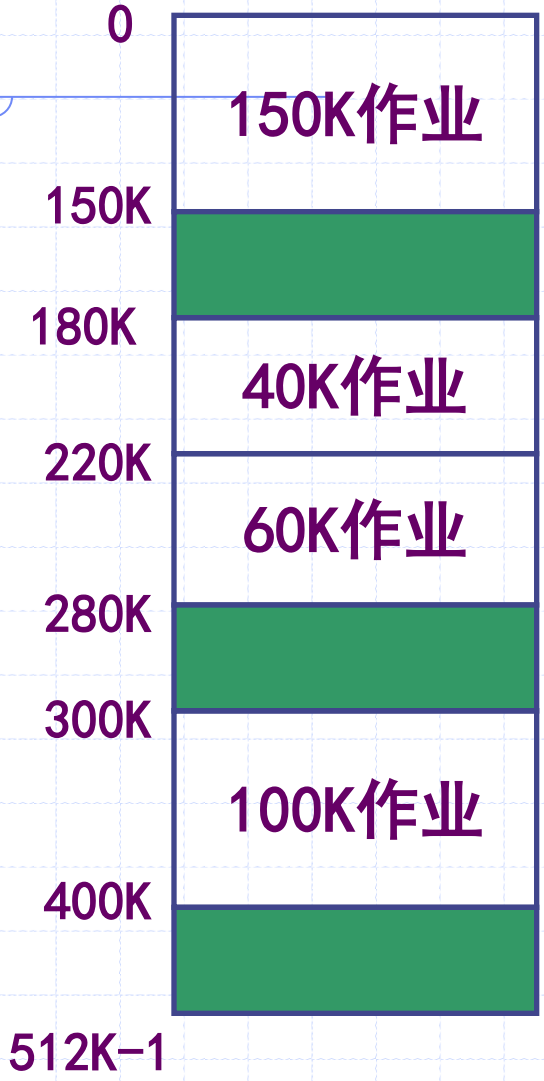
申请300K，申请100K，释放300K，申请150K，申请30K，申请40K，申请60K，释放30K

回答下列问题：

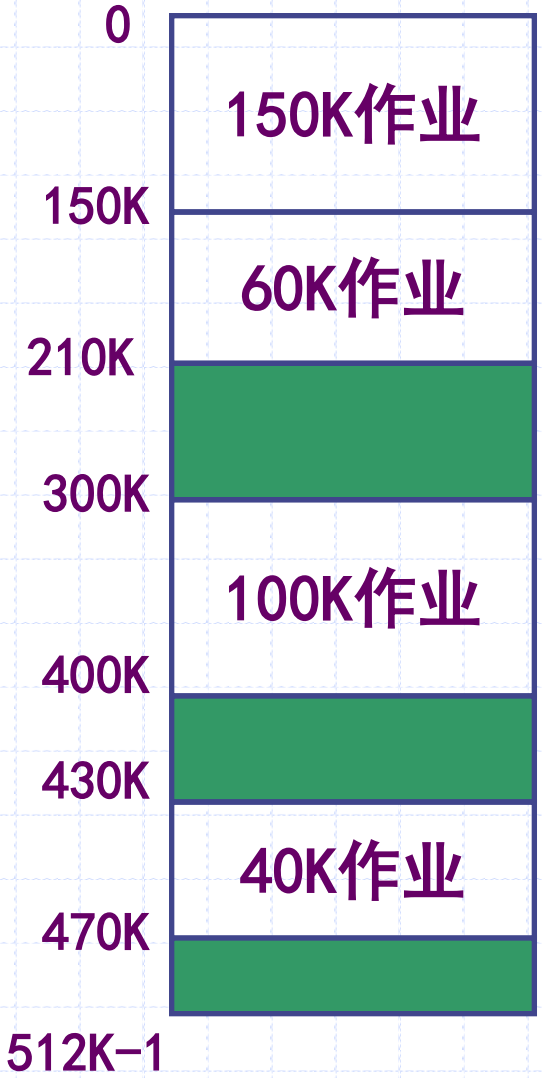
（1）请分别画出采用首次适应算法、最佳适应算法进行内存分配和回收后的内存使用状态。

（2）如果再申请100K，针对上述两种算法会有什么结果？

例题解答如下：



首次适应算法

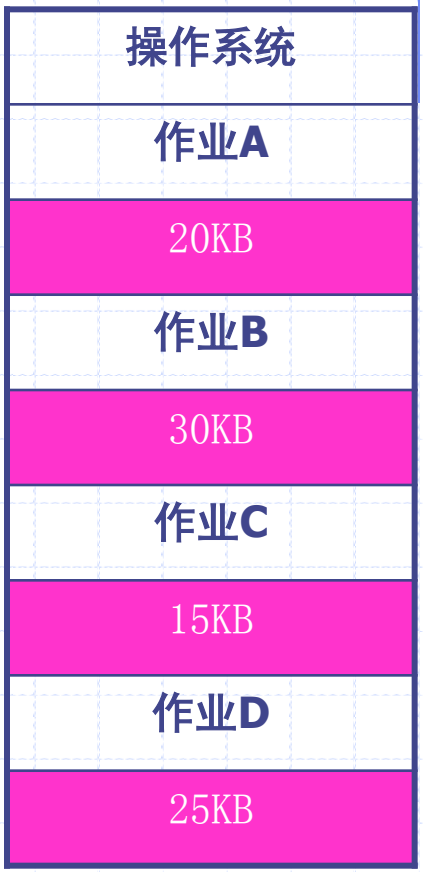


最佳适应算法

§4.2 连续内存分配

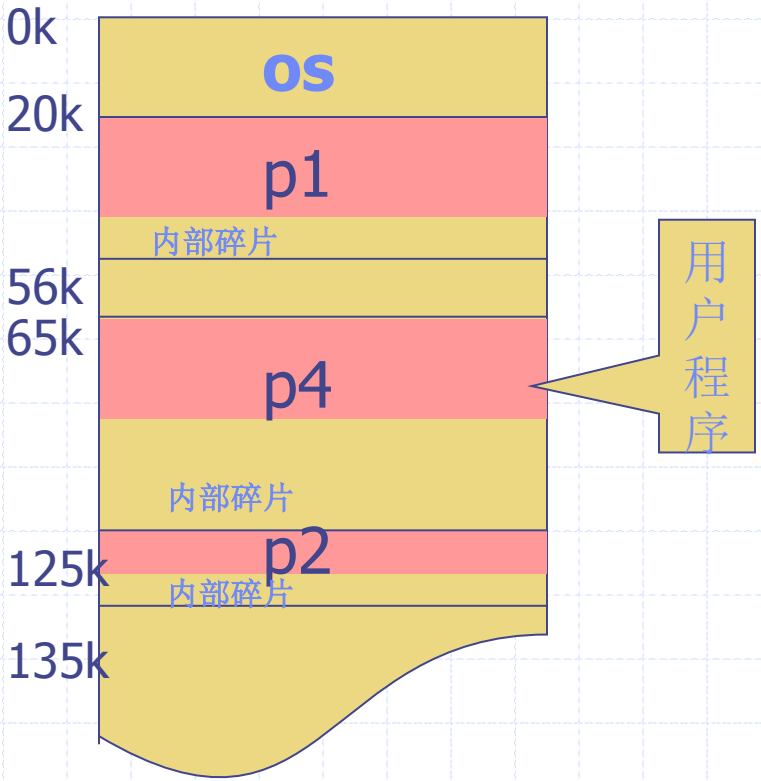
4.2.3 碎片问题

- 在连续内存分配中，必须把一个系统程序或用户程序装入一个连续的内存空间中。由于各个进程不但的申请和释放内存，导致在内存中出现大量的分散的小空闲区。内存中这种容量太小、无法利用的小分区称做“碎片”或“零头”。
- 如图所示系统中有四个小空闲分区，不相邻，但总容量为**90KB**，如果现有一作业要求分配**40KB**的内存空间，由于系统中所有空闲分区的容量均小于**40KB**，故此作业无法装入内存。



§4.3 连续分配存储管理方式

- **内部碎片**：指分配给作业的存储空间中未被利用的部分。如固定分区中存在的碎片。
- **外部碎片**：指系统中无法利用的小的空闲分区。如动态分区中存在的碎片。

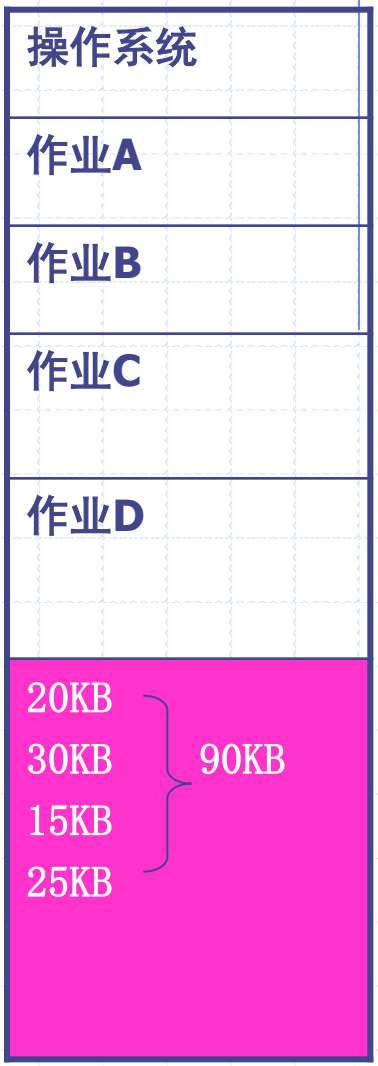


碎片问题的解决方法——拼接或紧凑或紧缩技术

- 将内存中所有作业移到内存一端（作业在内存中的位置发生了变化，这就必须对其地址加以修改或变换即称为重定位），使本来分散的多个小空闲分区连成一个大的空闲区。如图所示。这种通过移动作业从把多个分散的小分区拼接成一个大分区的方法称为拼接或紧凑或紧缩。
- 拼接时机：

