第三章 存储器管理

主讲人：毛启容

Email:mao_qr@ujs.edu.cn

江苏大学计算机科学与通信工程学院

概　述

- 主存储器（又称内部存储器、内存、主存）的管理一直是操作系统最主要的功能之一。
- 在现代计算机系统中，尽管主存容量已经很大，价格已相当便宜，但主存储器依然是四大硬件资源中最关键、最紧张的“瓶颈”资源。
- 因此，能否合理、有效地使用主存储器，在很大程度上反映了操作系统的性能，并直接影响到整个计算机系统作用的发挥。

存储管理

3.1 存储管理的功能
3.2 存储管理的几种形式与重定位
3.3 单道环境下的存储管理
3.4 分区存储管理
3.5 页式存储管理

3.1 存储管理的功能

- 计算机系统的多级存储结构

大多数计算机把存储器分为三级:

1、外部存储器（外存或辅存）

用来存放不立即使用的程序和数据，当用户的程序运行需要它们时，再从外存把它们读入到主存储器。

2、主存储器（主存或内存）

程序的运行总是存放在主存中，以便处理器的访问。

3、高速缓冲存储器

处理机取指令和存取数据就在高速缓冲存储器中进行。

多级存储之间的关系

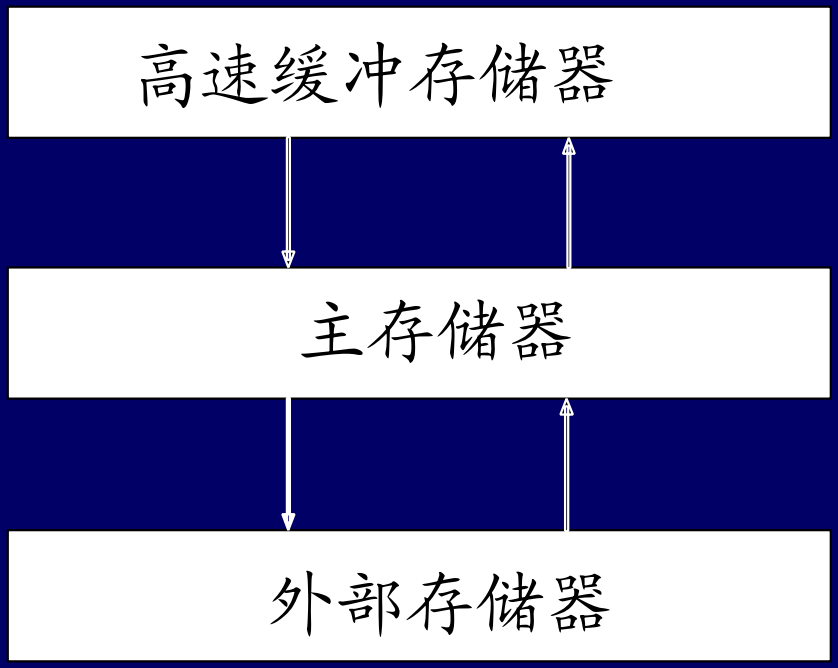


图3-1 多级存储结构

速度越来越快，
容量越来越小，
价格越来越昂贵。

本章主要介绍主存储器空间的管理原理和实现技术，且主要针对主存的用户区。

存储管理的功能

1. 存储空间的分配和回收

- 记住每个存储区域的状态（是否使用—主存分配记录表）
- 实施分配
- 接收系统或用户释放的存储区域，并相应地修改主存分配记录表

2. 地址映射和重定位：程序地址空间中的逻辑地址转换为主存空间中对应的物理地址 .

存储管理的功能（续）：

3. 存储共享与保护

存储共享两种含义：

- 共同使用存储空间，各自使用不同的存储区域；
- 共同使用主存中的某些程序和数据区——共享区。保护各存储区中的信息不被破坏和偷窃及共享区信息的完整性和一致性 。

4. 主存扩充：主存单元逻辑上的扩充 。

存储管理

3.1 存储管理的功能

3.2 存储管理的几种形式与重定位

3.3 单道环境下的存储管理

3.4 分区存储管理

3.5 页式存储管理

3.2 存储分配的几种形式与重定位

解决的问题:多道程序之间如何共享主存的存储空间.

3.2.1 存储分配的几种形式

3.2.2 重定位

3.2.1 存储分配的几种形式

1. 直接存储分配方式

程序员在编写程序或编译程序对源程序编译时采用内存物理地址。

2. 静态存储分配方式

在将作业装入内存时才确定它们在内存中的位置。运行中固定不变。

3. 动态存储分配方式

不是一次性将程序全部装入主存，而是根据执行需要动态装入和回收，也可动态申请存储空间。

3.2 存储分配的几种形式与重定位

解决的问题:多道程序之间如何共享主存的存储空间.

3.2.1 存储分配的几种形式

3.2.2 重定位

3.2.2 重定位

目的：为了实现静态/动态地址分配，必须将逻辑地址和物理地址分开，并将逻辑地址定位为物理地址.

- 地址空间和存储空间
- 重定位的概念

3.2.2 重定位

- 地址空间和存储空间
  - 相对地址：编译系统总是从零号地址单元开始，为目标程序指令顺序分配地址。这些地址被称为相对地址。
  - 逻辑地址空间：相对地址的集合。
  - 物理（绝对）地址：物理单元的编号。
  - 存储空间：主存中一系列存储信息的物理单元的集合。

3.2.2 重定位

- 重定位的概念

  - 重定位：相对地址转化为存储空间中的绝对地址的地址变换过程，称为地址重定位，也称地址映射。
  - 地址重定位的方式：静态地址重定位和动态地址重定位

静态地址重定位：是指用户程序在装入时由装配程序一次完成，即地址变换只是在装入时一次完成，以后不再改变 。

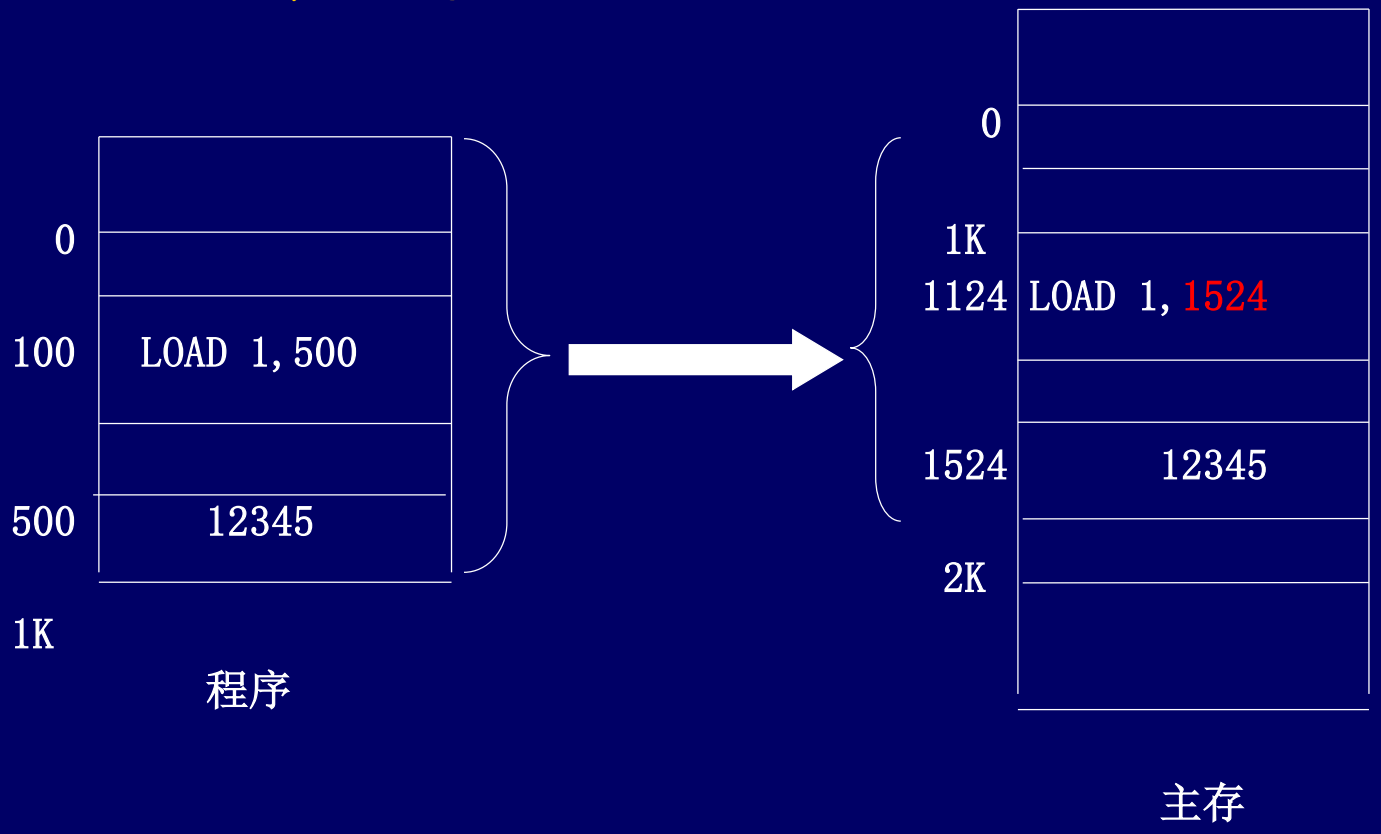


图 3-2　程序由地址空间装入存储空间

动态地址重定位

动态地址重定位：在程序执行的过程中，当CPU要对存储器进行访问时，通过硬件地址变换机构，将要访问的程序和数据地址转换成主存地址。

其具体过程如下：

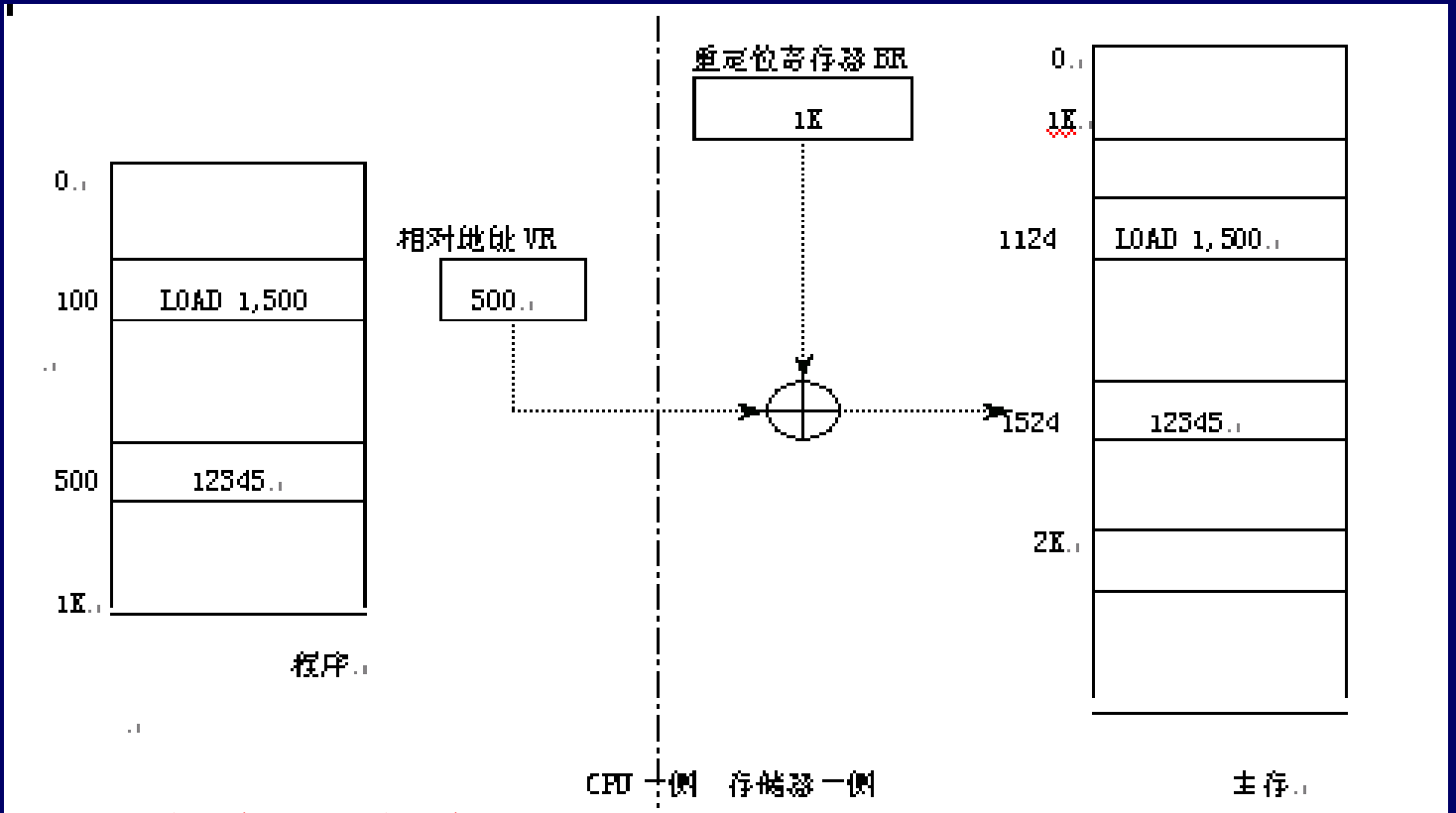
（1）设置重定位寄存器BR，相对地址寄存器VR。

（2）将程序段装入主存，且将其占用的主存区起始地址送入BR中，如：（BR）＝1K。

（3）在程序执行过程中，将所要访问的相对地址送入VR中，如：（VR）＝500。

(4)地址变换机构把VR和BR的内容相加，得到实际访问的物理地址。

动态地址重定位



MA＝（BR）＋（VR）

图 3-3　动态重定位过程

3.2.2 重定位

1. 静态地址重定位的缺点：

1) 用户程序必须分配一个连续的存储空间。

2) 难以实现程序和数据的共享。

2. 动态地址重定位的优点是：

1) 有利于提高主存的利用率和存储空间使用的灵活性。

2) 有利于程序段的共享实现。

3) 为实现虚拟存储管理提供了基础。

3. 动态地址重定位的缺点是：

1) 实现存储器管理的软件比较复杂。

2) 需要附加的硬件支持。

存储管理

3.1 存储管理的功能
3.2 存储管理的几种形式与重定位
3.3 单道环境下的存储管理
3.4 分区存储管理
3.5 页式存储管理

3.3 单道环境下的存储管理

- 单道环境下的存储管理:

  --任一时刻主存只有一个用户程序，且该程序占有的用户区域是连续的。如果系统资源能够满足用户程序要求，则系统分配主存资源给该用户程序，否则系统无法执行该程序，同时给出相应的提示信息。

- 单道环境下，一般采用单一连续存储管理。
- 单一连续存储管理方式是静态存储分配方式。

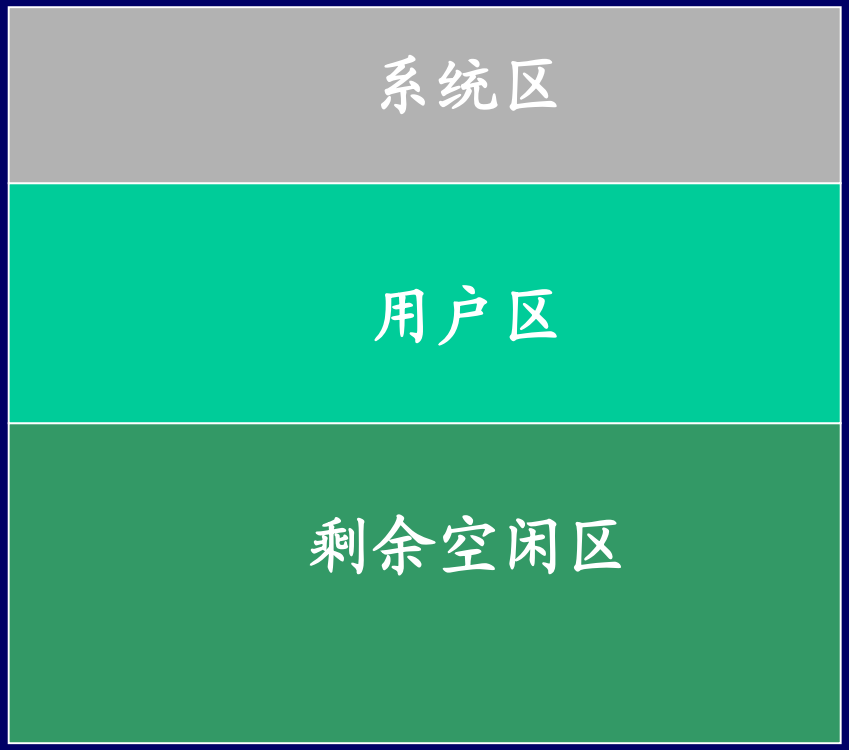


图3-4　单一连续区管理主存分配

单一连续存储管理的算法

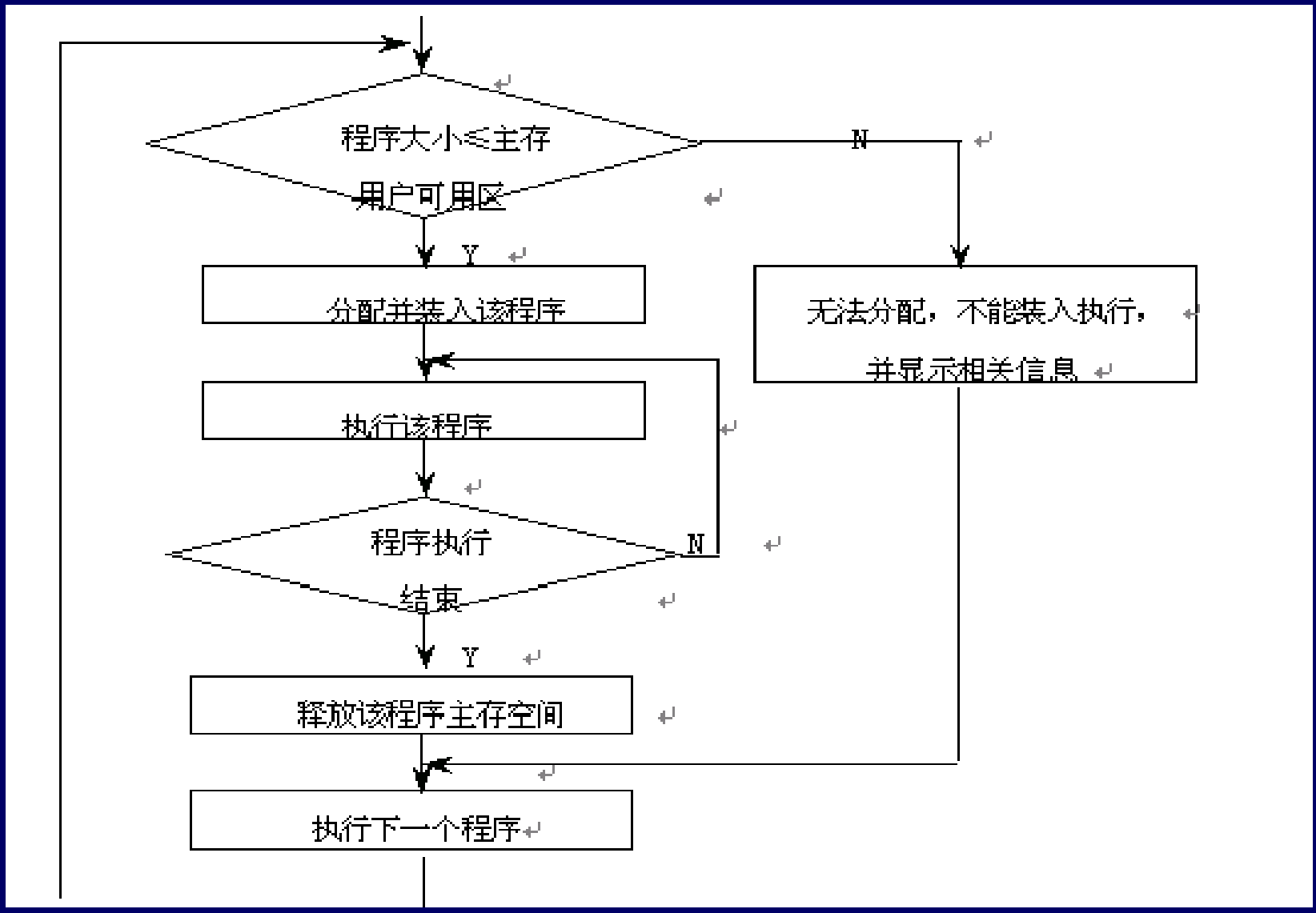


图3-5 单一连续区主存分配与回收

3.3 单道环境下的存储管理

优点：管理简单，只需很少的软硬件支持。

缺点：

(1) 系统的存储空间浪费较大。

(2) 当正在执行的程序出现等待资源，处理器就处于空闲状态。

(3) 主存中的程序和数据不能被共享。

(4) 系统的外围设备也只有一个程序使用，因此外部设备利用率低。

存储管理

3.1 存储管理的功能
3.2 存储管理的几种形式与重定位
3.3 单道环境下的存储管理
3.4 分区存储管理
3.5 页式存储管理

3.4 分区存储管理

目的：将主存的用户区分为若干个区域，从而实现多道程序设计环境下各并发进程共享存储空间

根据分区的时机不同分为：

3.4.1 固定分区法

3.4.2 动态分区法

3.4.1 固定分区法

- 思想：固定分区是指系统在初始化时，将主存空间划分为若干个固定大小的区域。用户程序在执行过程中，不允许改变划分区域的大小，只能够根据各自的要求，由系统分配一个存储区域。
- 下图表示在某一时刻，进程A和B分别被分配到2和3两个分区中，第1和4分区尚未分配。

3.4.1 固定分区法

表 3-1　分区说明表

分区号	大　小	起　始	状　态
1	16KB	20KB	未分配
2	64KB	36KB	已分配
3	80KB	100KB	已分配
4	128B	180KB	未分配

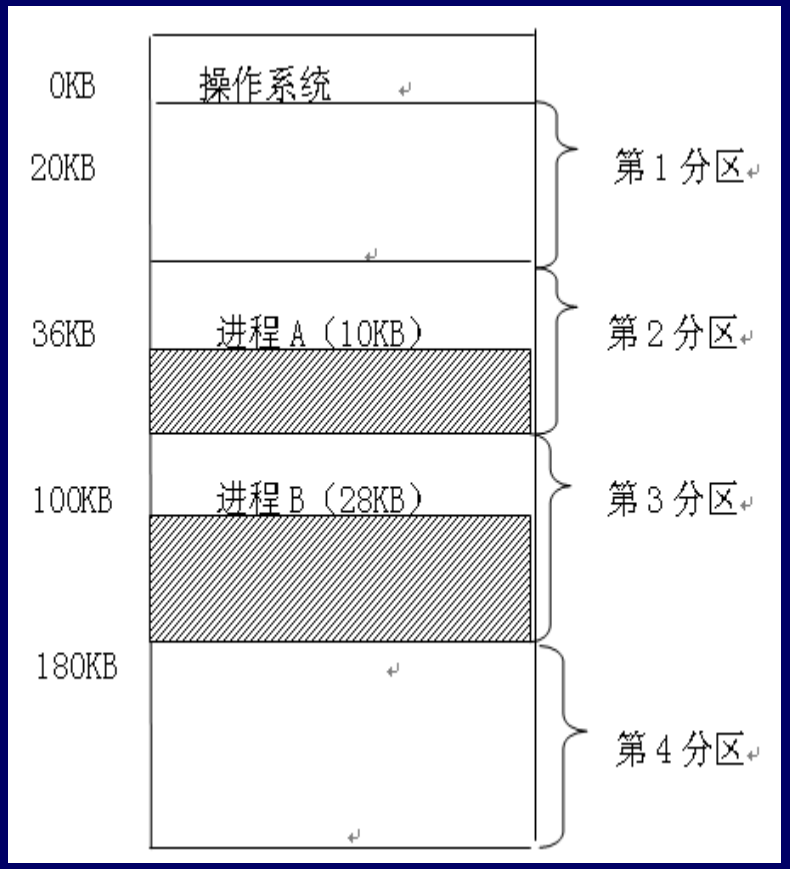
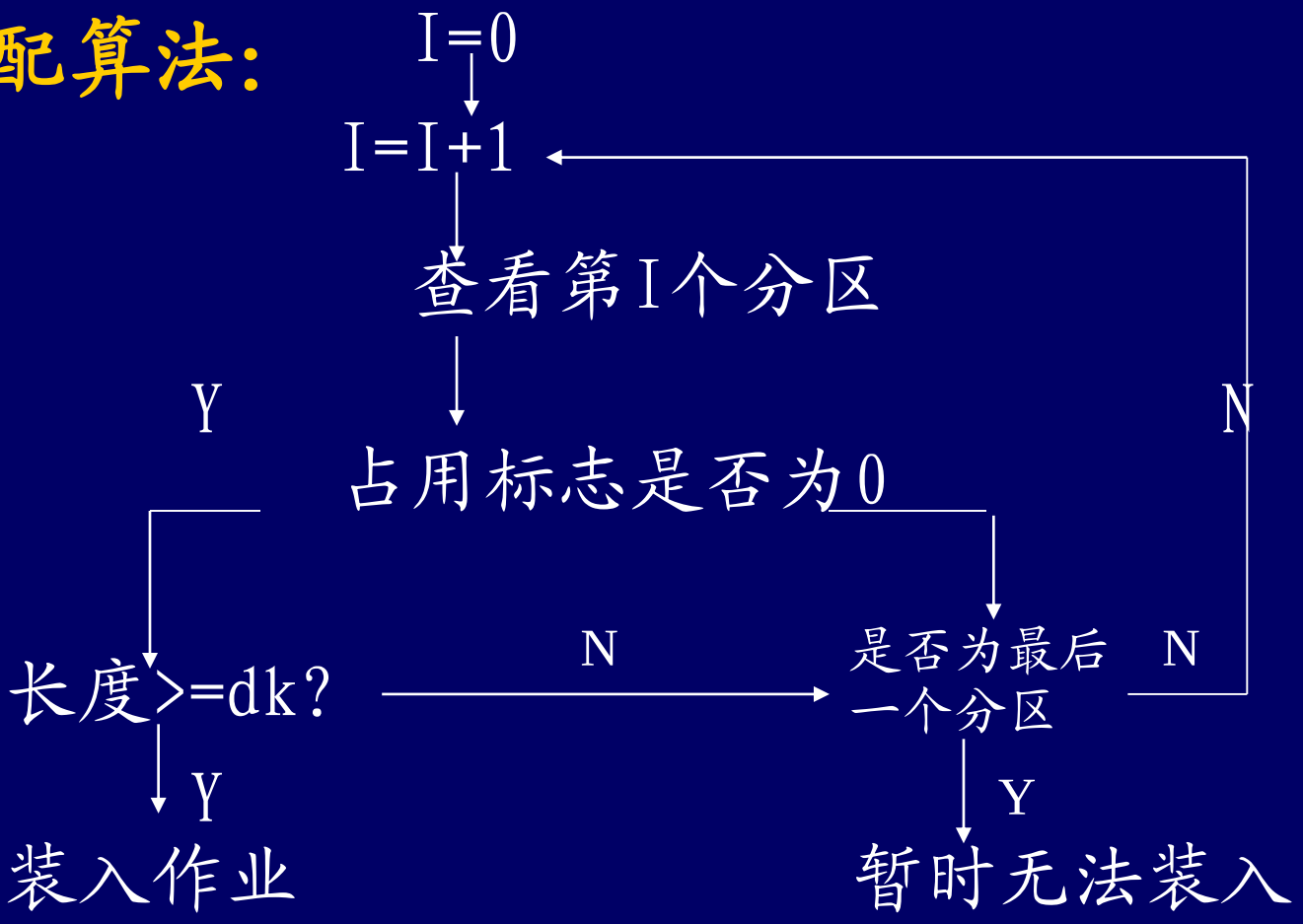