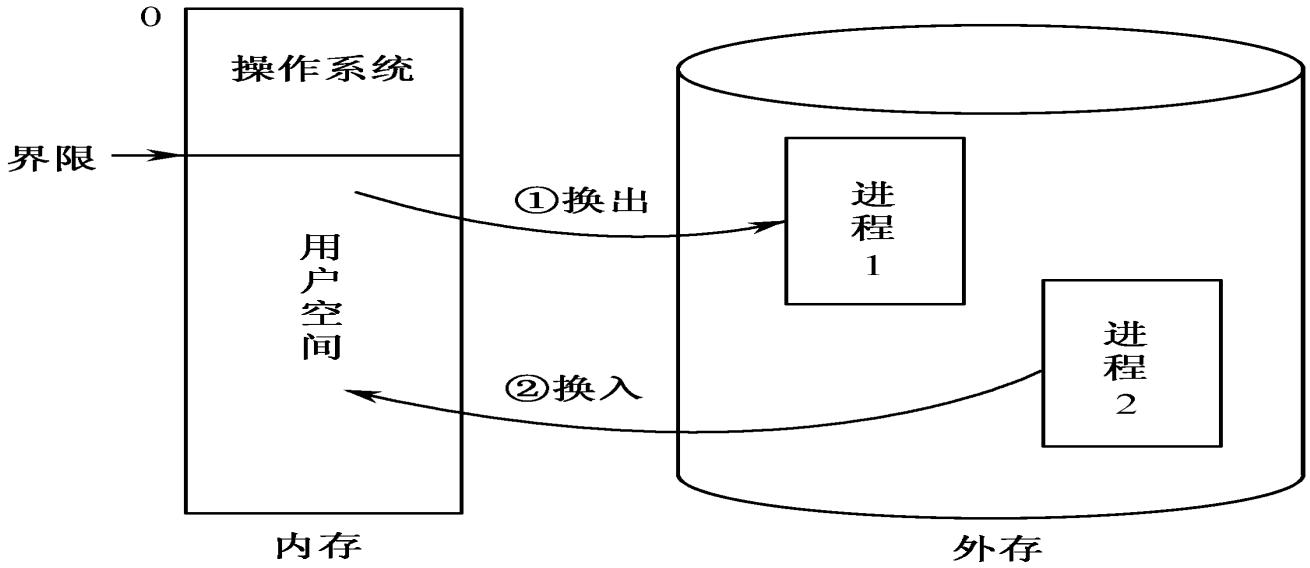
（1）分区回收时；当找不到足够大的空闲分区且总空闲分区容量可以满足作业要求时。

（2）定时。



§4.4 对换

对换是指先将内存某部分的程序或数据写入外存交换区，再从外存交换区中调入指定的程序或数据到内存中来，并让其执行的一种内存扩充技术 。

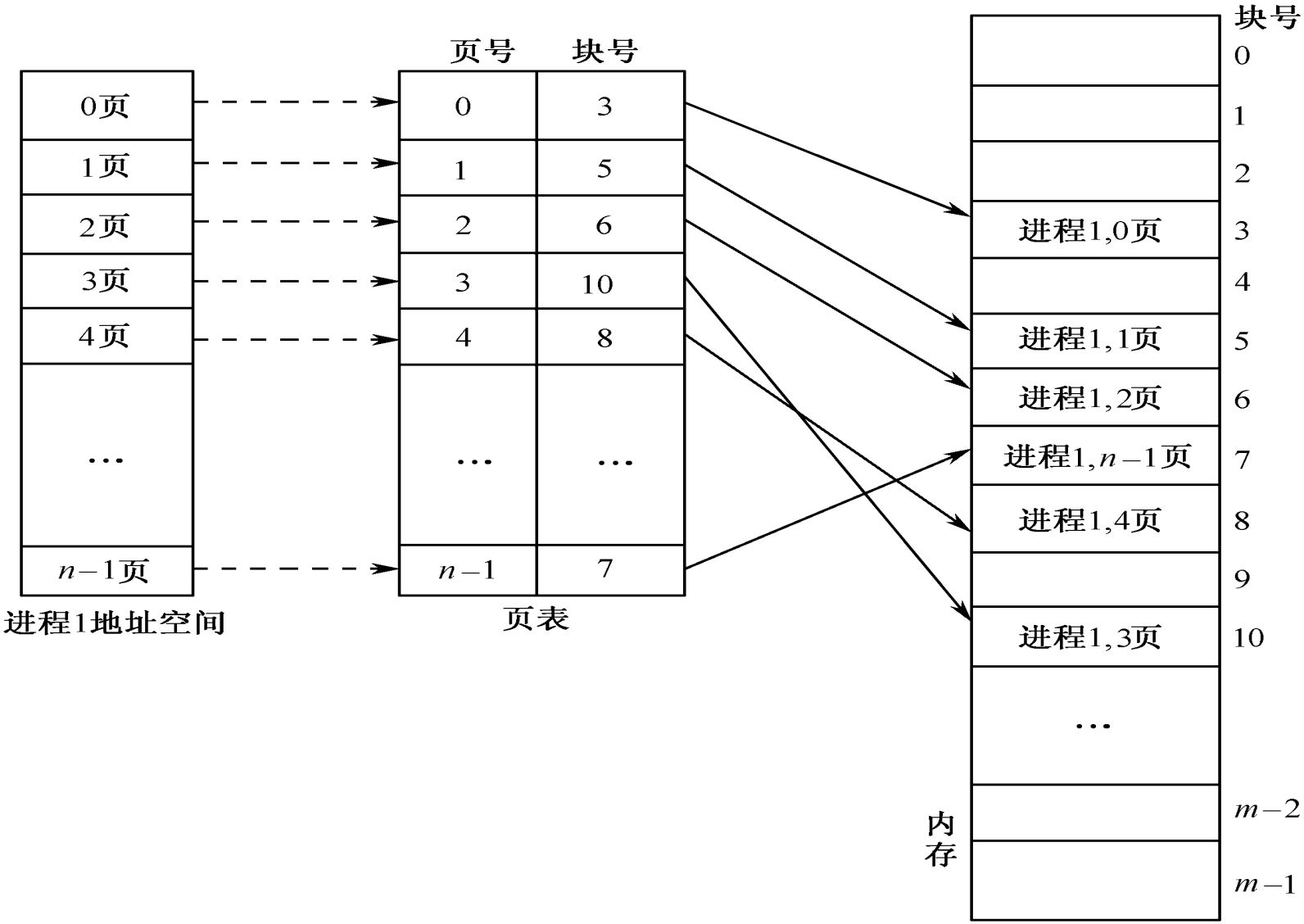


§4.5 分页存储管理方式

4.5.1 分页管理的基本原理

- 把用户程序的地址空间划分成若干大小相等的区域，每个区域称作页面或页。每个页都有一个编号，叫做页号。页号一般从0开始编号，如0，1，2，…等。
- 把内存空间划分成若干和页大小相同的物理块，这些物理块叫“帧”(frame)或内存块。同样，每个物理块也有一个编号，块号从0开始依次顺序排列。
- 以页为单位进行内存分配，并按作业的页数多少来分配。逻辑上相邻的页，物理上不一定相邻。

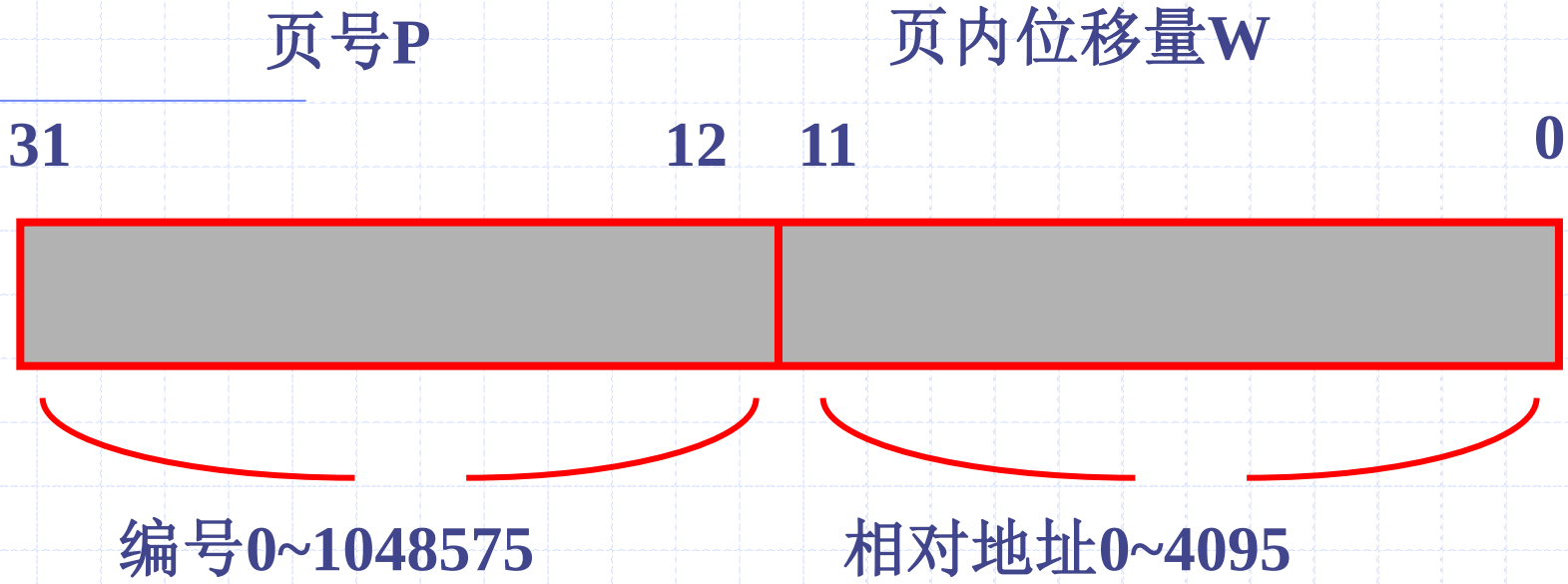
§4.5 分页存储管理方式


页号
块号
0页
1页
2页
3页
4页
…
n−1页
进程1地址空间
0
1
2
3
…
n−1
3
5
6
10
8
…
7
页表
块号
0
1
2
进程1,0页
3
4
进程1,1页
5
进程1,2页
6
进程1,n−1页
7
进程1,4页
8
9
进程1,3页
10
…
m−2
m−1
内存

§4.5 分页存储管理方式

- 在分页系统中，页面的大小是由硬件的地址结构所决定的。机器确定、页面大小便确定了。一般来说，页面的大小选择为2的若干次幂，根据计算机结构的不同，其大小从512B到16MB不等。
- 在分页系统中，由CPU生成的每个地址被硬件分成两个部分：页号（p）和页内偏移（w）。通常，如果逻辑地址空间为2^m，且页的大小为2^n单元（字节或词），那么逻辑地址的高m－n位表示页号，而低n位表示页偏移。这样，一个地址长度为20位的计算机系统，如果每页的大小为1KB(2^{10})，那么可以有2^{10}个页。

§4.5 分页存储管理方式



对于某台具体机器来说，其地址结构是一定的。如果给定的逻辑地址是A，页面的大小为L，则页号p和页内地址w可按下式求得：

$$p=INT[A/L],\quad w=[A]\ MOD\ L$$

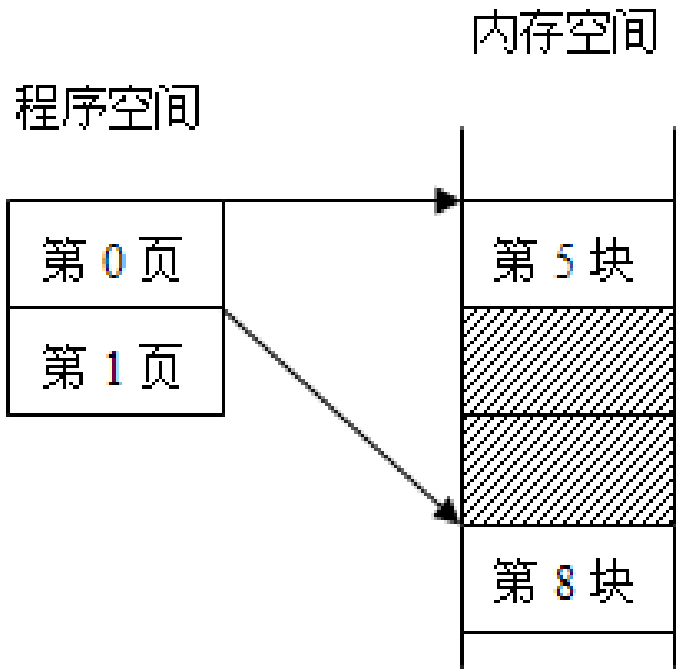
其中，INT是向下整除的函数，MOD是取余函数。例如，设系统的页面大小为1KB，A=3456，则p=INT(3456/1024)=3，w=3456 MOD 1024=384。

§4.5 分页存储管理方式

- 在分页系统中，允许将进程的各页离散地装入内存的任何空闲块中，这样就出现进程页号连续，而块号不连续的情况。为了找到每个页面在内存中对应的物理块，系统为每个进程设立一张页面映射表，简称**页表**。
- 进程的所有页依次在页表中有一个页表项，其中记载了相应页面在内存中对应的物理块号。进程执行时，按照逻辑地址中的页号查找页表中对应的项，找到该页在内存中物理块号。
- 页表的作用就是实现页号到物理块号的地址映射。