图 3-6　固定分区的存储分配存储空间分配情况

3.4.1 固定分区法

内存分配算法：



3.4.1 固定分区法

固定分区的优缺点：

优点：简单。

缺点：虽然可以使多个作业在同一时刻共享存储区，但它不能充分利用存储器资源。

“碎片”或“内零头”：在已分配的分区中，通常都有一部分未被进程占用而浪费的主存空间，这一部分空间称作为存储器的“碎片”或“内零头”。

3.4 分区存储管理

3.4.1 固定分区法

3.4.2 动态分区法

3.4.2 动态分区法

1. 动态分区的基本概念
2. 动态分区的分配方式
3. 动态分区的回收
4. 地址转换与存储保护
5. 分区的共享
6. 移动技术
7. 分区存储管理的优点

1. 动态分区的基本概念

- 在系统初启时，除了操作系统常驻主存部分以外，只存在一个空闲分区。随后，分配程序将该区依次划分给调度程序选中的进程，并且分配的大小可随用户进程对主存的要求而改变，这种分配方式不会产生“碎片”现象，从而大大提高了主存的利用率。

1. 动态分区的基本概念

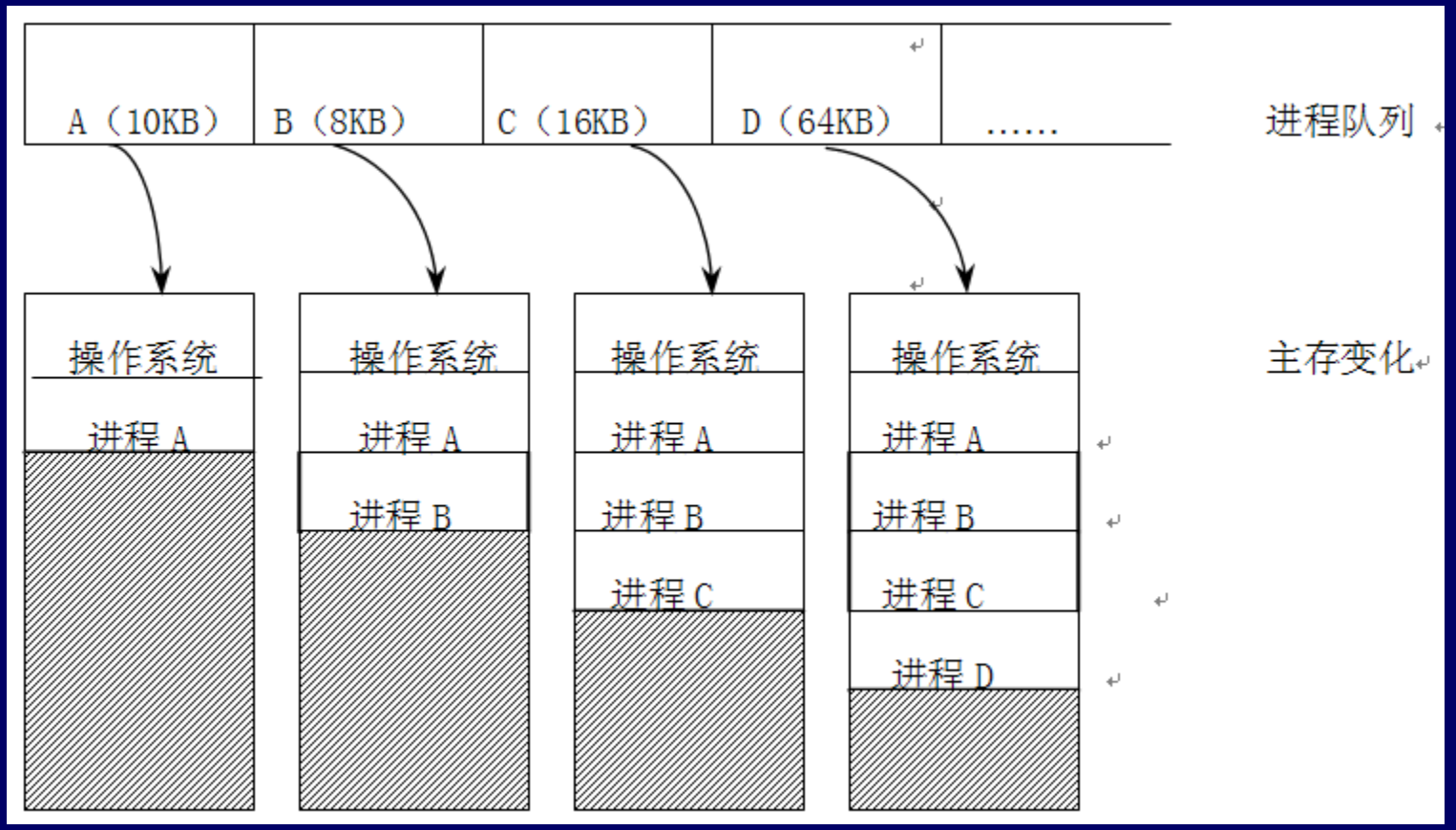


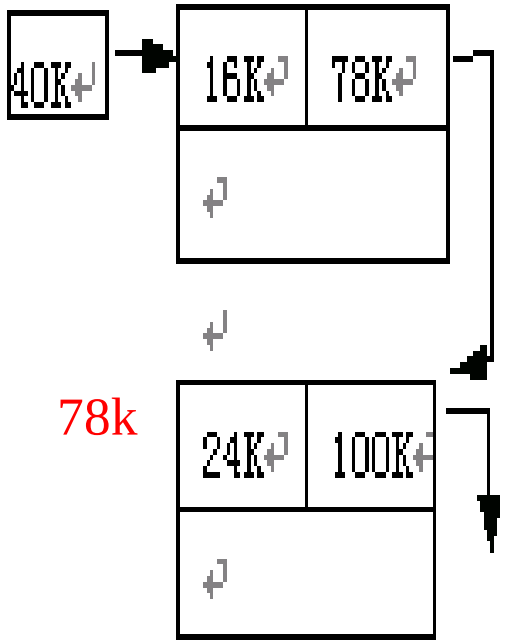
图 3-7 主存分配情况

采用数据结构

- 可用分区表：每个表目记录一个空闲区，其主要参数由区号、分区长度和起始地址组成。采用表格结构来管理空闲区比较直观，管理算法也简单，但表格的大小难以确定。
- 自由链表：利用每个空闲区的开始几个存储单元来存放本空闲区的大小及下一个空闲区的起始地址，从而将所有的空闲区都链接起来。然后，系统再设置一个自由链表首指针，让其指向第一个空闲区。
- 请求表：每个表目登记着请求主存资源的作业或进程号以及所需的主存大小。

区号	分区长度	起始地址
1	16K	40K
3	24K	78K
5	9K	100K

(a) 可用分区



(b) 自由链表

作业(进程)号	请求长度
P_1	15K
P_2	25K

(c) 请求表

图 3-9 可用分区表、自由链表和请求表

2. 动态分区的分配方式

动态分区的存储分配方式：是指如何从可用分区表或自由链表中寻找满足条件的空闲区分配给相应的作业。通常有三种方式：最先适应法、最佳适应法、最坏适应法。

（1）最先适应法：将作业分配到主存的第一个足够装入它的可用空闲区中。

- 缺点：可能将大的空闲区分割成一个小区，不利于大作业的装入与运行。
- 改进：将空闲区按从小到大排列。

2. 动态分区的分配方式

（2）最佳适应法：将作业分配到主存中与它所需大小最接近的一个可用空闲区中。

- 优点：可保证不会去分割一个更大的空闲区，便于今后大作业的装入运行。
- 缺点：由于空闲区通常不可能正好和作业所要求的大小相等，往往分割后剩下的空闲区非常小，以至几乎无法使用。造成了主存空间的浪费。

2. 动态分区的分配方式

（3）最坏适应法：把一个作业分配到主存中最大的空闲区中。

- 优点：在大空闲区中装入作业后，剩下的空闲区常常也很大，于是也能满足以后较大的作业的要求。该算法对中、小作业的运行是很有利的。

3. 动态分区的回收

在每一种存储分配方案中，都包含一定程度的浪费。

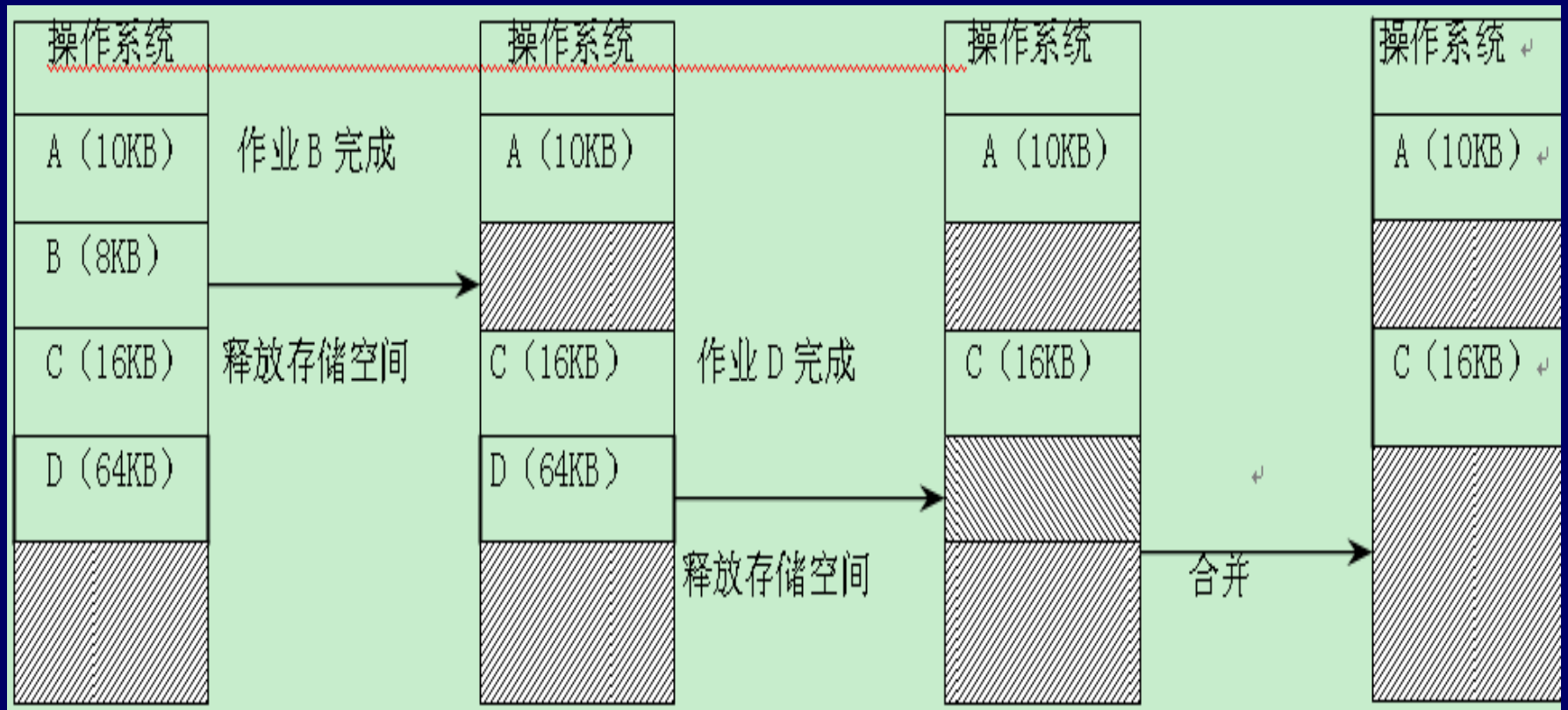


图 3-10　动态分区方式中存储区的释放和回收

3. 动态分区的回收

为了避免浪费现象，系统提供了相应的回收程序，将释放的分区与它相邻的空闲分区进行合并，形成一个更大的空闲分区。分区的回收有四种情况:

（1）释放区与上下两个空闲区相邻。

（2）释放区与上空闲区相邻。

（3）释放区与下空闲区相邻。

（4）释放区与上下两个空闲区都不相邻。

4. 地址转换与存储保护

注：动态分区应采用动态地址重定位作业。当作业执行时由硬件地址转换机构完成地址转换。

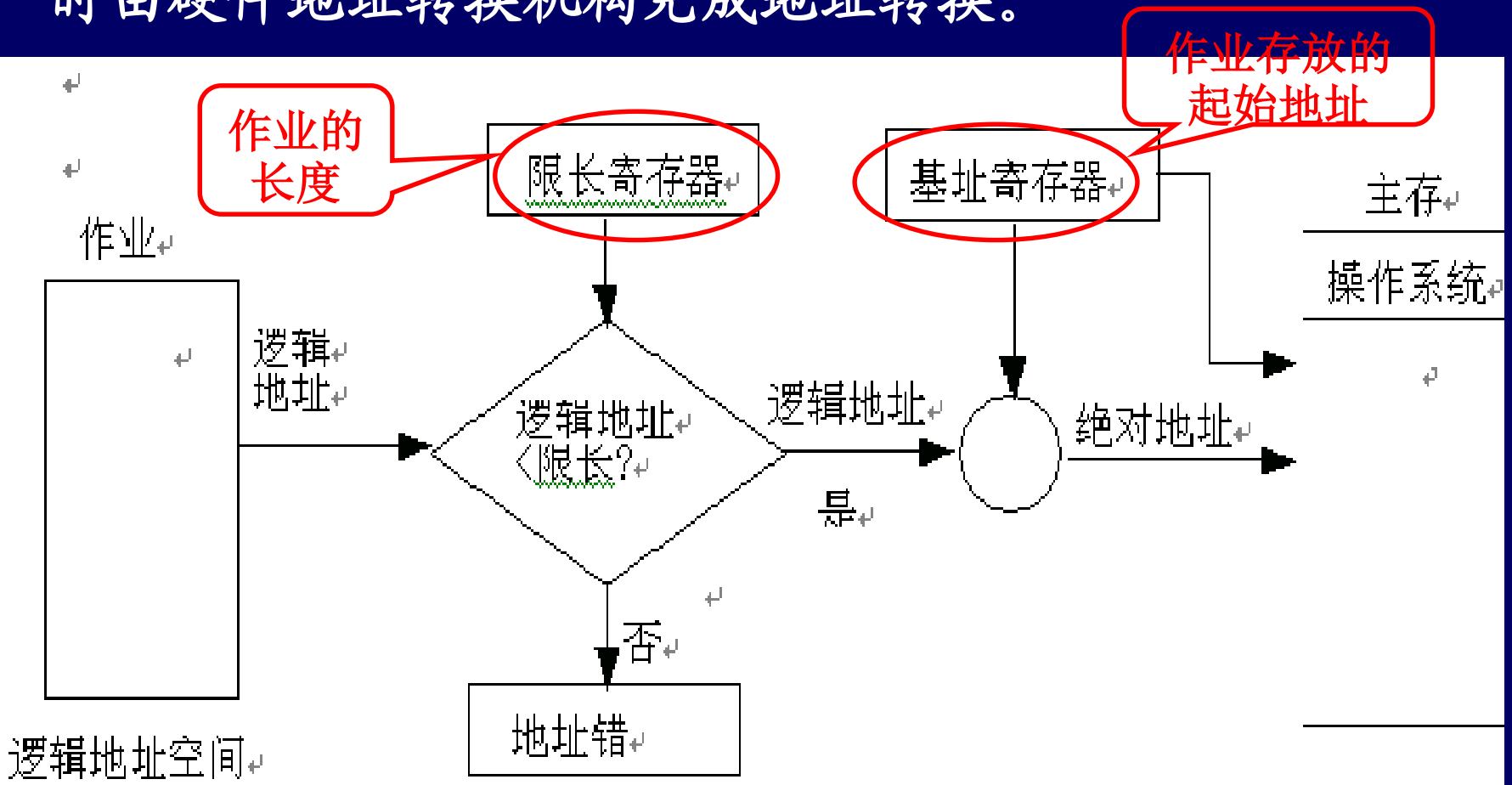


图 3-11 地址转换过程

4. 地址转换与存储保护

◆基址寄存器存放分配给作业使用的分区的最小绝对地址值;

◆限长寄存器存放作业占用的连续存储空间的长度。

◆基址寄存器、限长寄存器内容属于保护的现场，因此，在不考虑共享的情况下，即使是多道程序系统也只需要一对基址/限长寄存器。

◆当逻辑地址大于限长值时，表示作业欲访问的地址超出了所分得的区域，这时就产生地址越界中断，终止程序执行，报告地址出错信息，从而起到存储保护的作用。

5. 分区的共享

1）系统提供多对基址/限长寄存器；

2）几道作业共享的例行程序或数据可存放在一个共享的分区中，只要让各道作业的共享存储区域部分有相同的基址/限长值，就可实现分区共享；

3）规定对共享区的信息只执行或读，不能写；

6. 移动技术

当存储分配程序找不到一个足够大的空闲区来装入作业时，可以采用移动技术改变主存中的作业存放区域，同时修改它们的基址/限长值，从而使分散的小空闲区汇集成一个大的空闲区，有利于作业的装入。

6. 移动技术

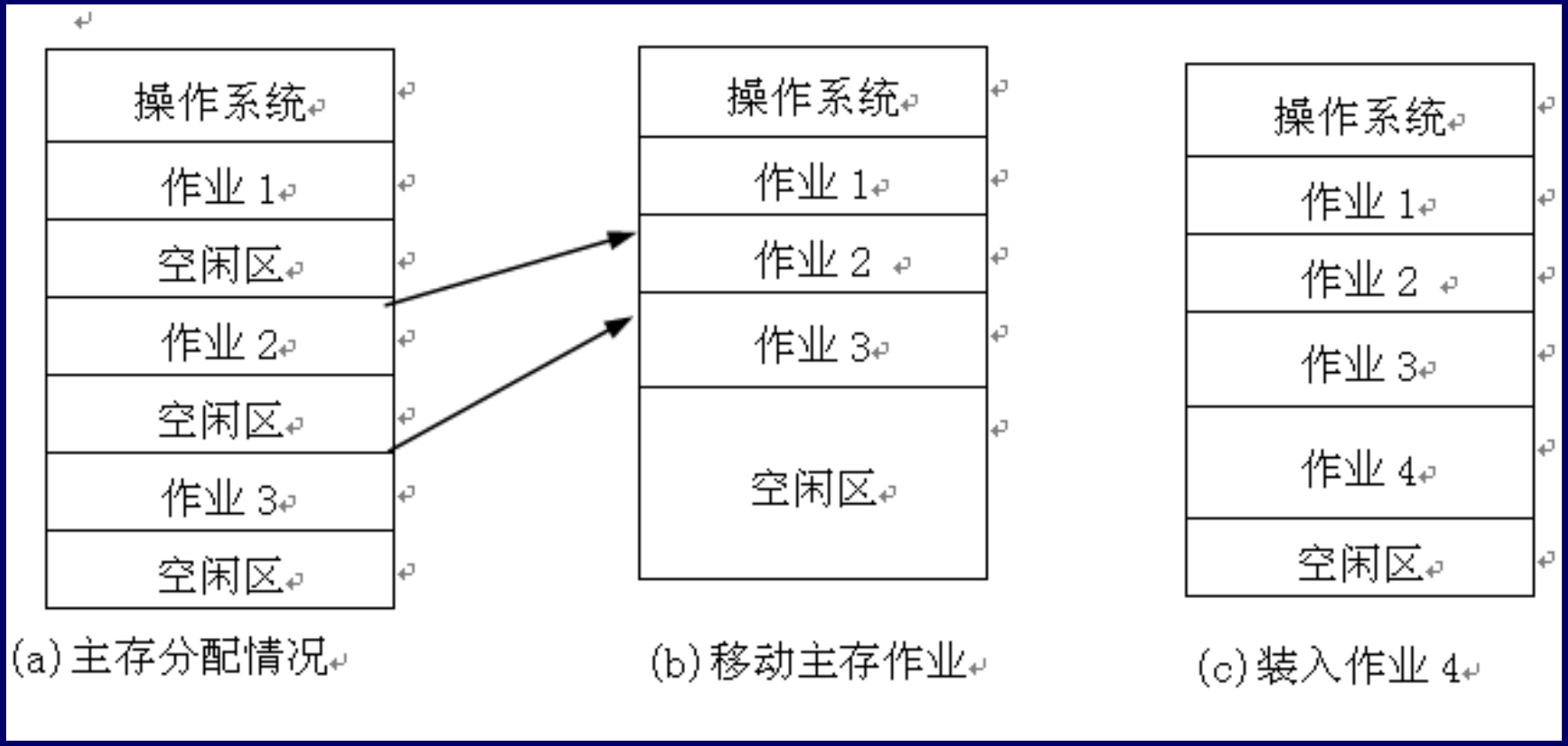


图 3-12 移动分配示例

6. 移动技术

移动技术的优缺点：

- 优点：

可使分散的“碎片”或小空闲区汇集成大的空闲区；为作业执行过程中扩充主存提供了方便。

- 缺点：

（1）增加了系统的开销；

（2）不是随时可移动，如：与外设交互时不能移动；

（3）作业动态申请主存会出现“死锁”。

解决的办法：撤出部分死锁作业，让一些作业获得主存运行，后归还主存，再将送出的作业调回运行。

7. 分区存储管理的优缺点

(1) 主要优点

1）实现了多道程序设计，从而提高了系统资源的利用率。

2）系统要求的硬件支持少，管理简单。

(2) 主要缺点

1）作业在装入时的连续性使主存的利用率不高。

2）移动技术提高主存利用率但带来一系列的问题。

3）主存的扩充只能采用覆盖与交换技术，无法真正实现虚拟存储。

虚拟存储技术：使进程的逻辑地址空间大于实际的主存物理空间，从而使有限的主存运行较大、较多的程序。

存储管理

3.1 存储管理的功能
3.2 存储管理的几种形式与重定位
3.3 单道环境下的存储管理
3.4 分区存储管理
3.5 页式存储管理

3.5.1 页式存储管理概述

一、分页管理的基本思想:

- 是将作业分配在不连续的大小相同存储区域中，同时又要保证作业的连续执行，每个区称为一块称为“页框”； 与此对应，编制程序的逻辑地址也分为页，称为“页面”，页的大小与块的大小相等 ，通常页的大小总是2的整数次幂。

二、分页存储器的逻辑地址格式:

页号	页内偏移

3.5.1 页式存储管理概述

三、改进（与分区管理相比）

- 欲实现连续存储到非连续存储的飞跃，为实现虚拟存储打下基础;
- 欲解决主存中的“碎片”问题，任意一个“内碎片”或“内零头”都不会大于整个页框的大小，从而提高主存的利用率。

四、分类

根据作业装入主存的时机不同，一般分为:

- 静态分页管理
- 虚拟分页管理。

3.5.2 静态分页管理

基本思想

- 用户作业在开始执行以前，将该作业的程序和数据全部装入到主存中，然后，操作系统通过页表和硬件地址变换机构实现逻辑地址到物理地址的转换，从而执行用户程序。

3.5.2 静态分页管理

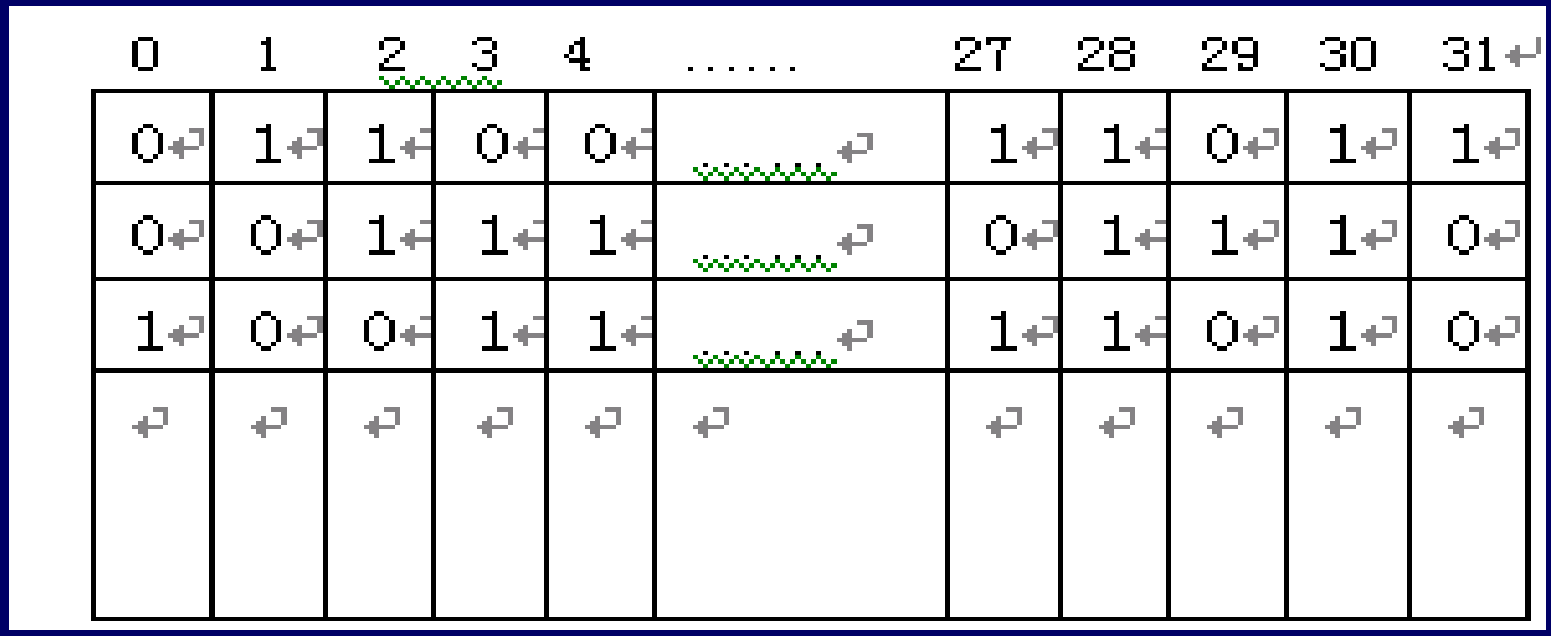
1．主存页框的分配与回收

为申请主存的作业或进程分配足够的页框。这就需要系统建立存储页框表、请求表和页表等数据结构，依据这些数据结构完成主存的分配和回收工作。

3.5.2 静态分页管理

- 存储页框表
  - 指出主存各页框是否已被分配，以及未被分配的页框总数。
- 形式 ：
  - 1、位示图：在主存中划分出一个固定的区域，该区域中每个单元的每个位表示一个页框的分配或空闲状况，若该位为1，代表所对应的页框已分配，若该位为0，代表所对应的页框空闲。浪费空间
  - 2、空闲页框链：在空闲页框链中，队首页框的第一单元和第二单元分别存放空闲页框的总数和指向下一个空闲页框的指针，其它页框的第一单元则分别存放指向下一个空闲页框的指针。比较经济合理

3.5.2 静态分页管理



0	1	2	3	4	.....	27	28	29	30	31
0	1	1	0	0		1	1	0	1	1
0	0	1	1	1		0	1	1	1	0
1	0	0	1	1		1	1	0	1	0

图3-13 位示图

3.5.2 静态分页管理

- 如何保证该程序能在非连续的存储空间里正确地运行呢？这就要求在执行每条指令时，要将程序中的逻辑地址变换为物理地址。
- 页表：

  在页式管理系统中实现地址变换的数据结构称为页面映像表，简称页表。
- 页表中的信息:

  一、页号。

  二、页面对应的页框，记录着该进程的每个页面分配到主存的哪些页框中。

3.5.2 静态分页管理

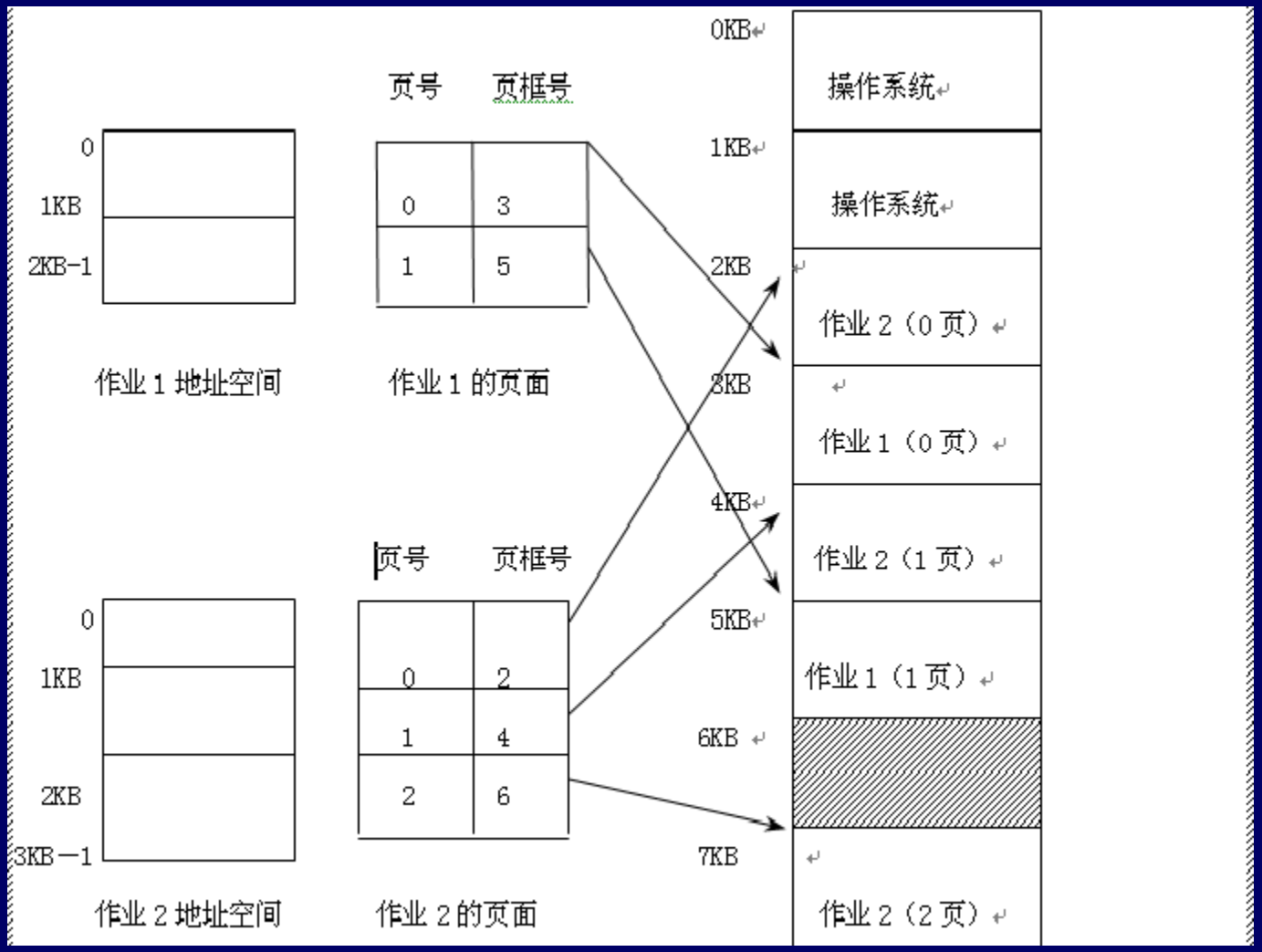


图3-14 静态分页主存分配映像

3.5.2 静态分页管理

- 请求表：用来确定进程的虚拟地址空间的各页表在主存中的实际对应位置。

进程号	请求页面数	页表始址	页表长度	状态
1	20	1024	20	已分配
2	30	1044	30	已分配
3	21	……	……	未分配
……	……	……	……	……

图3-15 请求表

3.5.2 静态分页管理

页框分配与回收算法:

一、分配:

- 从请求表中查出作业或进程所要求的页框数。
- 由存储页框表检查是否有空闲页框（块），若没有，则本次无法分配。如果有，则分配并设置页表，并填写请求表中的相应表项（页表始址、页表长度和状态）。
- 再按一定的查找算法，搜索出所要求的空闲页框（块），并将对应的页框（块）号填入页表中。

二、回收:

当进程执行完毕时，根据进程页表中登记的页框（块）号，将这些页框（块）插入到存储页框表中，使之成为空闲页框（块）。最后，拆除该进程所对应的页表即可。

3.5.2 静态分页管理

2.页式地址变换:

（1）根据主存的实际情况，将进程的每个页面分配到主存空闲页框中，同时，系统分配并设置页表的内容（通常一个进程具备一个页表）。此时，系统完成用户进程的存储器分配。

（2）当用户进程开始执行时，系统首先设置控制寄存器的内容，控制寄存器包括页表长度和页表起始地址两项。

（3）为了对逻辑地址进行变换，由硬件组成的地址变换机构必须将其分成两部分：页号和页内偏移（即2和452）。

（4）根据逻辑地址中提供的页号在页表中找到相对应的页框号（2 → 8）。

（5）将页表中的页框号和逻辑地址中的页内偏移分别写入绝对地址中的相应位置上（即8和452）。

（6）然后根据绝对地址提供的页框号和页内偏移计算出存储空间的物理地址（1KB=1024，8 × 1024 + 452 = 8644）。此时，用户进程便可以访问主存中的绝对地址，取出数据或取出指令执行。