§4.5 分页存储管理方式

- 页表是页式存储管理的数据结构，它包括用户程序空间的页面与内存块的对应关系、页面的存储保护和存取控制方面的信息，是OS进行分页管理的依据。

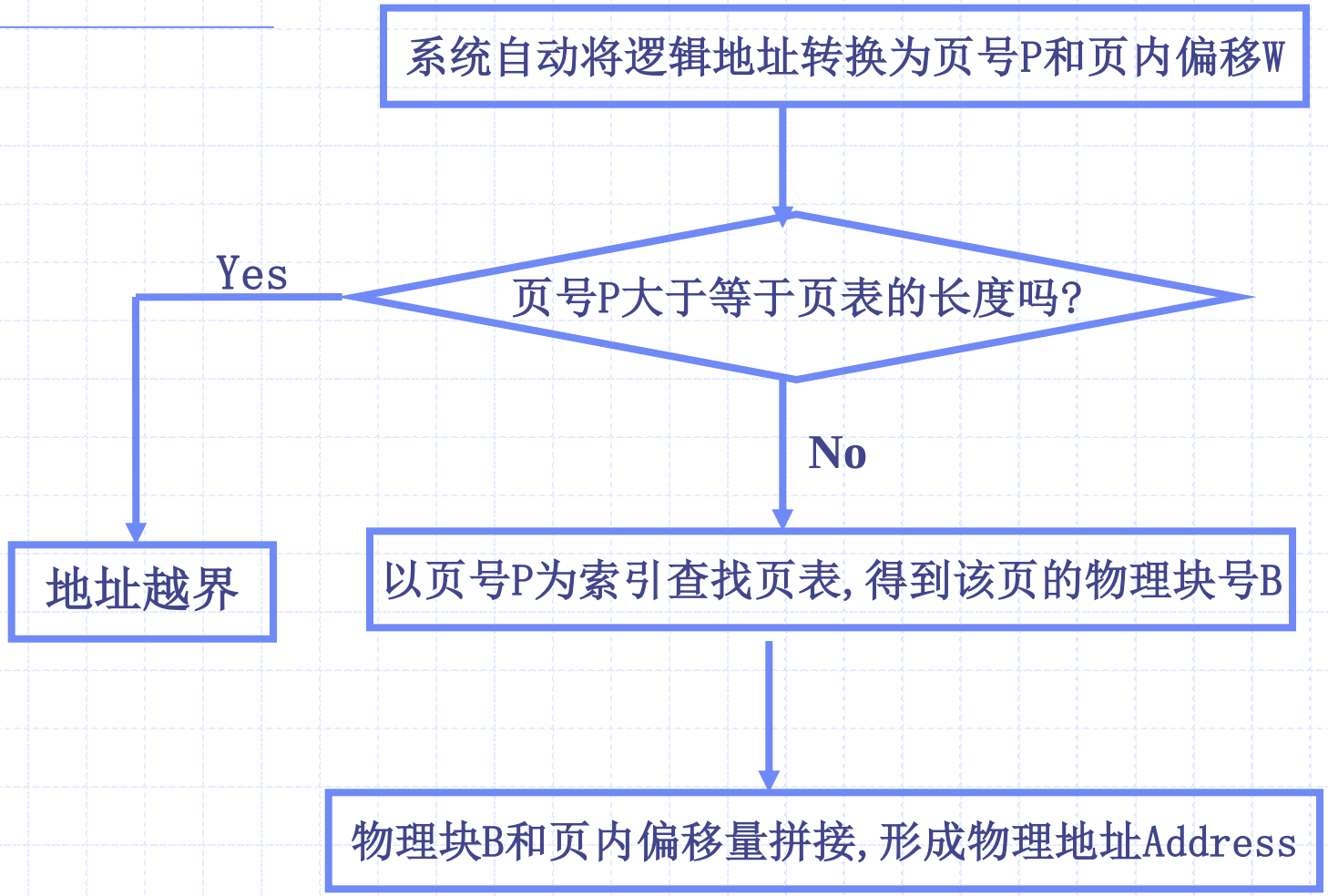


页号	块号
0	5
1	8

页表

§4.5 分页存储管理方式

4.5.2 地址映射



§4.5 分页存储管理方式

【注意】将逻辑地址线性分割求出页号P和页内位移W：

1、逻辑地址以十进制数给出：

页号P=逻辑地址 % 页大小

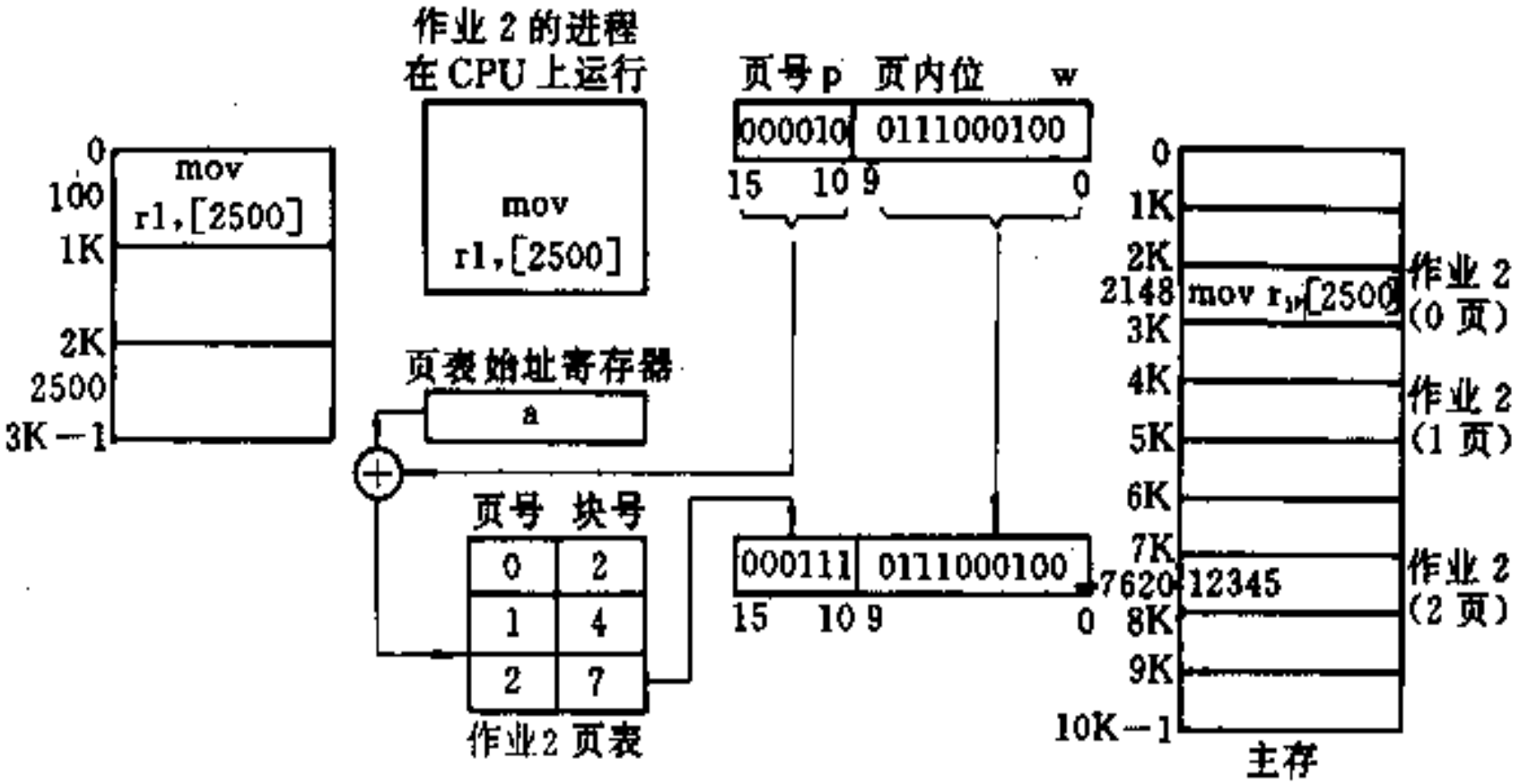
页内位移W=逻辑地址 mod 页大小

2、逻辑地址以十六进制、八进制、二进制的形式给出：

- 将逻辑地址转换成二进制；
- 按页的大小分离出页号P和位移量W（低位部分是位移量，高位部分是页号）；
- 将位移量直接复制到内存地址寄存器的低位部分；
- 以页号查页表，得到对应页装入内存的块号B，将块号转换成二进制数填入地址寄存器的高位部分，从而形成内存地址。

§4.5 分页存储管理方式

- 设页长为1K，程序地址字长为16位，用户程序空间和页表如图。



§4.5 分页存储管理方式

例题与习题：

例1：设有8页的逻辑地址空间，每页有1024个字节，它们被映射到32块的的物理存储区，那么逻辑地址的有效位是多少，物理地址至少多少位？

例2：在一分页系统中，逻辑地址的长度为16位，页面大小为4096字节，现有一逻辑地址2F6AH，且第0、1、2页依次存放在物理块5、10、11中，问相应的物理地址是多少？

例3：在某分页系统，主存的容量为64K，页面的大小为1K，对于一个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，试将十进制的逻辑地址1023、2500、3500和4500转化成物理地址。

§4.5 分页存储管理方式

快表和联想寄存器

由于页表是驻留在内存的某个固定区域中，而取数据或指令又必须经过页表变换才能得到实际物理地址。因此，取一个数据或指令至少要访问内存两次以上。一次访问页表以确定所取数据或指令的物理地址，另一次是根据地址取数据或指令，这比通常执行指令的速度慢了一倍。

解决这个问题的一种方法是把页表放在一组快速存储器中（Cache），从而加快访问内存的速度。我们把这种快速存储器组成的页表称为**快表**，把存放在内存中的页表称为慢表。快表又叫相联（联想）存储器（associative memory）。

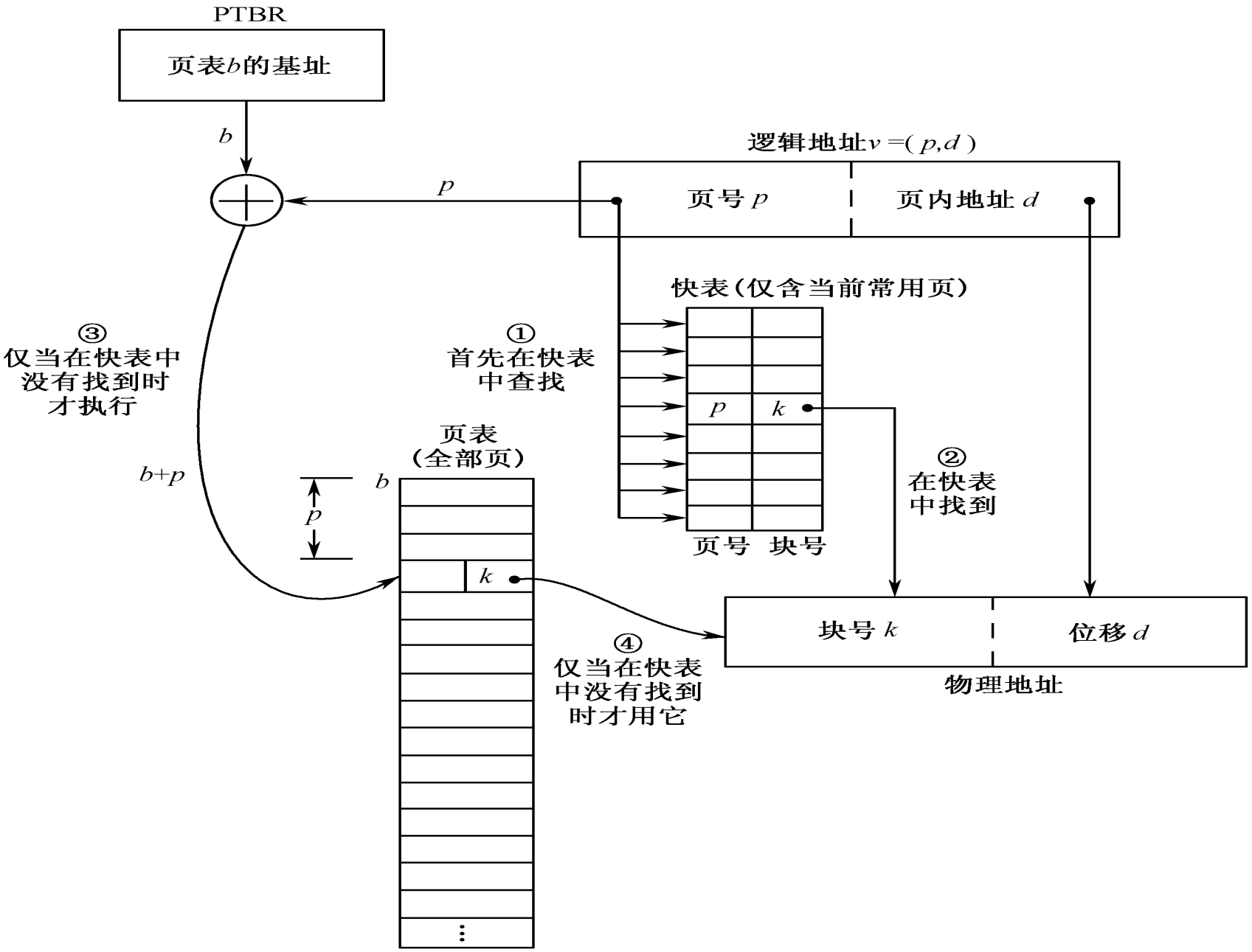
§4.5 分页存储管理方式

- **关于联想寄存器的讨论：**

一个程序可能会很大，如1M，若页长为1K，则该程序有1000个页，则该程序的页表就需要1000个表项，当程序更大时，页表会更大，那么我们应该有一个多大的快速存储器才能满足要求呢？这会遇到两个问题：