3.5.2 静态分页管理

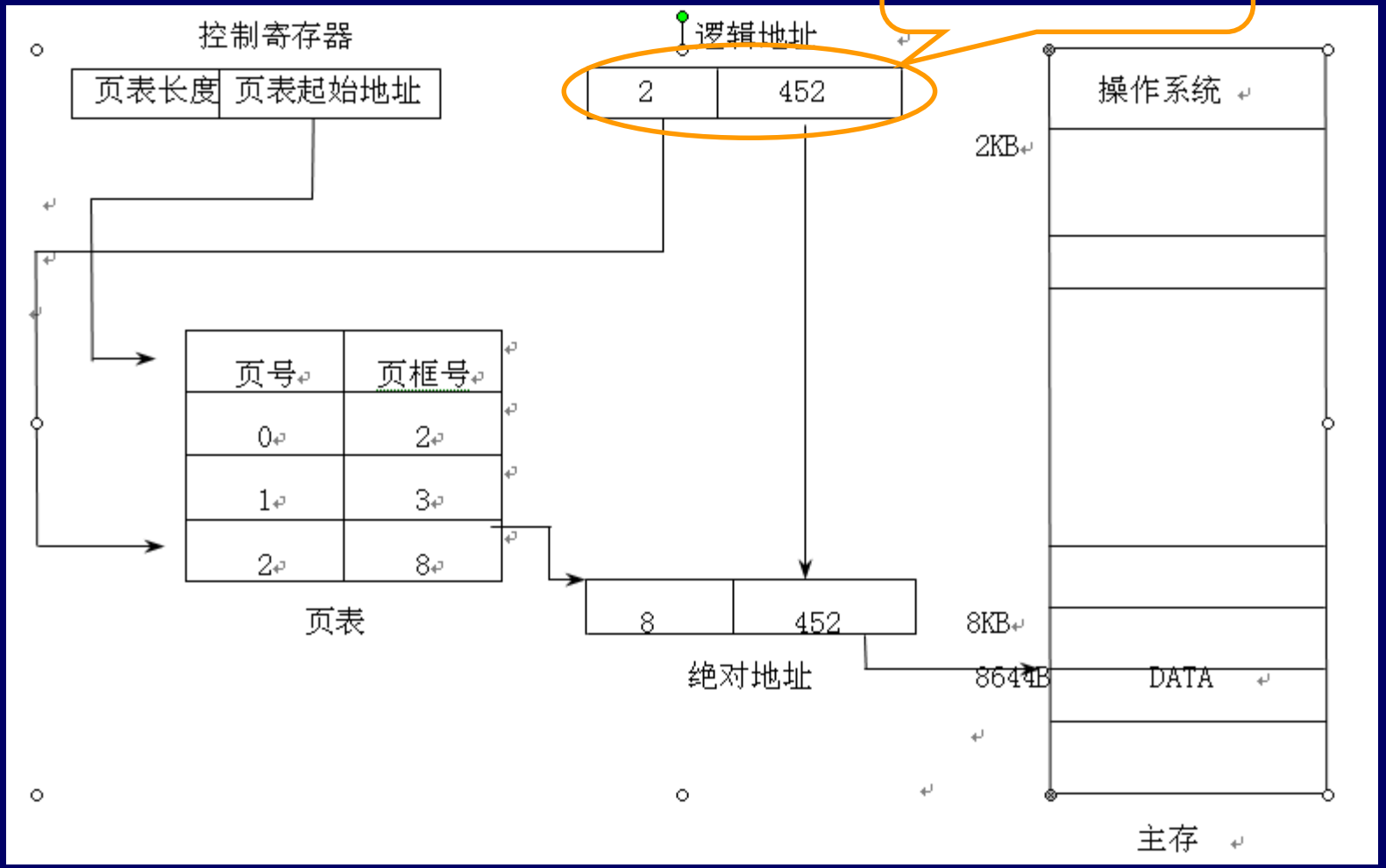


图3-16　地址变换

3.5.2 静态分页管理

绝对地址=块号*块长+单元号

例：假设一块为0.5k，作业为4.1k ，一页为0.5k，逻辑地址为1568，求绝对地址。

页表为：

页号	块号
0	3
1	2
2	5
3	4
4	1
5	10

解：[1568/512]=3　所以该页处在第4块

所求=1024*0.5*4 + (1568%512)=2080

3.5.2 静态分页管理

分页式地址转换机构的优缺点：

- 优点：简洁、清楚 。
- 缺点：执行一条访问主存的指令都要访问主存两次 ，执行速度下降了一半。 （1）一次访问页表得到所要访问指令的绝对地址 ；（2）根据绝对地址访问实际所需的单元 。

3.5.2 静态分页管理

3. 快表:

一、引入:

执行一条访问主存的指令都要访问主存两次 ，为了尽量减少访问主存的次数，提高地址变换的速度，可在地址变换机构中增设一个具有并行查寻能力的特殊高速缓冲存储器，用来存放页表的一部分。

二、概念:

快表：存放在高速缓冲存储器中的页表。

3.5.2 静态分页管理

加入快表后地址转换过程:

- CPU在给出逻辑地址后，地址变换机构首先根据页号在快表中进行检索，若存在相应的页号，则直接从“快表”中读出该页号对应的页框号，形成物理地址;
- 否则，需要再访问主存中的页表，从页表中读出相应的页框号，形成物理地址，同时将找到的页表项登记到“快表”中。

注：当“快表”填满后，又要在“快表”中登记一新的页表项时，则需采用一定的淘汰策略在“快表”中淘汰一个老的、已被认为不再需要的页表项。

先进先出—FIFO；最近最少用淘汰法—LRU

3.5.2 静态分页管理

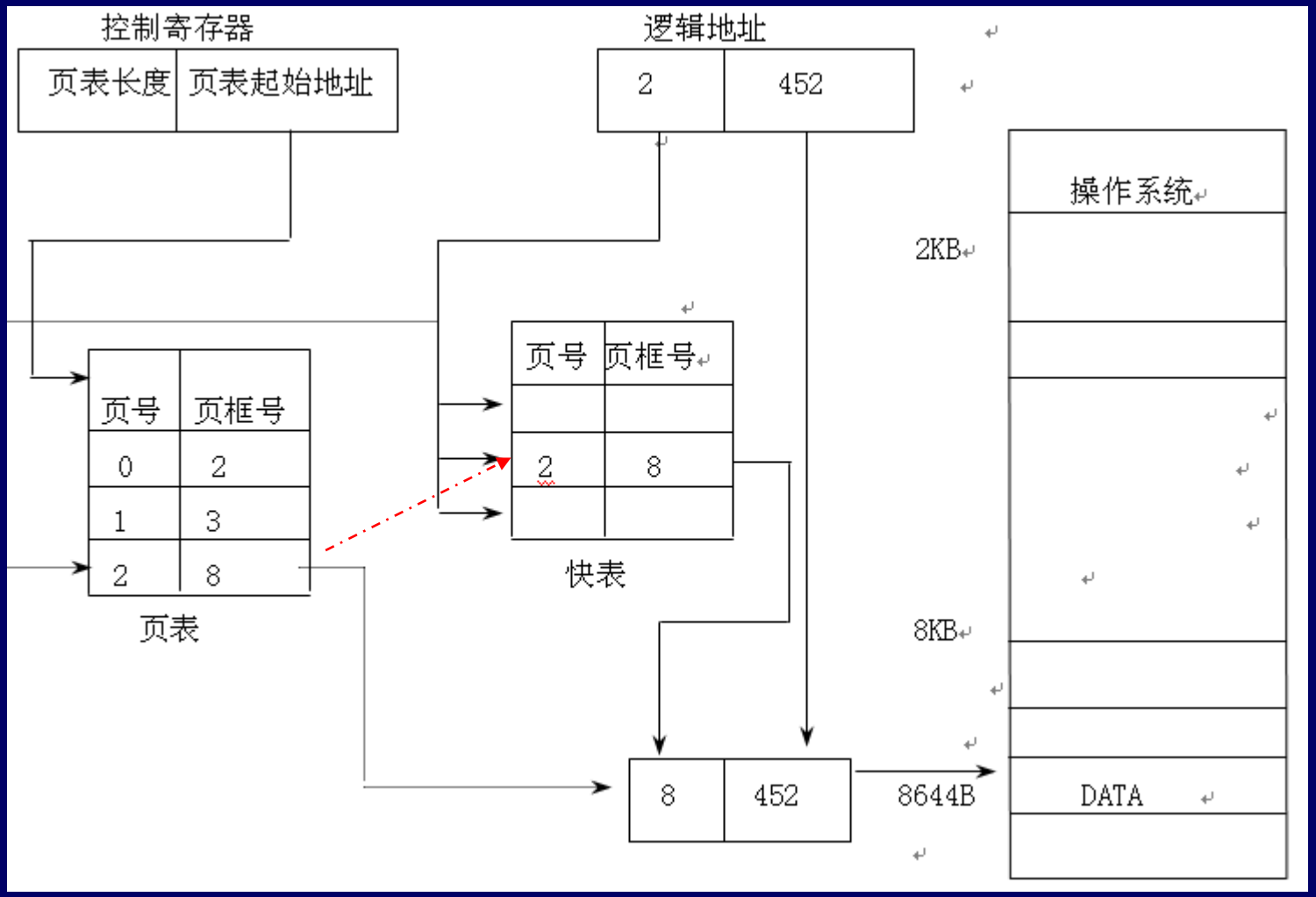


图3-17 快表实现地址变换

3.5.2 静态分页管理

4. 页的共享与保护:

页的共享：提高主存空间的利用率。

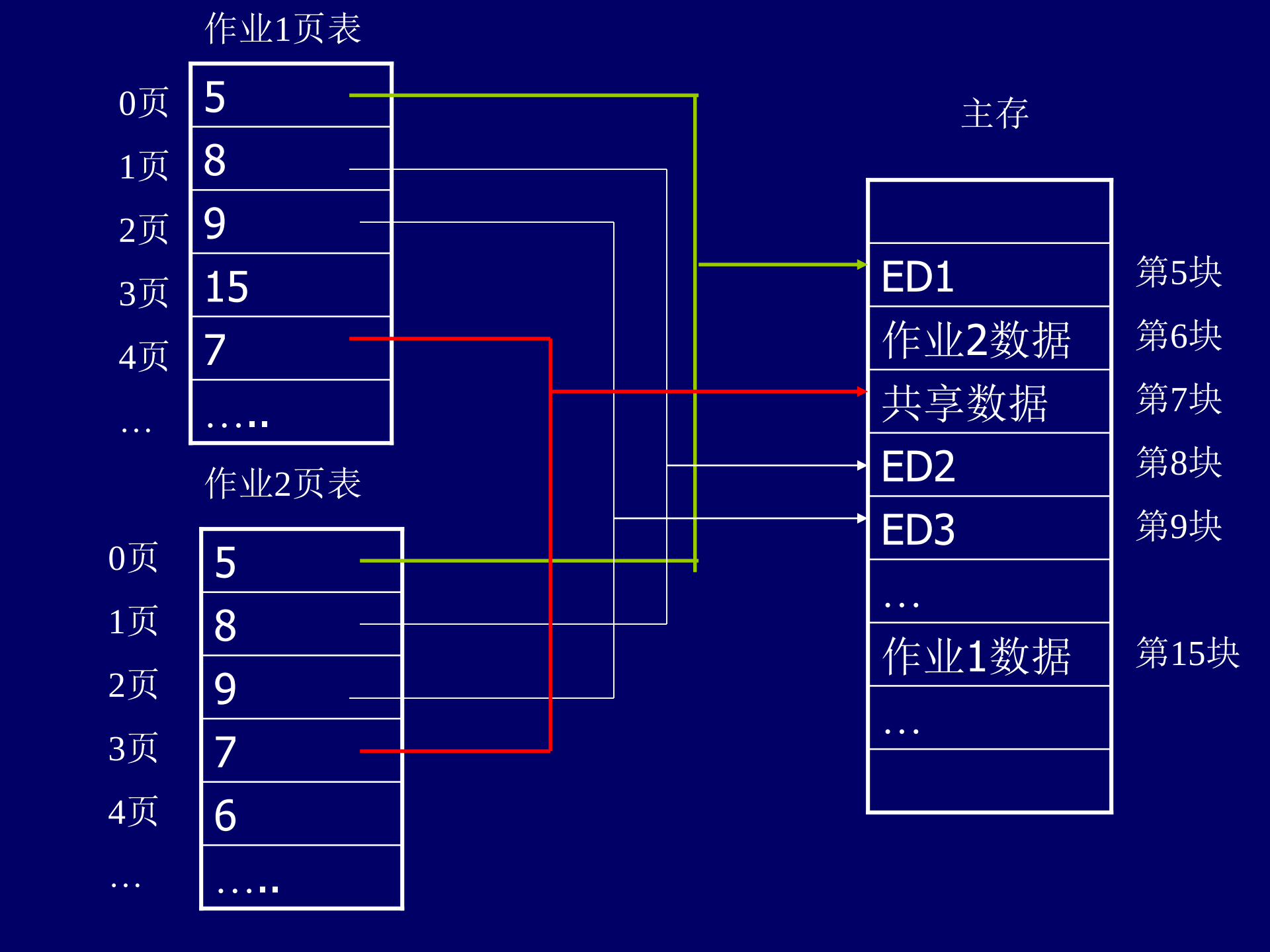
共享的实现

（1）实现数据共享

- 作业对共享数据页可使用不同的页号，只要相应页表项指到同一主存块即可。

（2）实现程序共享

- 对共享程序必须规定一个统一的页号。


作业1页表
0页
1页
2页
3页
4页
…
5
8
9
15
7
…..
作业2页表
0页
1页
2页
3页
4页
…
5
8
9
7
6
…..
主存
ED1
作业2数据
共享数据
ED2
ED3
…
作业1数据
…
第5块
第6块
第7块
第8块
第9块
第15块

3.5.2 静态分页管理

共享信息的保护：

（1）页表中增加一个保护权限域

- 指出该页的信息为：可读/写、只读、只执行和不可访问等，指令执行时进行操作权限核对 。

（2）保护键法

- 系统为每道作业设置一个保护键，为作业分配主存时，根据它的保护键在相应的页表中建立键标志。程序执行时将程序状态字中的键和访问页的保护键进行核对，相符时才可访问该页框。

3.5.2 静态分页管理

总结：

- 优点：解决了分区管理时的碎片问题 。

- 缺点：由于静态分页管理要求作业在装入时必须一次性地全部装入主存，如果当时系统中可用的页框数小于用户要求时，该作业只好等待，即作业的大小仍受主存中可用页框数的限制。

3.5.3 虚拟页式存储管理

局部性原理：在一较短的时间里，程序的执行仅局限在某个部分，相应地，它访问的存储空间也局限在某个区域。（虚拟页式管理的依据）

3.5.3 虚拟页式存储管理

- 局部性又表现为时间局部性和空间局部性
- 时间局部性:
  - 如果程序中某一条指令一旦执行，则在不久以后还可能被继续执行；同样，若某一个数据被访问后不久，还可能被继续访问。其典型的情况是程序中存在着大量的循环。
- 空间局部性
  - 如果程序访问了某一个存储单元，其附近的存储单元则在不久也会被访问。即程序在一段时间内访问的地址，可能集中在一定的范围内。其典型的情况是程序顺序执行。

3.5.3 虚拟页式存储管理

- 1.虚拟存储器的基本思想：当用户作业要求的存储空间很大，不能被装入主存时，基于局部性原理，系统可以把当前要用的程序和数据装入主存中启动程序运行，而暂时不用的程序和数据驻留在外存中。在执行中需要用到不在主存中的信息时，可将暂时不用的程序和数据调出主存，腾出主存空间让系统调入要用的程序和数据。从用户角度上看，系统具备了比实际主存容量大得多的存储器，人们把这样的存储器称为虚拟存储器。

3.5.3 虚拟页式存储管理

虚拟存储器的特征:

1）多次性

——用户程序在运行前，并不是一次将全部内容装入到主存中，而是在程序的运行过程中，系统不断地对程序和数据部分地调入、调出，完成程序的多次装入工作。

2）对换性

——程序在运行期间，允许将暂时不用的程序和数据调出主存（换出），放入外存的对换区中，待以后需要时再将它调入主存中（换入），这便是虚拟存储器的换入、换出操作，即对换性。

- 虚拟存储的多次性和对换性必须建立在离散分配的基础上。

3.5.3 虚拟页式存储管理

2. 用分页技术实现虚拟存储器

数据结构

页框号	状态位	访问字段	修改位	保护权限	外存地址

图 3-19 分页式虚拟存储系统中页表项

(1) 状态位 用于指示该页是否已调入主存，如用1表示在主存，0表示不在主存。

(2) 访问字段 用于记录本页在一段时间内被访问的情况，提供给置换机构参考。

3.5.3 虚拟页式存储管理

(3) 修改位 表示该页在调入主存后是否被修改过，由于主存中的每一页都在外存上保留一份副本，因此，若未被修改，在置换该页时就不需将该页写回到外存上；若已被修改，则必须将该页重写到外存上，以保证外存中所保留的始终是最新副本。

(4) 保护权限：说明该页允许什么类型的访问。它指出了该页的信息可能为：可读/写、只读、只执行和不可访问等.

(5) 外存地址：指出该页在外存上的地址，供调入该页时使用。

3.5.3 虚拟页式存储管理

二级页表的地址变换:

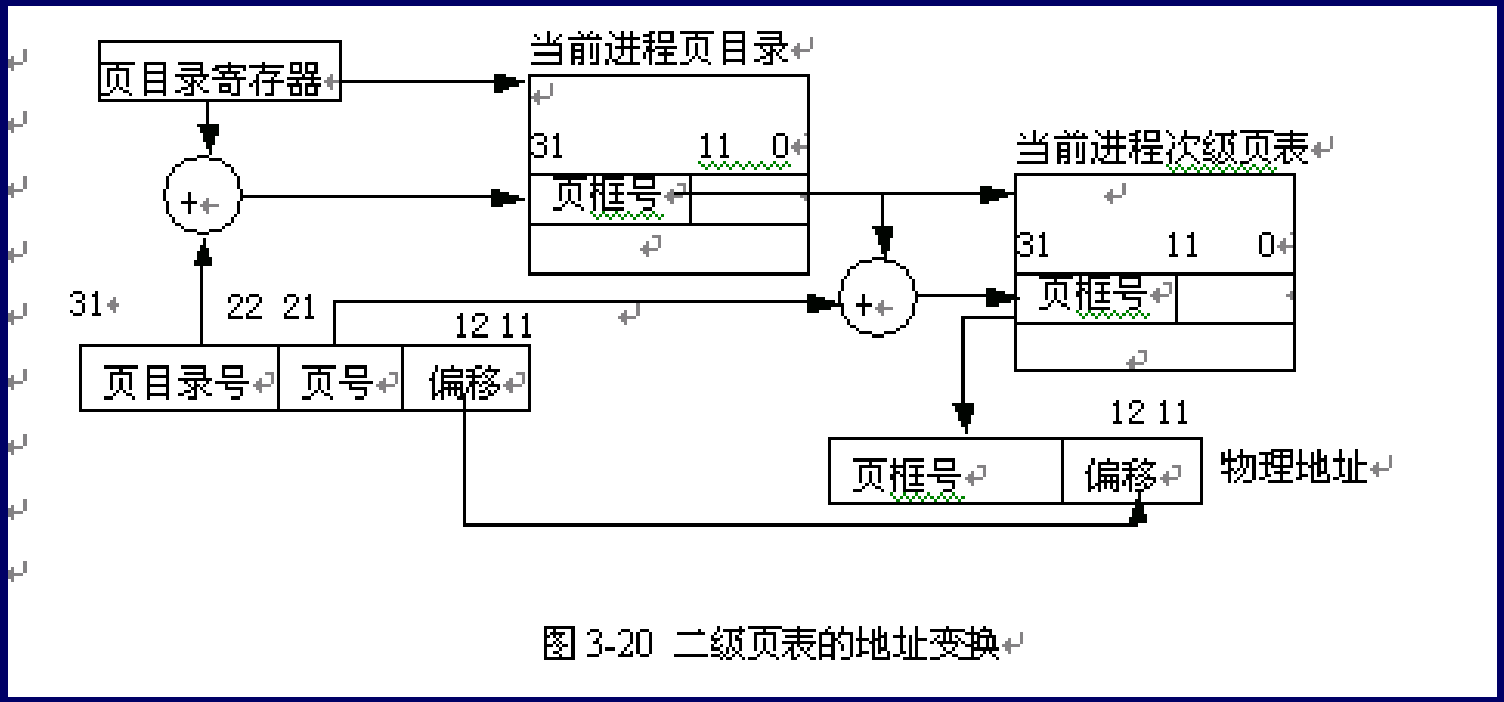


图 3-20 二级页表的地址变换

三次访问主存:

一次访问页目录，一次访问页表，最后才访问数据所在的物理地址。通过采用反向页表等方式进行改进。

3.5.3 虚拟页式存储管理

- 硬件环境 ：

(1) 具有一定容量主存，用于存放一个操作系统，以及每个进程的部分程序、数据和相应的页表表项;

(2) 相当容量外存，用于存放每个进程未装入主存的部分、后备作业以及大量的文件。

(3) 地址变换机构，用于将用户程序地址空间中的逻辑地址变换为主存的物理地址。

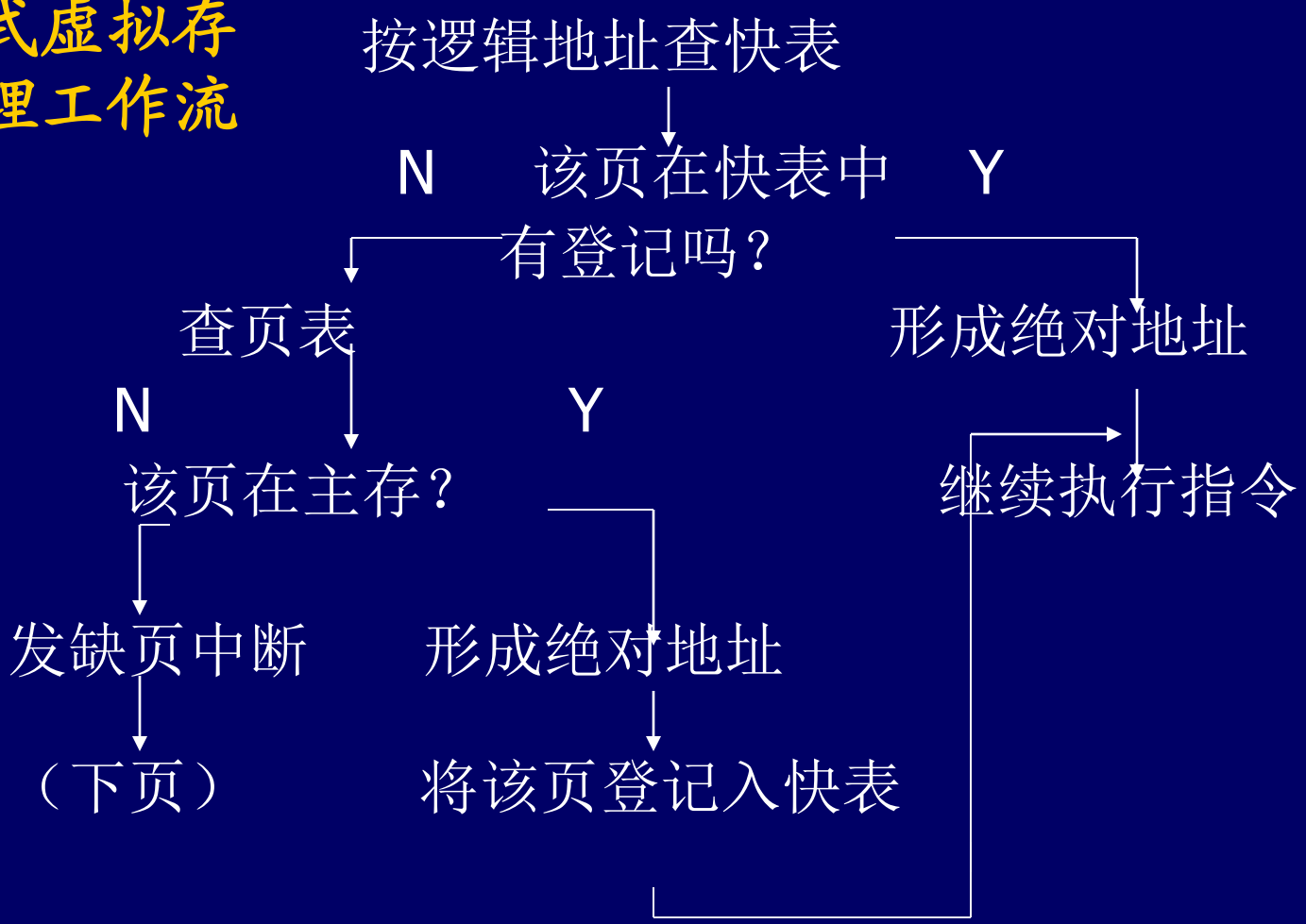
(4) 缺页中断机构，当发现所要访问的页不在主存时，应立即发出缺页中断信号，以请求操作系统将所缺之页调入主存。

3.5.3 虚拟页式存储管理

- 缺页中断与一般中断的区别:
  - 在指令执行期间产生和处理中断信号。通常CPU都是在一条指令执行完后检查是否有中断请求到达，若有，便去响应，否则继续执行下一条指令。然而缺页中断是在指令执行期间发现所要访问的指令或数据不在主存时产生和处理的。
  - 一条指令在执行期间，可能产生多次缺页中断。如采用多级页表时，页表也是作为动态调入的，地址翻译过程就可能出现访问页表的缺页中断，最后才产生所要访问的页的缺页中断。

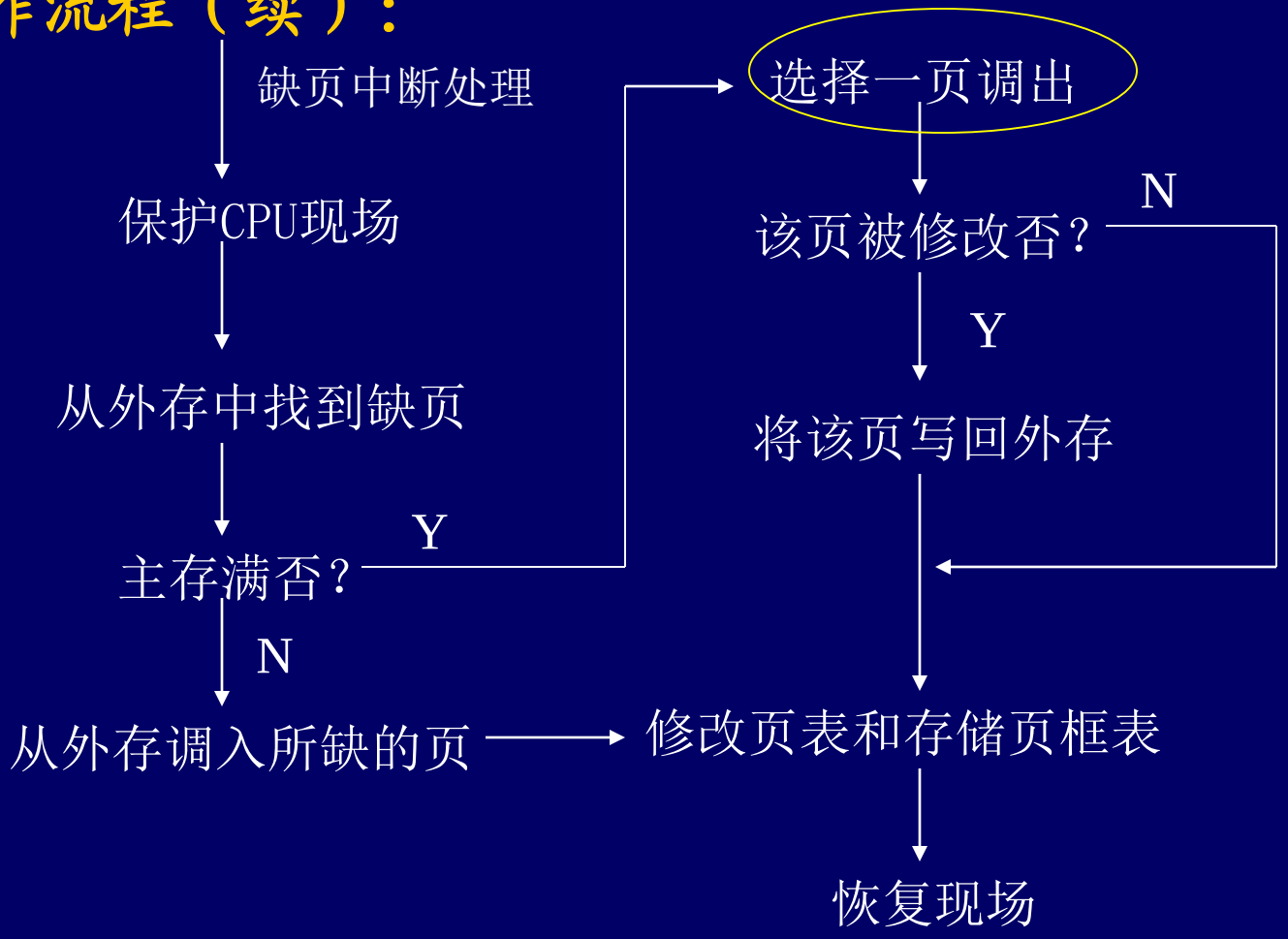
3.5.3 虚拟页式存储管理

分页式虚拟存储管理工作流程:



3.5.3 虚拟页式存储管理

分页式虚拟存储管理工作流程（续）：



3.5.3 虚拟页式存储管理

3. 页面置换

（1）. 调页策略

---系统应在何时及采用什么策略将进程所需的程序和数据页面调入主存呢？目前有两种调度页面的策略：预调页策略和请求调页策略。

1）预调页策略

- 该策略应以预测为基础，只将那些预计不久后便会被访问的程序或数据所在的页面预先调入主存。

3.5.3 虚拟页式存储管理

2）请求调页策略

- 当进程运行中需要访问某部分程序和数据，而其所在页面又不在主存时，立即提出请求，由系统将其所需页面调入主存。每次请求后只调入一页，系统开销较大。

从何处调？

（1）硬盘文件区：从未运行过的页面。

（2）硬盘对换区：曾经运行过但被调出的页面。

（3）主存中的页面缓冲池：共享页面。

3.5.3 虚拟页式存储管理

（2）.分配策略

1）固定分配策略

- 思想：每个进程分配一固定页数的主存空间，在整个运行期间不再改变。缺页时只能调出自己的页面。
- 缺点：应为每个进程分配多少个页框的主存难于确定。

2）可变分配策略

- 思想：为每个进程分配一定数量的主存空间，如果该进程在运行过程中频繁地发生缺页中断，则系统再为该进程分配若干附加的物理页框，直至进程减到适当的缺页率为止；反之，若一个进程在运行过程中的缺页率特别低，则此时可适当减少分配给该进程的物理页框。

3.5.3 虚拟页式存储管理

（3）. 页面置换算法

- 定义：主存已满，又需装入新页时，将主存中的某些页调出去。这一工作称为页面调度或页面置换。页面调度实质上是决定淘汰那一页。
- 不合适的调度算法：
  - 若刚被淘汰的页面又立即要用到，因此又要把它调入而调入不久再被淘汰，淘汰不久又再被调入。如此反复，使得整个系统的页面置换非常频繁，以至于大部分时间都花费再来回调度上。这种现象称作“抖动”或“颠簸”。
- 缺页中断率为：f=不成功的访问次数/(成功的访问次数+不成功的访问次数)

3.5.3 虚拟页式存储管理

（3）页面置换算法:

1）优化算法（OPT）

这是一种理论化的算法，其所选择的被淘汰的页将是永不使用的页，或者是在最长时间内不再访问的页。

- 注意：OPT是一种理想的算法,不实际,只能作为衡量其它算法的标准。

3.5.3 虚拟页式存储管理

2）先进先出算法（FIFO）

该算法总是淘汰最先进入主存的页面，认为最先调入的页访问可能性最小。

优点：实现简单，只需把一个进程已调入主存的页面，按先后次序链接成一个队列，并设置一个指向最老页面的替换指针即可。

缺点：

(1). 该算法是基于CPU按线性顺序访问地址空间的假设上的，与进程实际运行的规律不一定相适应。

(2).会出现一种奇异现象—Belady现象。

3.5.3 虚拟页式存储管理

- **奇异现象:**

  在未给作业分配足够满足它要求的页面数时，有时会出现分配的**页框数增多**，而**缺页中断率反而增高**的奇异现象 。

3.5.3 虚拟页式存储管理

FIFO算法执行过程（设P共有8个页面，分配3个页框）

访问次序	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1
主存页号	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0
		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1
			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2
淘汰				7		0	1	2	3	0	4			2	3		

图 3-23 FIFO 算法执行情况

缺页中断率 = （12 / 17） * 100% = 75%

3.5.3 虚拟页式存储管理

主存分配的页框若增加为4：

访问次序	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1
主存页号	7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2
		0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4
			1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
				2	2	2	2	2	2	2	2	2	2	1	1	1	1
淘汰						7		0			1			2	3		

图3-24 FIFO算法执行情况

缺页中断率＝9／17＊100％＝52.9％。

3.5.3 虚拟页式存储管理

Belady现象(1)

访问次序	1	2	3	4	1	2	5	1	2	3	4	5
主存页号	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
淘汰				1	2	3	4			1	2	

图 3-25 FIFO 算法的 Belady 现象（1）

缺页中断率 = 9 / 12 * 100% = 75%

3.5.3 虚拟页式存储管理

奇异现象（2）

访问次序	1	2	3	4	1	2	5	1	2	3	4	5
主存页号	1	1	1	1	1	1	5	5	5	5	4	4
		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
				4	4	4	4	4	4	3	3	3
淘汰							1	2	3	4	5	1

图 3-26 FIFO 算法的 Belady 现象（2）

缺页中断率 = 10 / 12 * 100% = 83.3%

3.5.3 虚拟页式存储管理

3）最近最少用置换算法（LRU）

- **思想：** 该算法要求淘汰的页面是在最近一段时间里较久未被访问的那一页。根据是程序执行时所具有的局部性。
- **实现方法：** 为了比较准确地淘汰最近最少使用的页面，可以采用堆栈的方法来实现。栈中存放当前主存中的页号，每当访问一页时就调整一次栈。于是，发生缺页中断时总是淘汰栈底所指示的页。

优点：性能很好。

缺点：堆栈的调整要花费很长的时间。

3.5.3 虚拟页式存储管理

例：P共有5个页面，分配3个页框：

缺页率=6/9*100%=64.4%

3.5.3 虚拟页式存储管理

4）最近未用置换算法（NRU）

思想:

- 该算法要求页表中有一个访问位和一个修改位。当某页被访问时，访问位被自动置1，若执行的指令是写指令，则修改位也被置1。系统周期性地将所有访问位置0。在选择一页淘汰时，总是选择其访问位为0且修改位也为0的页。若无修改位为0的页，就选访问位为0且页号最小的页淘汰。

评价:

- 该算法不但希望淘汰的页是最近未使用的页，而且还希望被淘汰的页是在主存驻留期间其页面内容未被修改过。
- 系统对访问位清0的间隔时间T的确定是很关键的。如果间隔时间T太大，可能所有页的访问位均已成为1，无法选择淘汰的页面。如果间隔时间T太小，则可能很多页的访问位均是为0 。

3.5.3 虚拟页式存储管理

分页式虚拟存储系统的性能分析:

影响缺页率的因素:

1）程序的局部化

2）页面的大小

页面越大，缺页率越低。

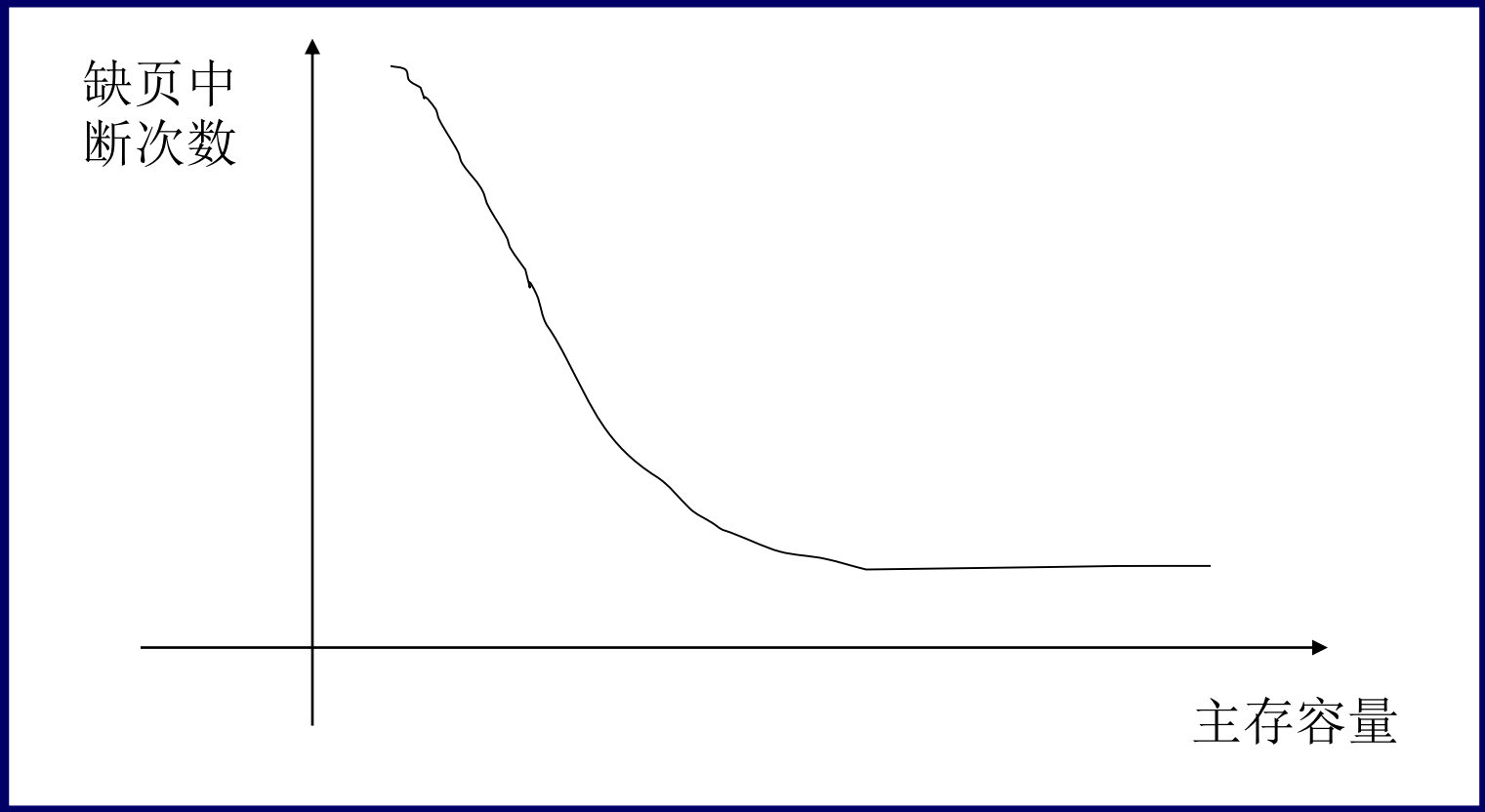
3）每个进程分配的页框的个数

同时运行的程序越多，缺页率越高。

4）调度算法

3.5.3 虚拟页式存储管理

分页式虚拟存储系统的性能分析



3.5.3 虚拟页式存储管理

分页存储管理优缺点：

优点：

1）解决主存的零头问题，能有效地利用主存。

2）方便多道程序设计，并且程序运行的道数增加了。

3）可提供大容量的虚拟存储器，作业的地址空间不再受实际主存大小的限制。

4）更加方便了用户，特别是大作业的用户。当某作业地址空间超过主存空间时，用户也无需考虑覆盖结构。

3.5.3 虚拟页式存储管理

缺点：

1）要有相应的硬件支持，如需要动态地址变换机构、缺页中断处理机构等。

2）必须提供相应的数据结构来管理存储器，它们不仅占用了部分主存空间，同时还要花费CPU时间。

3）在分页系统中页内的零头问题仍然存在。

4）在请求分页管理中，需要进行缺页中断处理，还有可能出现抖动现象，增加了系统开销，降低系统效率。

存储管理

3.1 存储管理的功能
3.2 存储管理的几种形式与重定位
3.3 单道环境下的存储管理
3.4 分区存储管理
3.5 页式存储管理