- 可能快速存储器多大都是不够的，因为程序可能会更大。
- 快速存储器是非常非常昂贵的。

实际上我们并不需要一个很大的快速存储器，有一个能存放16个页表表目的快速存储器就够了。


PTBR
页表b的基址
b
逻辑地址v =(p,d)
p
页号 p
页内地址 d
快表(仅含当前常用页)
③
仅当在快表中
没有找到时
才执行
①
首先在快表
中查找
p
k
页表
(全部页)
b+p
b
p
②
在快表
中找到
页号
块号
k
块号 k
位移 d
④
仅当在快表
中没有找到
时才用它
物理地址

§4.5 分页存储管理方式

例题：

假定访问主存时间为100毫微秒，访问相联存储器时间为20毫微秒，相联存储器为32个单元时快表命中率可达90%，按逻辑地址存取的平均时间为：

（100＋20）×90%＋(100+100+20)×(1-90%)＝130毫微秒

比两次访问主存的时间100毫微秒×2＝200毫微秒下降了四成多。

有效访问内存的时间

$$T=P_{TLB}*（T_{TLB}+T_M）+（1-P_{TLB}）*（T_{TLB}+2T_M）$$

其中,P_{TLB}为快表的命中率,T_{TLB}为快表的访问时间,T_M为内存的访问时间

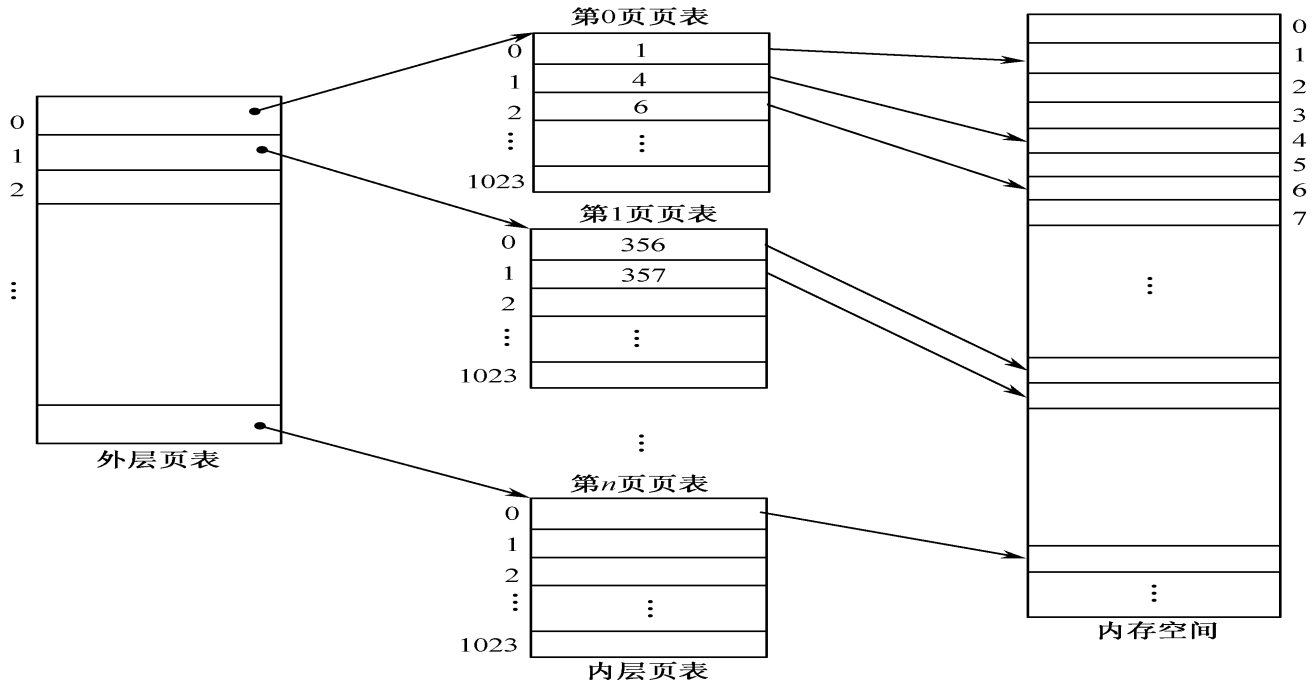
§4.5 分页存储管理方式

4.5.4 两级和多级页表

- CPU具有32位地址时，使用2^{32}逻辑地址空间的分页系统，规定页面4KB时，每个进程页表的表项有1兆(2^{20})个，若表项占用4个字节，则每个进程需要占用4MB连续内存空间存放页表。
- 多级页表概念：页表和页面一样也进行分页，内存仅存放当前使用的页表，暂时不用部分放在磁盘上，待用到时再行调进。
- 具体做法：把整个页表进行分页，分成一张张小页表(称为页表页），小页表的大小与页框相同，为进行索引查找，应该为这些小页表建一张页目录表，其表项指出小页表所在页框号及相关信息。

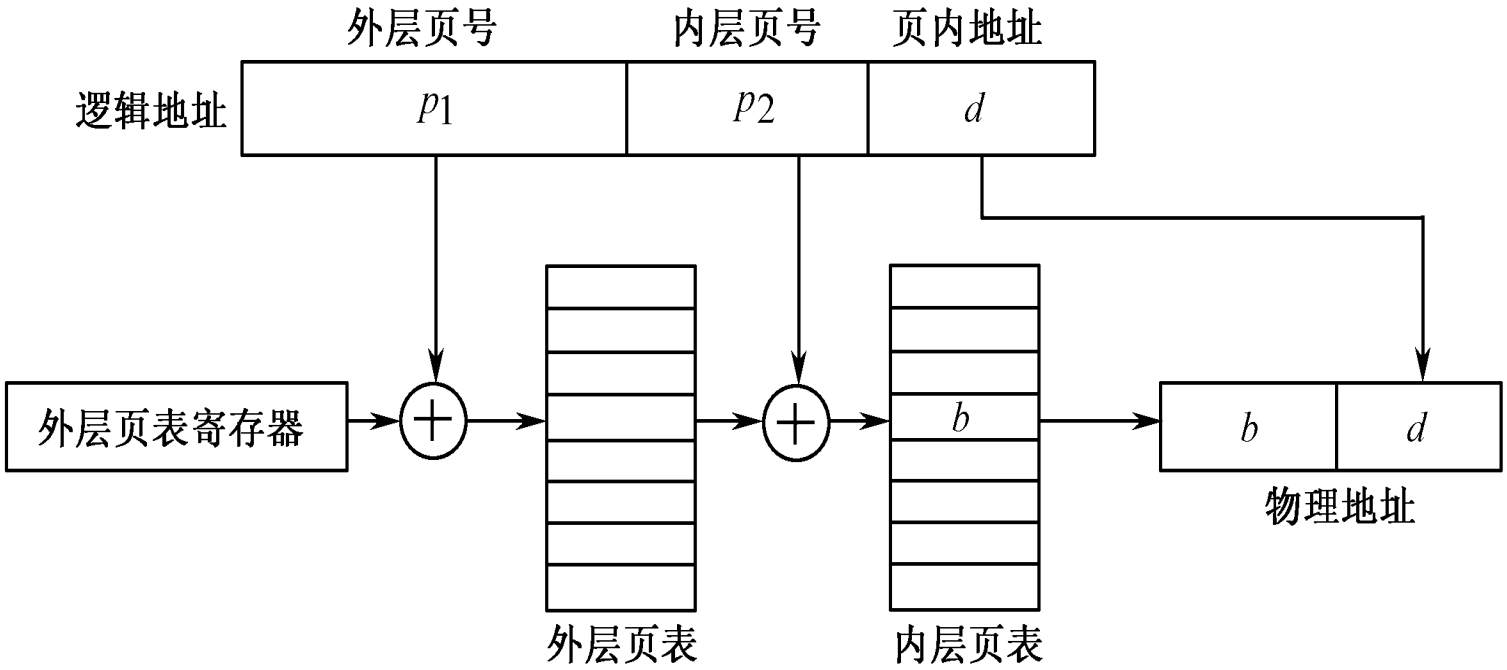
§4.5 分页存储管理方式

- 系统为每个进程建一张页目录表，它的每个表项对应一个页表页，而页表页的每个表项给出了页面和页框的对应关系，页目录表是一级页表，页表页是二级页表。
- 逻辑地址结构有三部分组成：页目录、页表页和位移。



§4.5 分页存储管理方式

- 在具有两级页表结构的系统中，地址转换的方法是：利用外层页号p1检索外层页表，从中找到相应内层页表的基址，再利用p2作为该内层页表的索引，找到该页面在内存的块号，用该块号和页内地址d拼接起来形成访问物块内存的物理地址。



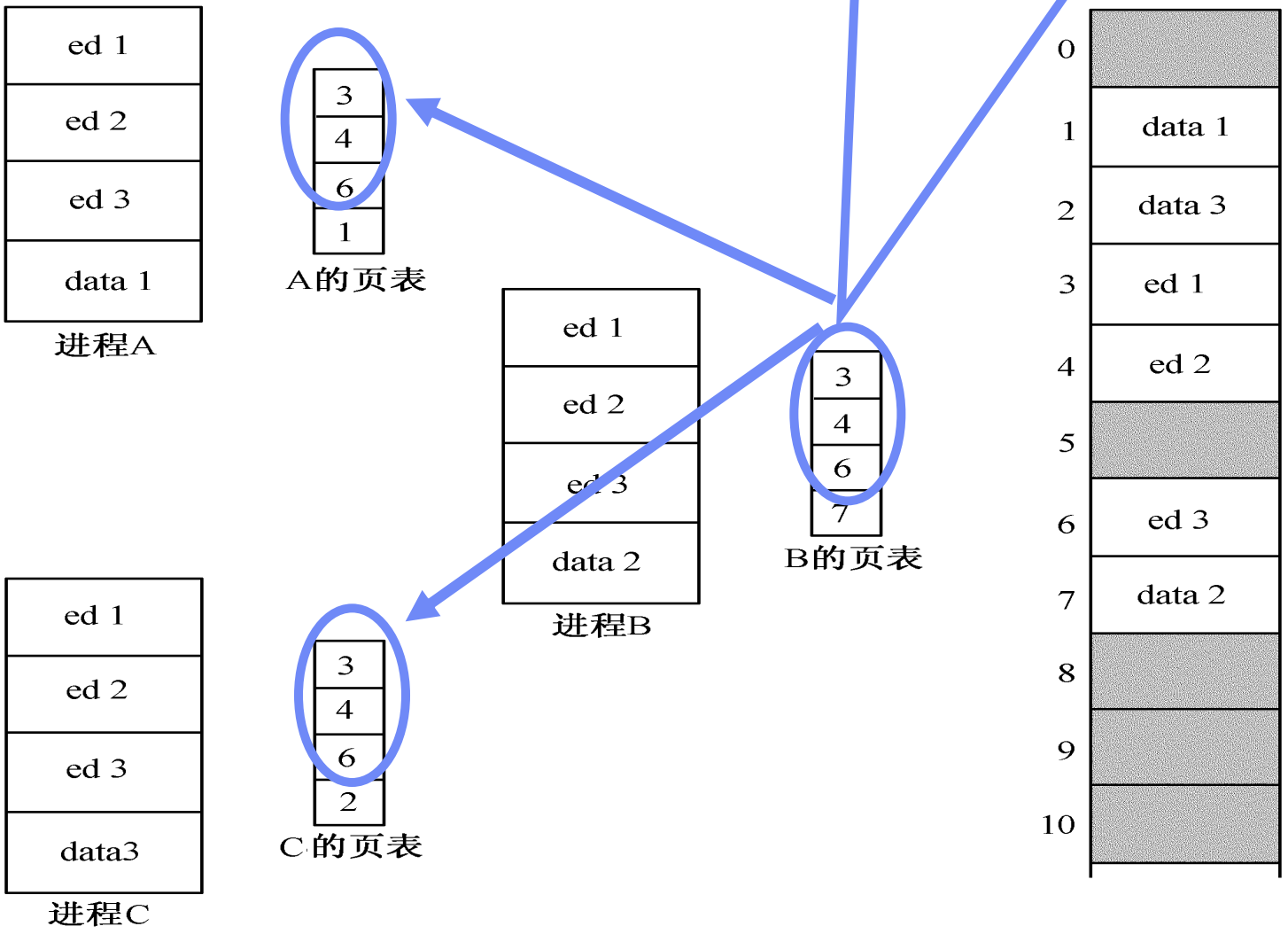
§4.5 分页存储管理方式

4.5.5 页面的共享

- 设想一下这样的系统，有40个用户，每个用户都执行一个文本编辑器。如果文本编辑器有150KB代码段和50KB数据段，需要8000KB来支持40个用户。
- 如果代码是可重入代码，那么就可以共享。可重入代码（或纯代码）是在其执行过程中本身不做任何修改的代码，通常由指令和常数组成。
- 共享页面时只需要在物理内存中保存一个编辑器的拷贝。每个用户的页表映射到编辑器的同一物理拷贝，而数据页映射到不同的帧。

考研试题：（南航）举例说明在分页系统如何实现内存共享？要求图示说明。

分页系统中，内存共享是通过页表指向相同的物理块的方式来实现共享的。



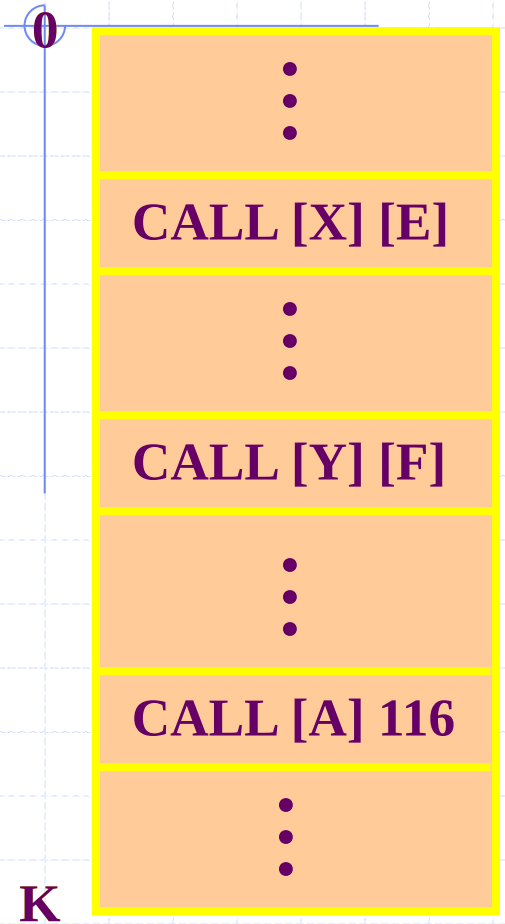
§4.6 分段存储管理方式

4.6.1 分段存储管理的基本原理

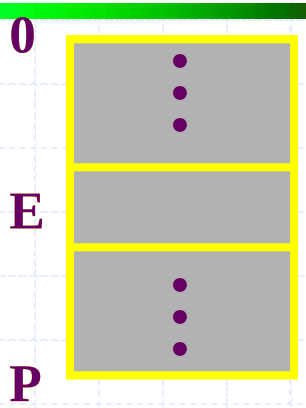
- 用户程序划分：按程序自身的逻辑关系划分为若干个程序段，每个程序段都有一个段名，且有一个段号。段号从0开始，每一段段内也从0开始编址，段内地址是连续的。
- 逻辑地址：

- 内存划分：内存空间被动态的划分为若干个长度不相同的区域，称为物理段，每个物理段由起始地址和长度确定。
- 内存分配：以段为单位分配内存，每一个段在内存中占据连续空间（内存随机分割，需要多少分配多少），但各段之间可以不连续存放。

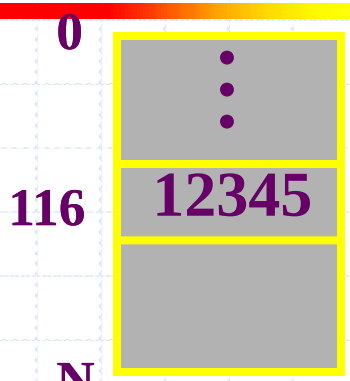
§4.6 分段存储管理方式



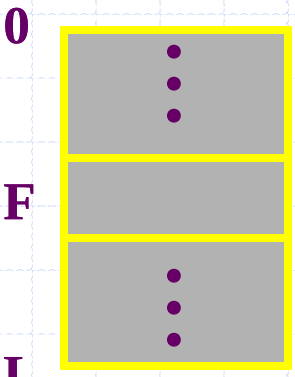
主程序段[M]



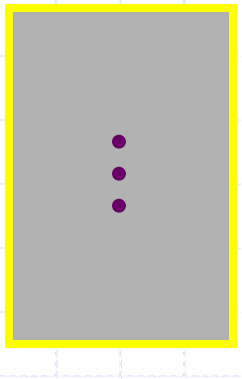
子程序段[X]



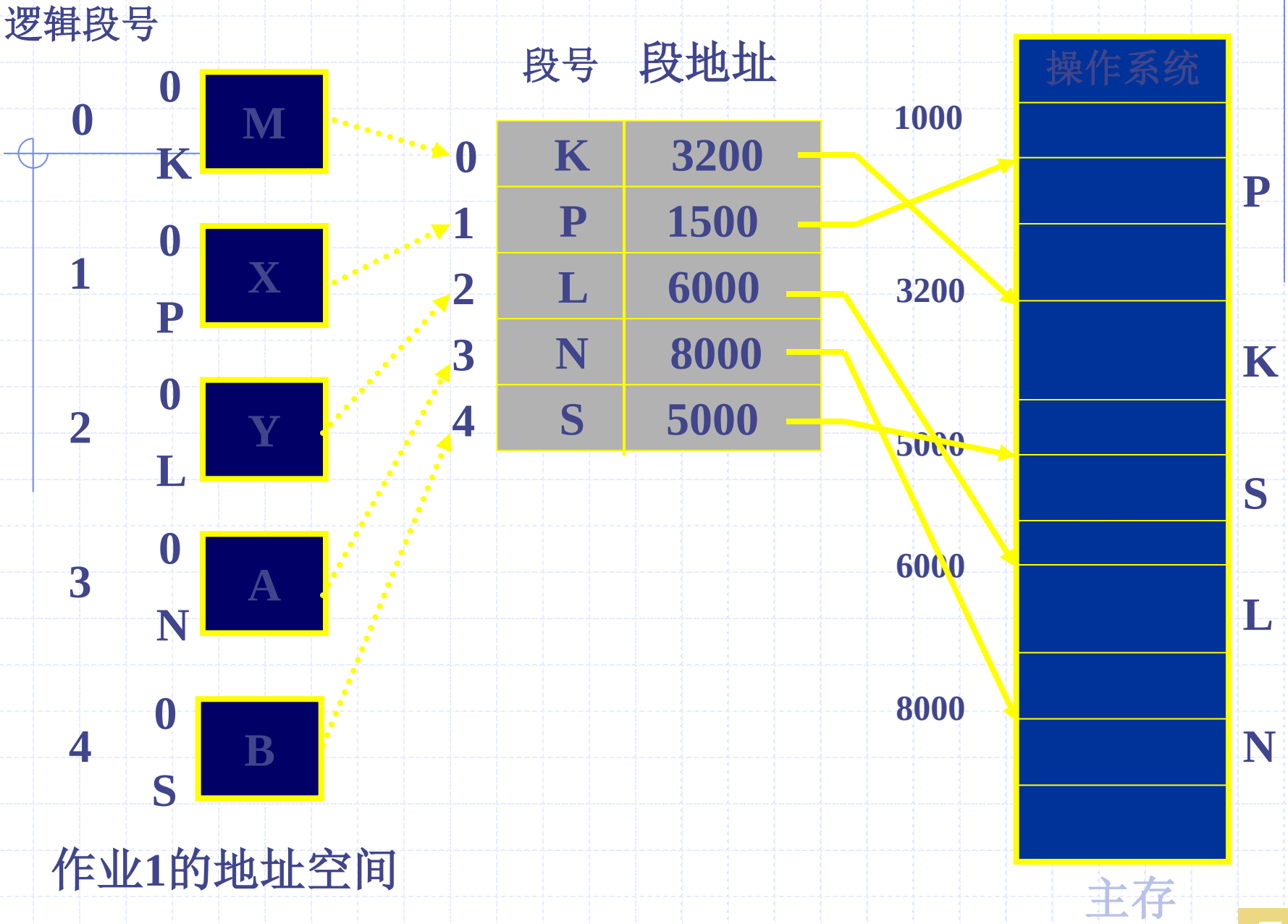
数组[A]



子程序段[Y]



工作区段[B]


逻辑段号
0
0
K
M
1
0
P
X
2
0
L
Y
3
0
N
A
4
0
S
B
段号
段地址
0
K
3200
1
P
1500
2
L
6000
3
N
8000
4
S
5000
1000
3200
5000
6000
8000
操作系统
P
K
S
L
N
主存
作业1的地址空间

§ 4.6 分段存储管理方式

4.6.2 分段存储管理的地址映射

1. 地址映射的数据结构

- 在分段存储方式下，作业的各个段 被分配到了内存中相互不连续的存储区域。各段的长度不同，占用的内存区域也大小不一。
- 每个作业在内存中设置了一个段映射表，简称段表。段表保存在内存中某个固定的区域，表中记录着一个作业各段的大小和首址。在某些操作系统中把段表的表项称为段描述符。

2. 内存的分配

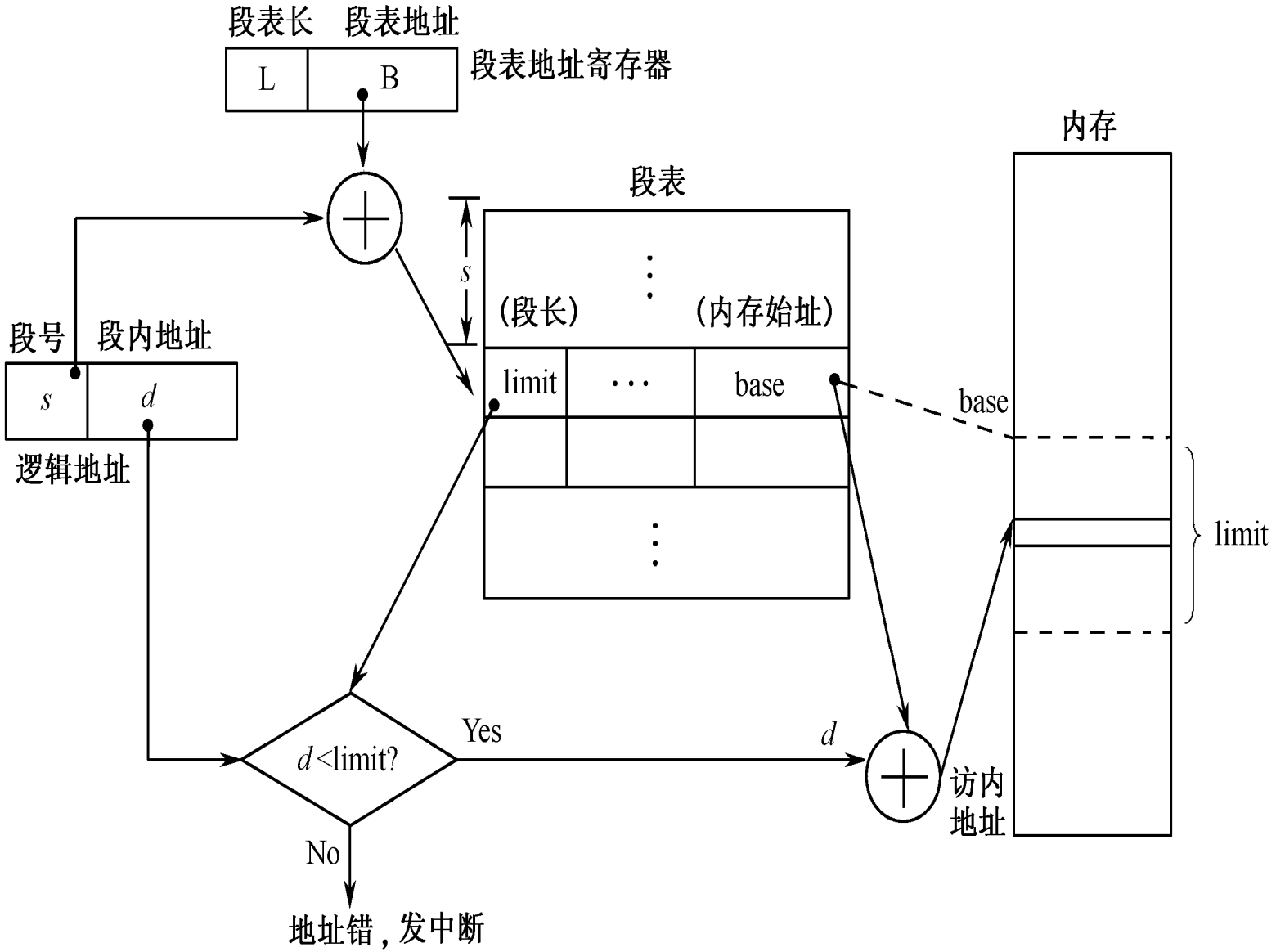
分段系统的内存分配策略和分区管理相同。

§4.6 分段存储管理方式

3. 地址映射

- 一个逻辑地址由两部分组成：段号s和段内地址d。系统根据段表地址寄存器的内容（表示段表的起始地址）找到进程的段表，以段号为索引查找相应的表项，得出该段的长度limit及该段在内存的起始地址base。
- 将段内地址d与段长limit进行比较。如果d不小于limit，这表示地址越界，系统发出地址越界中断，终止程序的执行；如果d小于limit，则表示地址合法，将段内地址d与该段的内存始址base相加，得到所要访问单元的内存地址。

§4.6 分段存储管理方式


段表长
段表地址
L
B
段表地址寄存器
内存
段表
s
(段长)
(内存始址)
limit
base
段号
段内地址
s
d
逻辑地址
base
limit
d <limit?
Yes
d
访内
地址
No
地址错，发中断

§4.6 分段存储管理方式

- 例题：在一分段存储系统中，其段表如下：

段号	内存起始地址	段长
0	210	500
1	2350	20
2	100	90
3	1350	590
4	1938	95

试求下列逻辑地址对应的物理地址是什么？

（1）[0,430]；（2）[1,10]；（3）[2,500]；

（4）[3,400]；（5）[4,112]；（6）[5,32]

§4.6 分段存储管理方式

分段和分页的比较：

（1）页是信息的物理单位，而段是信息的逻辑单位。分页时为了实现离散分配方式，以减少内存碎片，提高内存利用率。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。段则是信息的逻辑单位，它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）页的大小是由系统确定的，由系统把逻辑地址划分成页号和页内地址两部分，整个系统只能有一种大小的页面；而段的长度却不固定，决定于用户的程序。通常由编译程序在对源码进行编译时，根据信息的性质来划分。

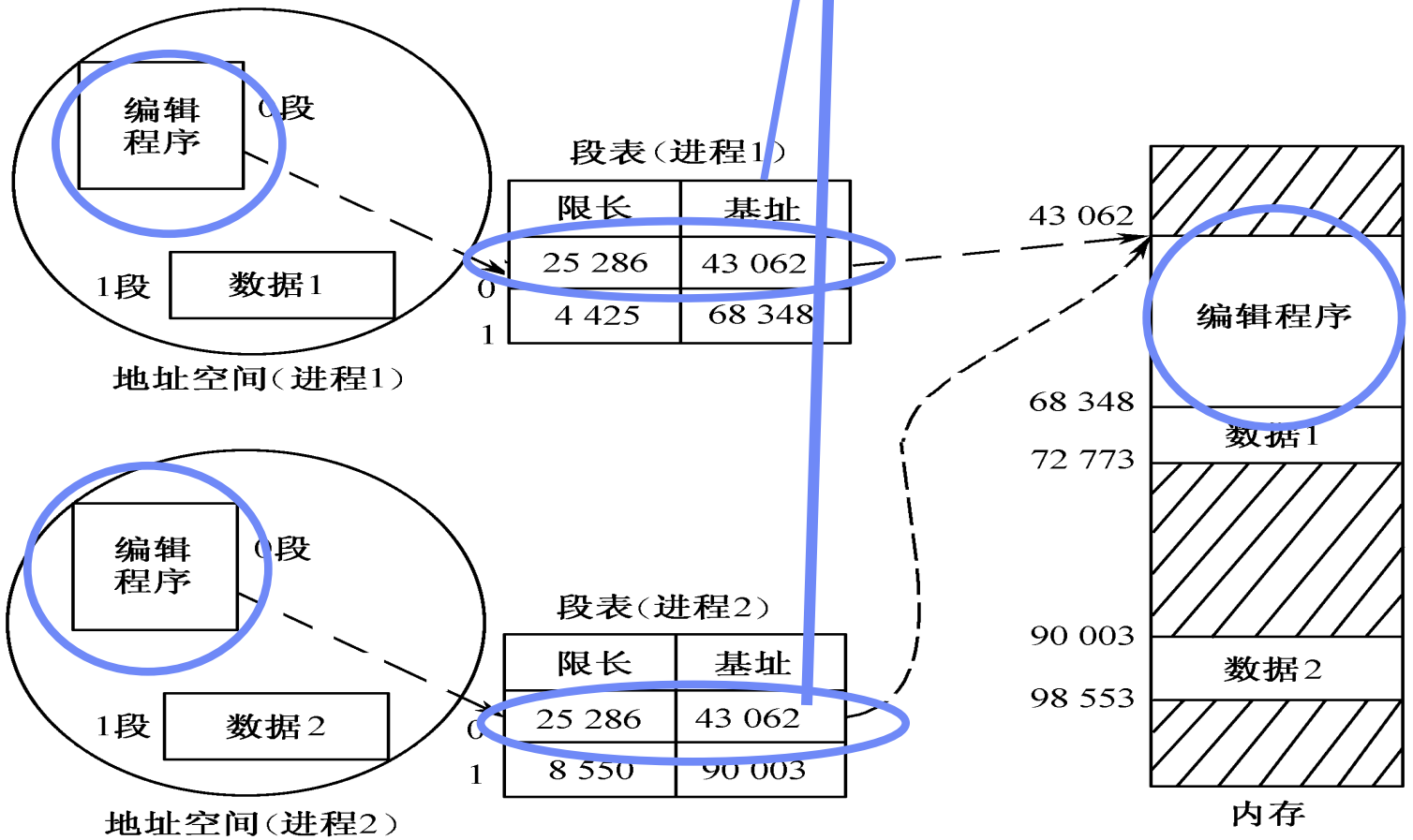
（3）分页的进程地址空间是一维的，即单一的线性空间；而分段的进程地址空间是二维的，有段号和段内地址两部分组成。

§4.6 分段存储管理方式

4.6.3 段的共享和保护

1. 段的共享

段的共享是通过不同进程段表中的项指向同一基址来实现的。



§4.6 分段存储管理方式

2. 段的保护

- 地址越界保护法。地址越界保护则是利用段表中的段长项与虚拟地址中的段内相对地址比较进行的。若段内相对地址大于段长，系统就会产生保护中断。不过，在允许段动态增长的系统中，段内相对地址大于段长是允许的。为此，段表中设置相应的增补位以指示该段是否允许该段动态增长。
- 存取方式控制保护法。存取控制保护是通过在段表中增加相应的访问权限位，用来记录对本段的存取控制方式，如可读、可写、可执行等。在程序执行时，存储映射硬件对段表中的保护信息进行检验，防止对信息进行非法存取。

§4.5 分段存储管理

4.4.4 段页式存储管理

1. 基本思想：

（1）作业地址空间进行段式管理。

（2）每段内再分成若干大小固定的页，每段都从零开始为自己的各页依次编写连续的页号。

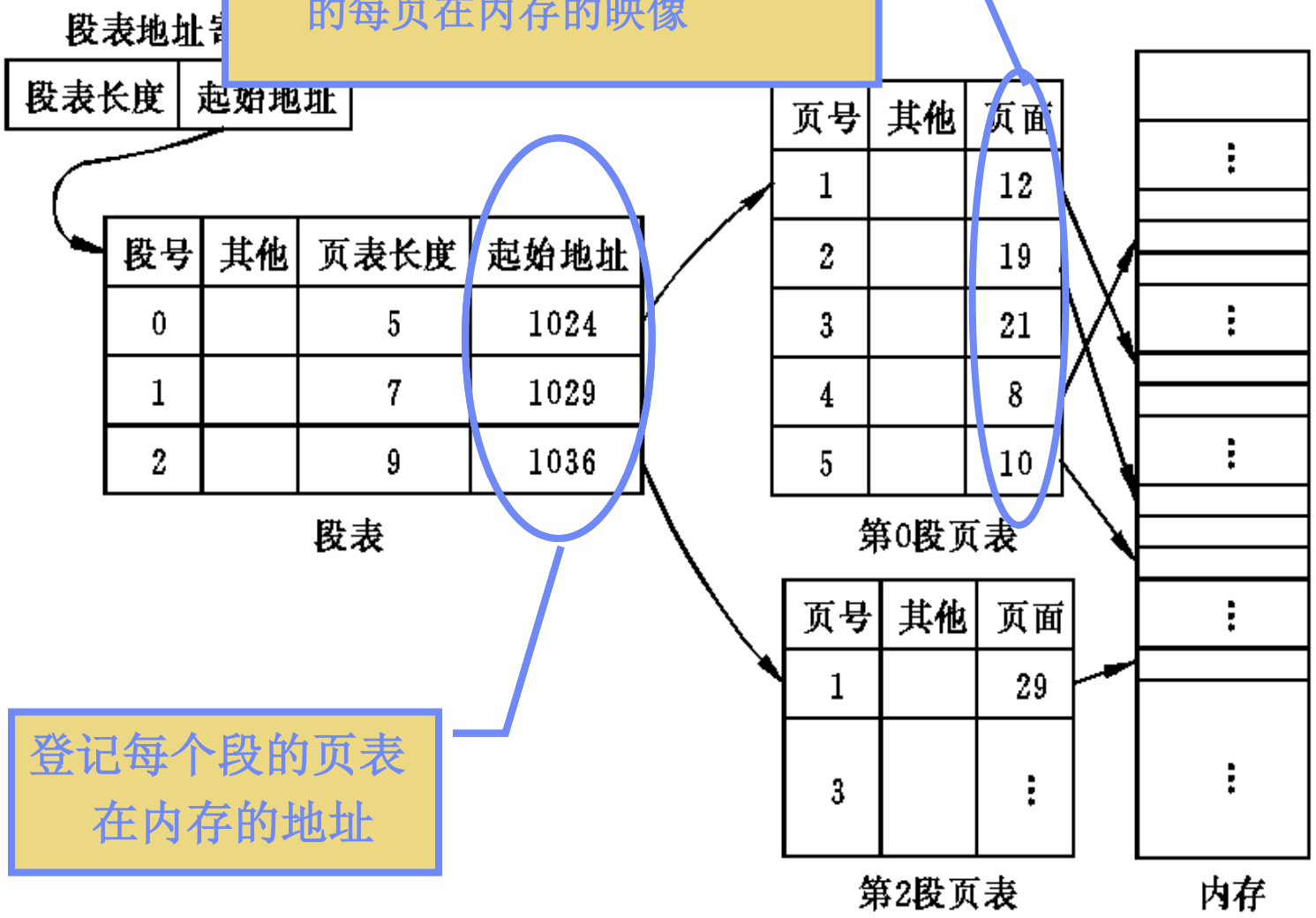
（3）对内存空间的管理仍然和分页存储管理一样，将其分成若干个和页面大小相同的物理块。

（4）作业的逻辑地址包括3个部分：段号、页号和页内位移。

（5）为实现地址变换，段页式系统设立了段表和页表。

§4.5 分段存储管理

每个段有一个页表，其中登记该段的每页在内存的映像



§4.5 分段存储管理

2. 地址转换过程: