華中科技大學
计算机科学与技术学院
School of Computer Science & Technology, HUST

操作系统原理

第七章 主存管理

授课人：郑然

主存管理——主要内容

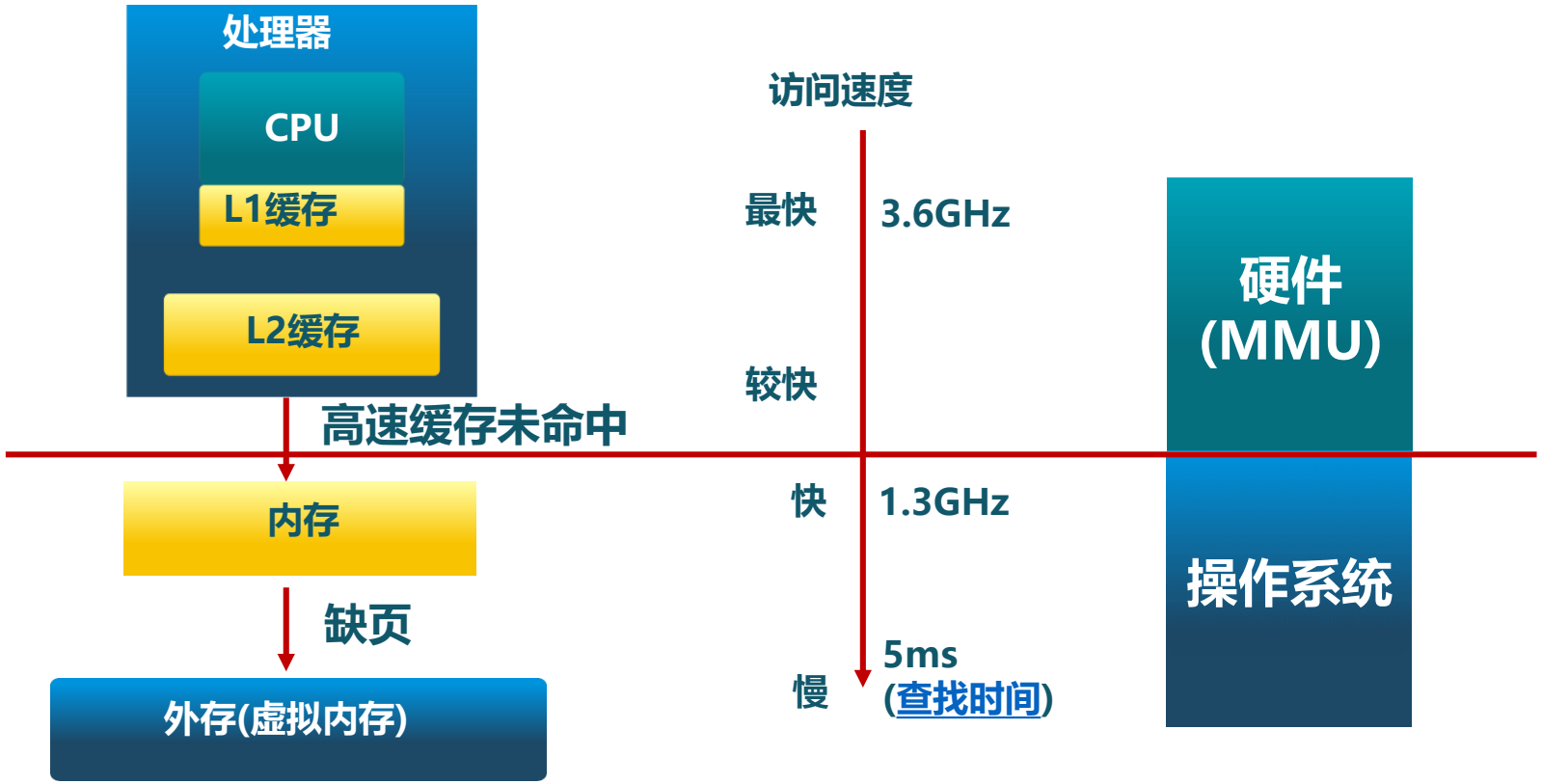
- 主存管理概述
- 主存管理的功能
- 分区存储管理
- 页式存储管理
- 段式及段页式存储管理

主存管理概述

主存管理——主存管理概述

存储器：能接收和保存数据、并根据命令提供这些数据的装置，分为主存和辅存。

内存层次



主存管理——主存管理概述

1. 主存共享方式

(1) 大小不等的区域

① 分区存储管理

② 段式存储管理

(2) 大小相等的区域

◆页式存储管理

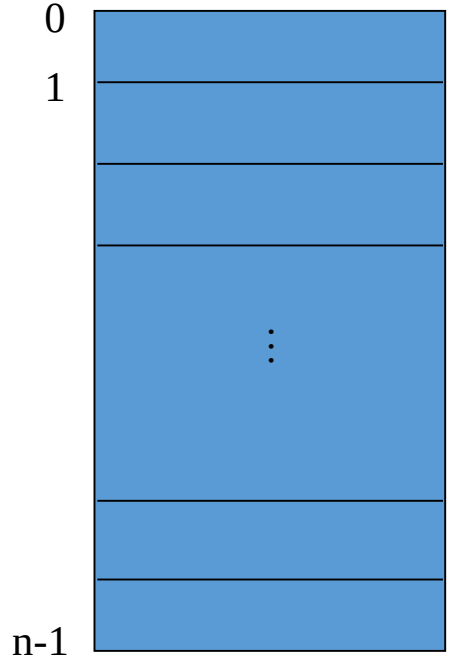
(3) 二者结合

◆段页式存储管理

2. 程序的逻辑组织

(1) 一维地址结构

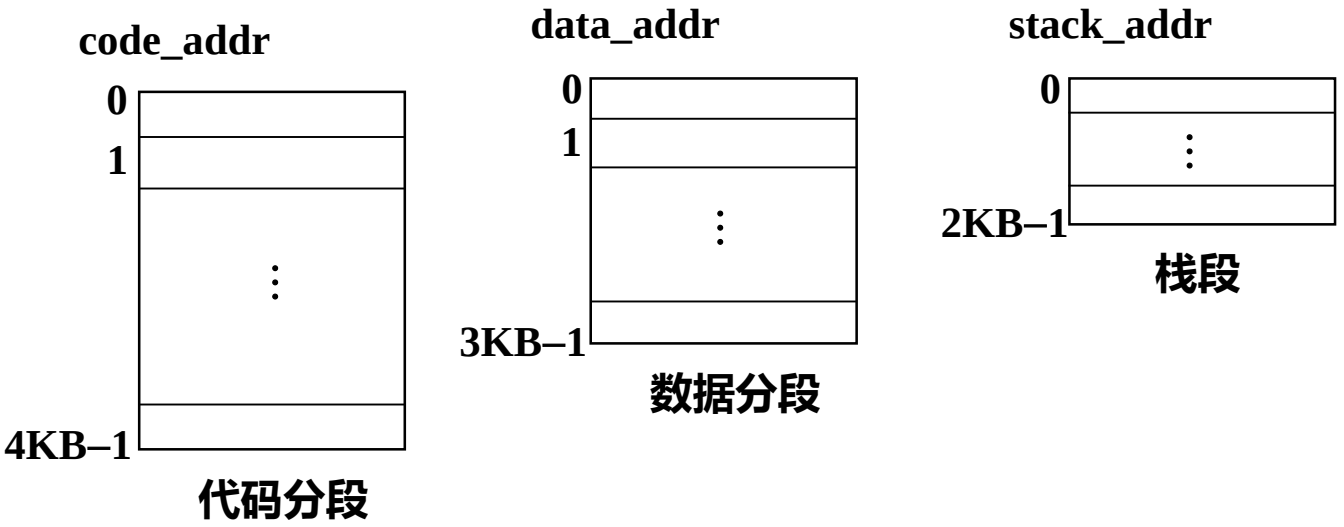
- 一个程序是一个连续、线性的地址结构；
- 确定线性地址空间中的指令地址或操作数地址只需要一个信息。



程序地址空间
一维地址结构

(2) 二维地址结构

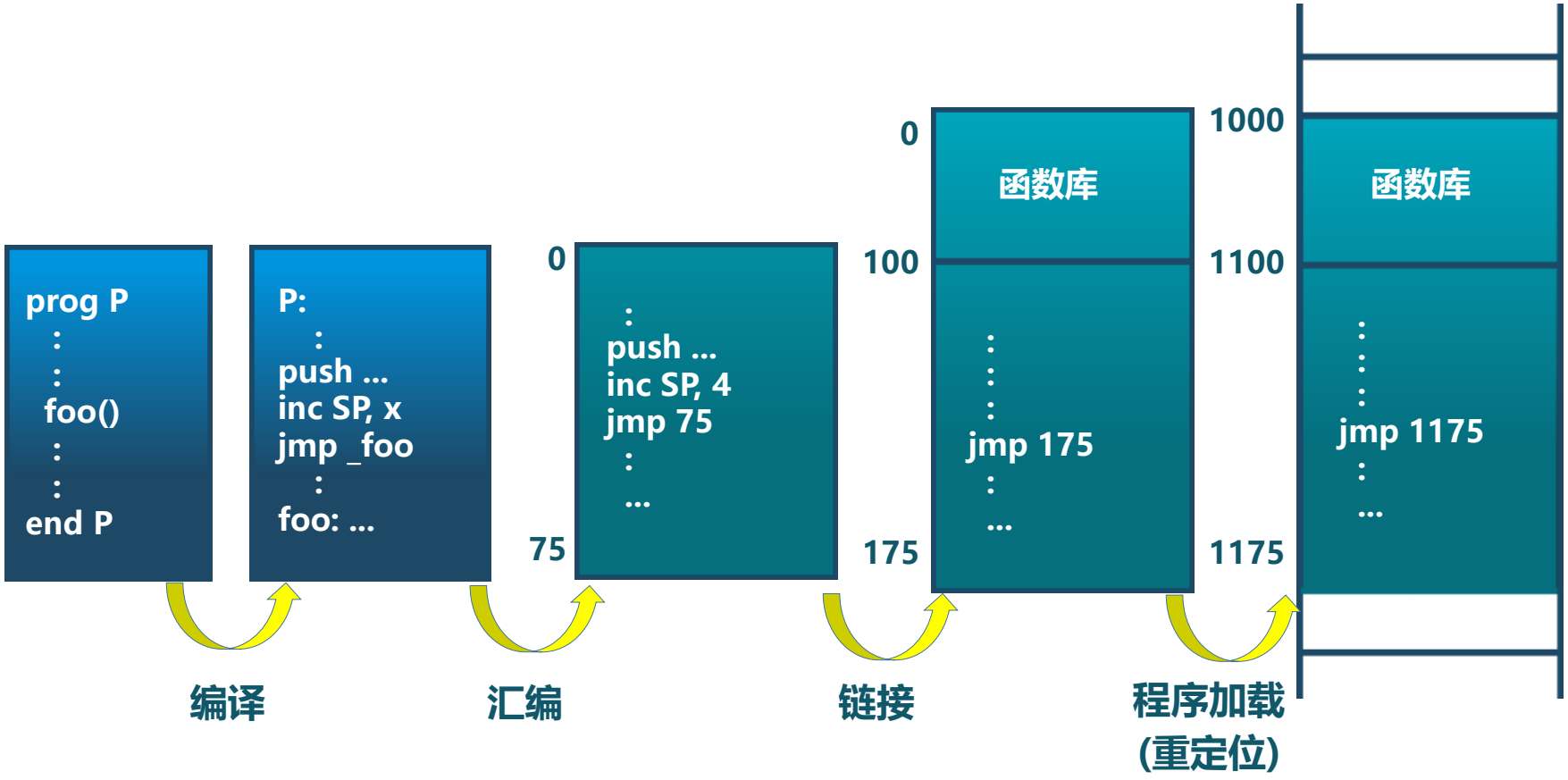
- 一个程序由若干个分段组成，每个分段是一个连续的地址区；
- 确定线性地址空间中的指令地址或操作数地址需要两个信息，一是该信息所在的分段，另一个是该信息在段内的偏移量。



程序地址空间——二维地址结构

主存管理功能

地址生成



程序名字空间　　逻辑地址空间　　物理地址空间

主存管理——主存管理功能

1. 几个概念

(1) 物理地址 (绝对地址、实地址)

物理地址是计算机主存单元的真实地址，又称为绝对地址或实地址。

(2) 主存空间

物理地址的集合所对应的空间组成了主存空间。

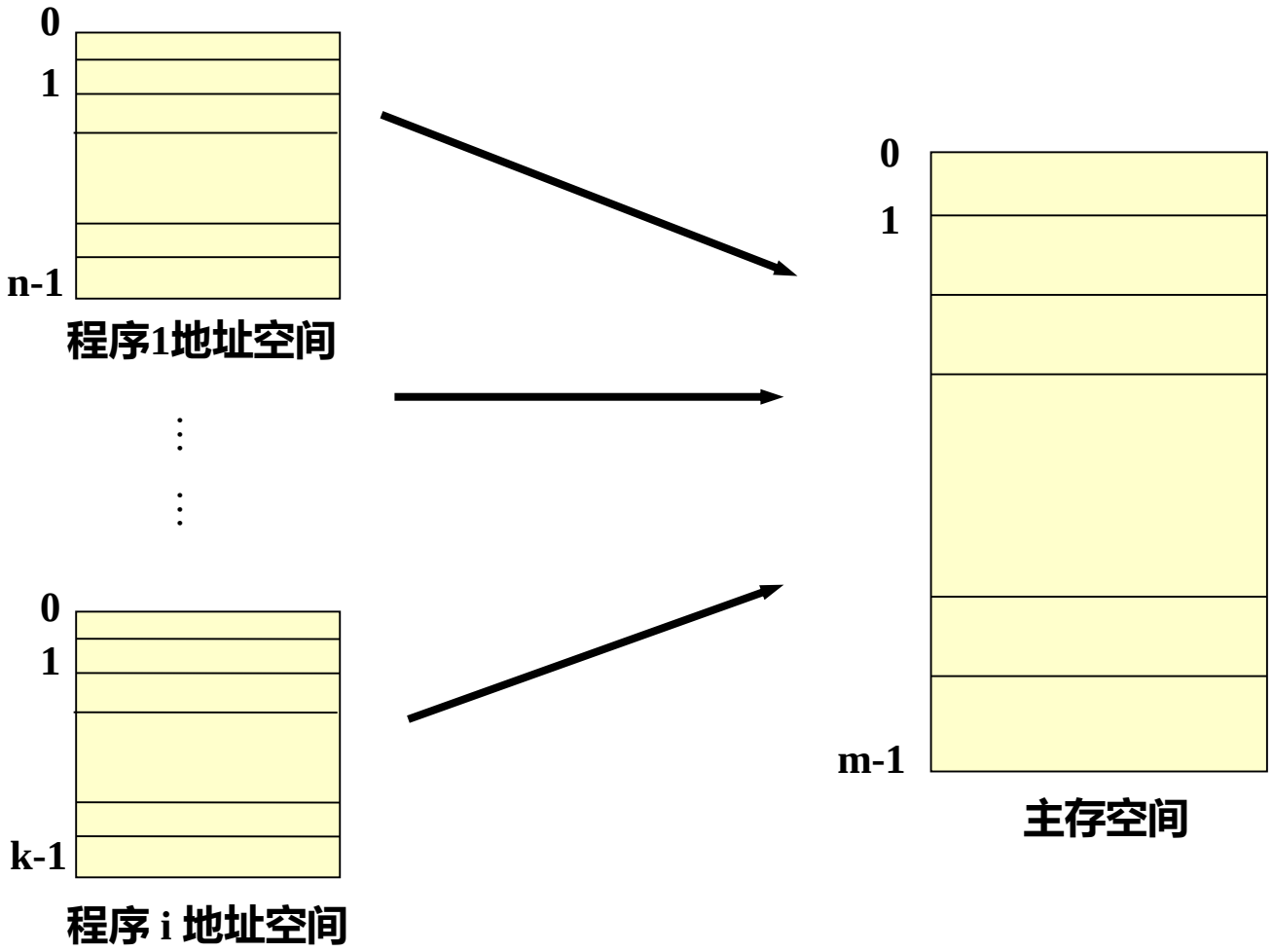
(3) 逻辑地址 (相对地址、虚地址)

用户的程序地址 (指令地址或操作数地址)均为逻辑地址。

(4) 程序地址空间

用户程序所有的逻辑地址集合对应的空间。

(5) 程序地址空间与主存空间



程序地址空间与主存空间示意图

需要考虑什么问题?

- 如何寻址? 地址映射
- 资源分配? 主存分配
- 共享使用? 存储保护
- 空间有限? 主存扩充

主存管理——主存管理功能

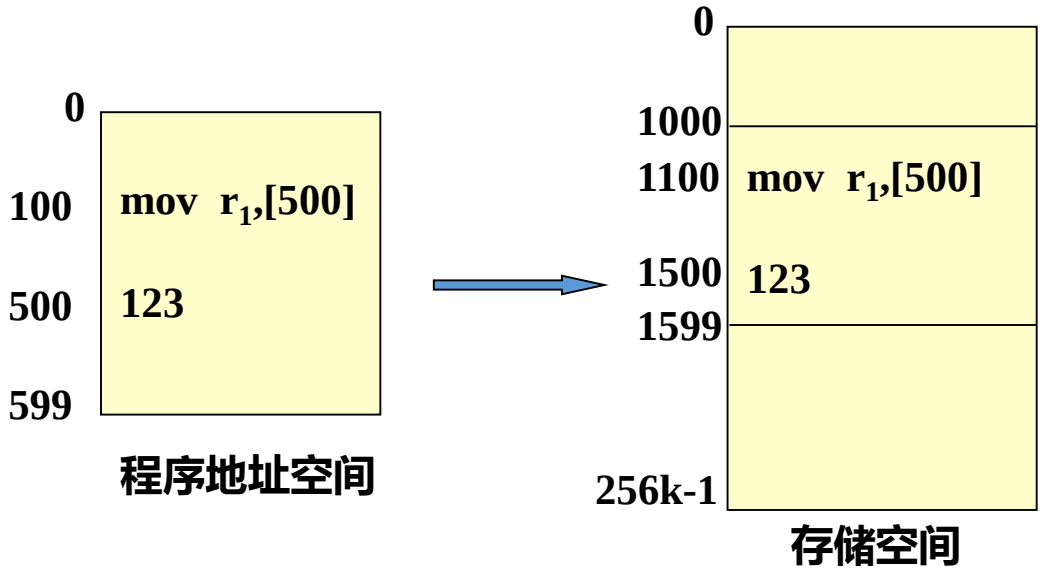
2. 主存管理功能

- 实现逻辑地址到物理主存地址的映射
- 主存分配
- 存储保护
- 主存扩充

3. 地址映射

(1) 什么是地址映射

将程序地址空间中使用的逻辑地址变换成主存中的物理地址的过程，称为地址映射。



程序地址空间装入主存

(2) 地址映射的时机和类别

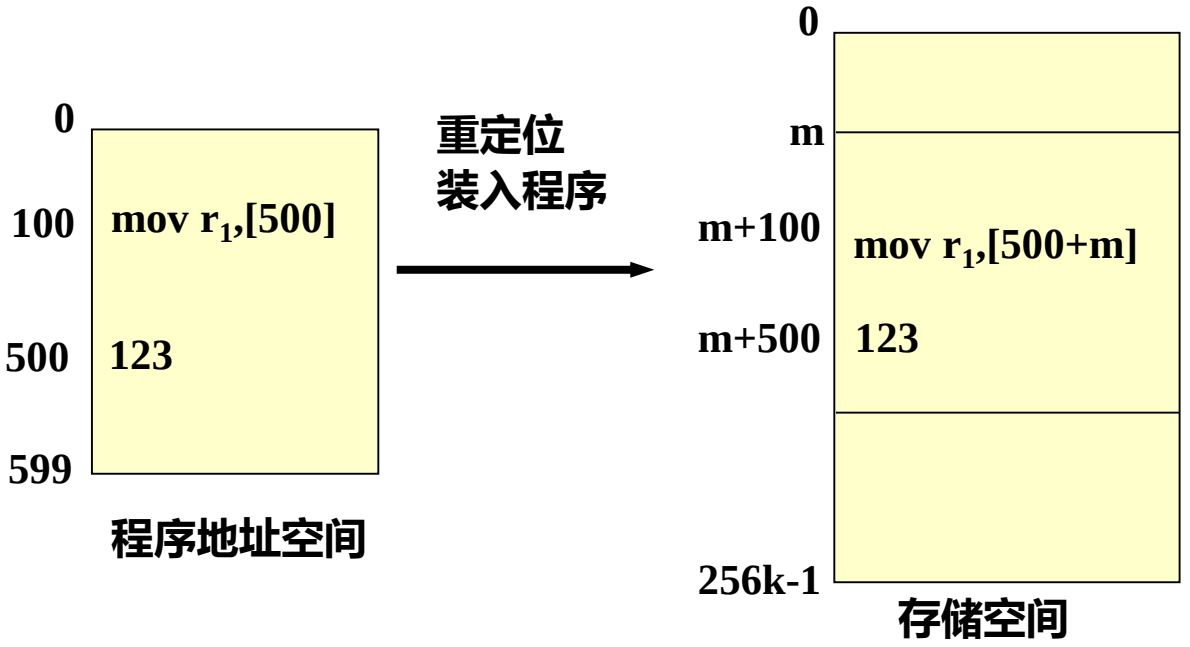
① 编程或编译时确定地址映射关系

在程序编写或程序编译时确定虚、实地址之间的对应关系，结果是一个不能浮动的程序模块。（如果起始地址改变，必须重新编译）

② 在程序装入时确定地址映射关系

在程序装入过程中随即进行的地址变换方式称为**静态地址映射**。

- **不需要硬件支持，易于实现**
- **占用连续的内存空间，装入后不能移动**

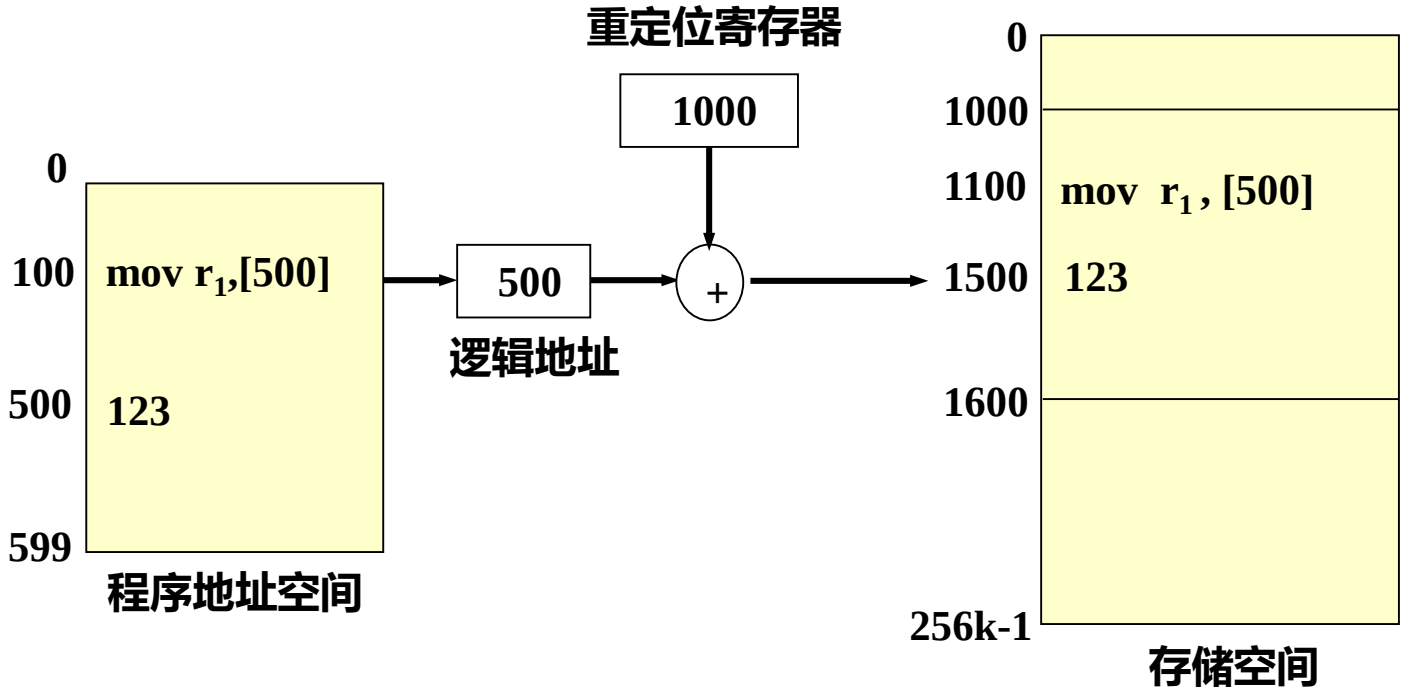


静态地址重定位示意图

主存管理——主存管理功能

③ 在程序运行时确定地址映射关系

在程序执行期间，随着每条指令和数据的访问自动地连续地进行地址映射，这种地址变换方式称为**动态地址映射**。



动态地址重定位示意图

- 可以移动，便于共享，不一定占用连续的内存空间
- 需要硬件支持，算法较复杂

④ 静态地址映射与动态地址映射的区别

静态地址映射	动态地址映射
在程序装入过程中 进行地址映射	在程序执行期间 进行地址映射
需软件 (重定位装入程序)	需硬件地址变换机构 (重定位寄存器)
需花费较多CPU时间	地址变换快
不灵活	灵活

4. 主存分配

(1) 构造分配用的数据结构

主存资源信息块：等待队列；空闲区队列；主存分配程序

(2) 制定策略

① 分配策略 —— 在众多个请求者中选择一个请求者的原则

② 放置策略 —— 在可用资源中，选择一个空闲区的原则

③ 调入策略 —— 决定信息装入主存的时机

预调策略：预先将信息调入主存

请调策略：当需要信息时，将信息调入主存

④ 淘汰策略 —— 在主存中没有可用的空闲区 (对某一程序而言)时，决定哪些信息从主存中移走，即确定淘汰已占用的内存区的原则。

(3) 实施主存分配与回收

主存管理——主存管理功能

5. 主存扩充

存储需求迅速增长

- 程序规模的增长速度远远大于存储器容量的增长速度
- 计算机系统中内存空间往往不够用

内存覆盖？ 内存交换？

虚拟存储：物理内存 + 磁盘

5. 主存扩充

(1) 实现思路

- 程序的全部代码和数据存放在辅存中；
- 将程序当前执行所涉及的那部分程序代码放入主存中；
- 程序执行时，当所需信息不在主存，由操作系统和硬件相配合来完成主存从辅存中调入信息，程序继续执行。

(2) 可行性　　局部性原理

- 时间局部性：一条指令的一次执行和下次执行，一个数据的一次访问和下次访问都集中在一个较短时期内
- 空间局部性：当前指令和邻近的几条指令，当前访问的数据和邻近的几个数据都集中在一个较小区域内
- 分支局部性：一条跳转指令的两次执行，很可能跳到相同的内存位置

5. 主存扩充

不同程序编写方法的局部性特征

例子：页面大小为4K，分配给每个进程的物理页面数为1。在一个进程中，定义了如下的二维数组int A[1024][1024]，该数组按行存放在内存，每一行放在一个页面中

程序编写方法1：

```
for (j = 0; j < 1024; j++)
   for (i = 0; i < 1024; i++)
      A[i][j] = 0;
```

程序编写方法2：

```
for (i=0; i<1024; i++)
   for (j=0; j<1024; j++)
         A[i][j] = 0;
```

(3)虚拟存储器

① 什么是虚拟存储器

由操作系统和硬件相配合来完成主存和辅存之间的信息的动态调度。这样的计算机系统好像为用户提供了一个其存储容量比实际主存大得多的存储器，这个存储器称为虚拟存储器。

② 虚拟存储器的核心

- **逻辑地址与物理地址分开**
- **存储空间与虚地址空间分开**
- **提供地址变换机构**

③ 实现虚拟存储器的物质基础

- **有相当容量的辅存：**足以存放应用程序的虚地址空间
- **有一定容量的主存：**存放进入主存的多进程的信息
- **地址变换机构**

6. 存储保护

(1) 什么是存储保护

在多用户环境中，主存储器按区分配给各用户程序使用。

为了互不影响，必须由硬件 (软件配合)保证各用户程序只能在给定的存储区域内活动，这种措施叫做存储保护。

(2) 实现方法

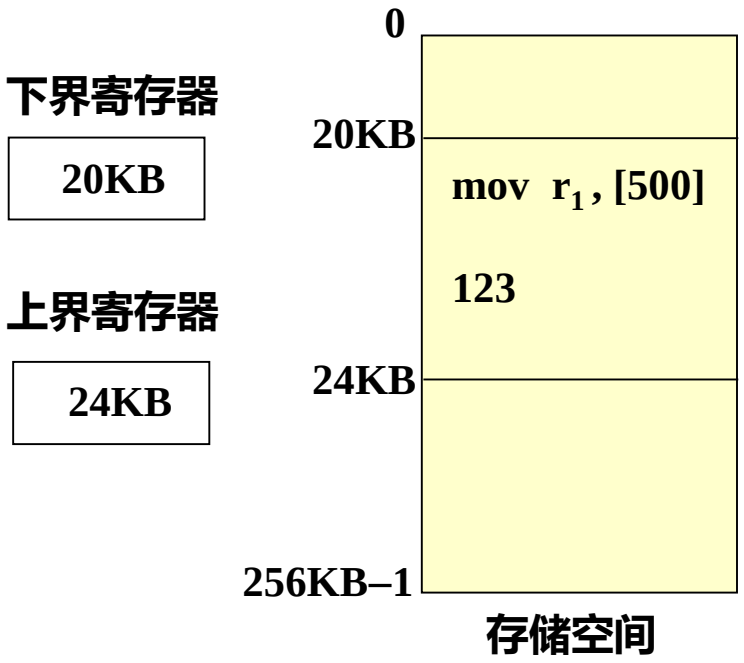
- 界地址保护
- 存储键保护

主存管理——主存管理功能

界地址保护

① 上下界防护

例：程序大小为4KB，主存首址为20KB。



界限寄存器保护示意图

- 如何设置上下界寄存器内容？
- 如何判断是否越界？

若 20KB≤ 物理地址 < 24KB

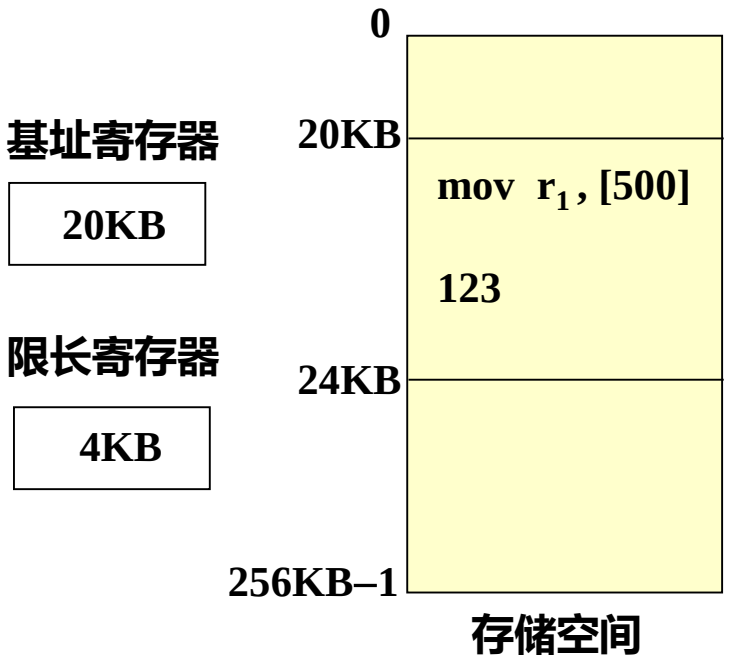
允许访问；

否则发生越界中断

主存管理——主存管理功能

② 基地址、限长防护

例：程序大小为**4KB**，主存首址为**20KB**。



界限寄存器保护示意图

- **如何设置基址、限长寄存器内容？**
- **如何判断是否越界？**

  若 逻辑地址 < 4KB

  允许访问；

  否则发生越界中断

分区存储管理

主存管理——分区存储管理

- **多个程序共享内存空间，分为固定分区和可变分区**
- **固定分区存储管理(静态分区)**
  - 把内存预先划分成多个分区，分区大小可以相同或不同
  - 分区个数固定，分区大小固定
  - 一个分区装入一个作业

早期的IBM OS/MFT采用固定分区方法
——具有固定任务数的多道程序系统

数据结构？分配程序？性能情况？

主存管理——分区存储管理

1. 动态分区分配

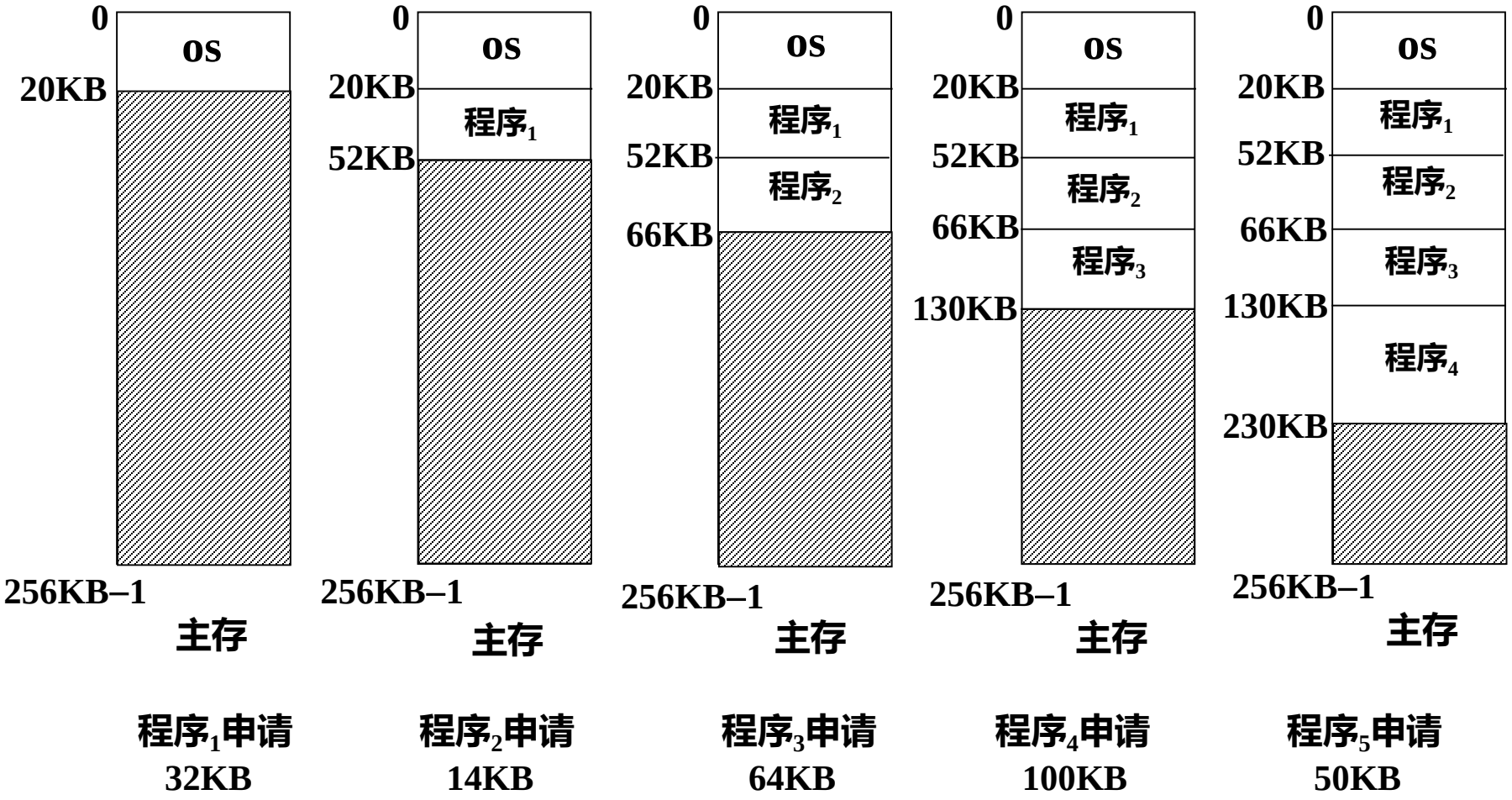
(1) 什么是动态分区分配

在处理程序的过程中，建立分区，依用户请求的大小分配分区。

(2) 动态分区的分配、回收过程

① 动态分区的分配过程

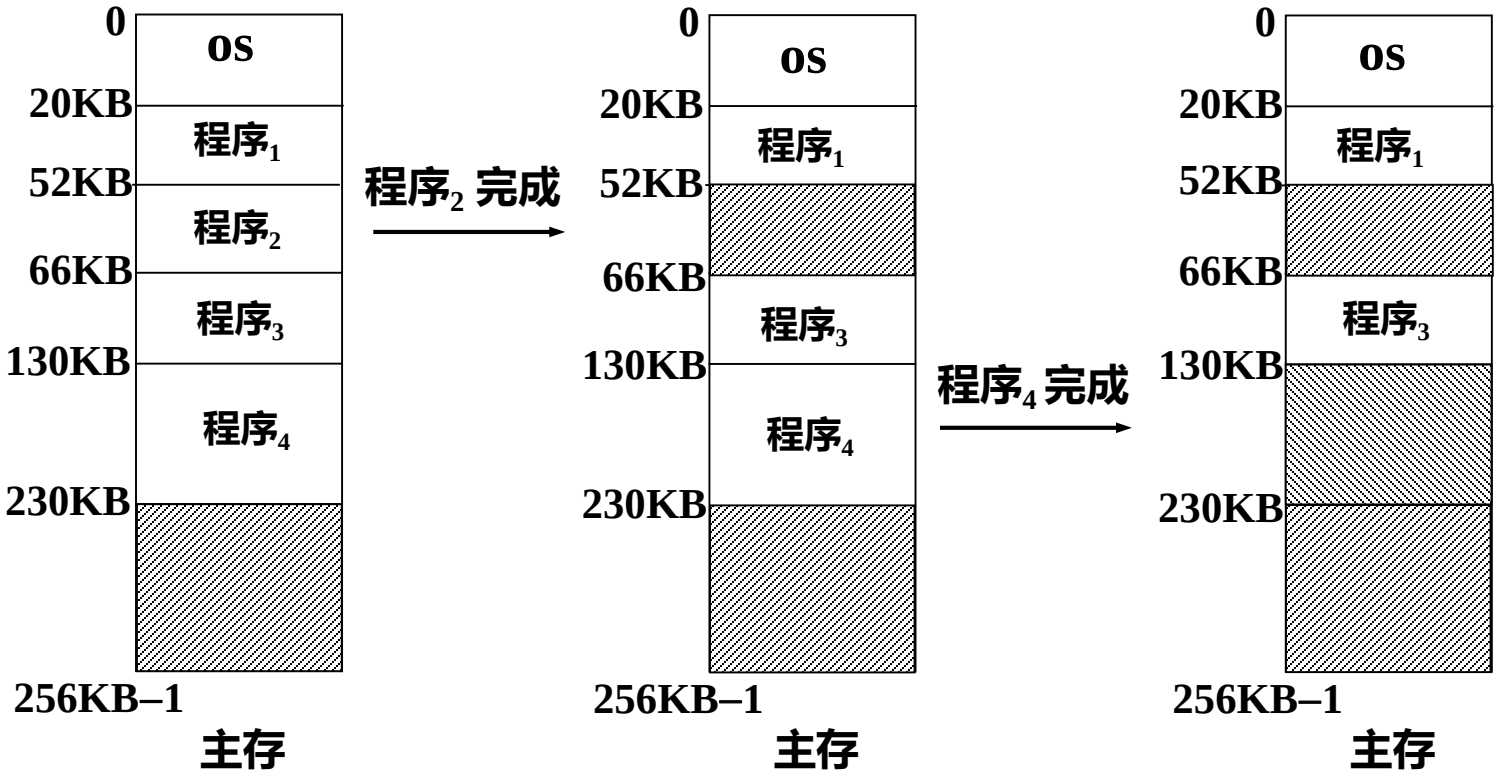
主存管理——分区存储管理



动态分区分配示意图

主存管理——分区存储管理

② 动态分区的回收过程



动态分区分配中的存储区的释放

(3) 分区分配数据结构

① 主存资源信息块 (M_RIB)

② 分区描述器 (PD)

flag：为 0 —— 空闲区

为 1 —— 已分配区

size：分区大小

next：空闲区——自由主存队列中的勾链字

已分配区——此项为零

M_RIB

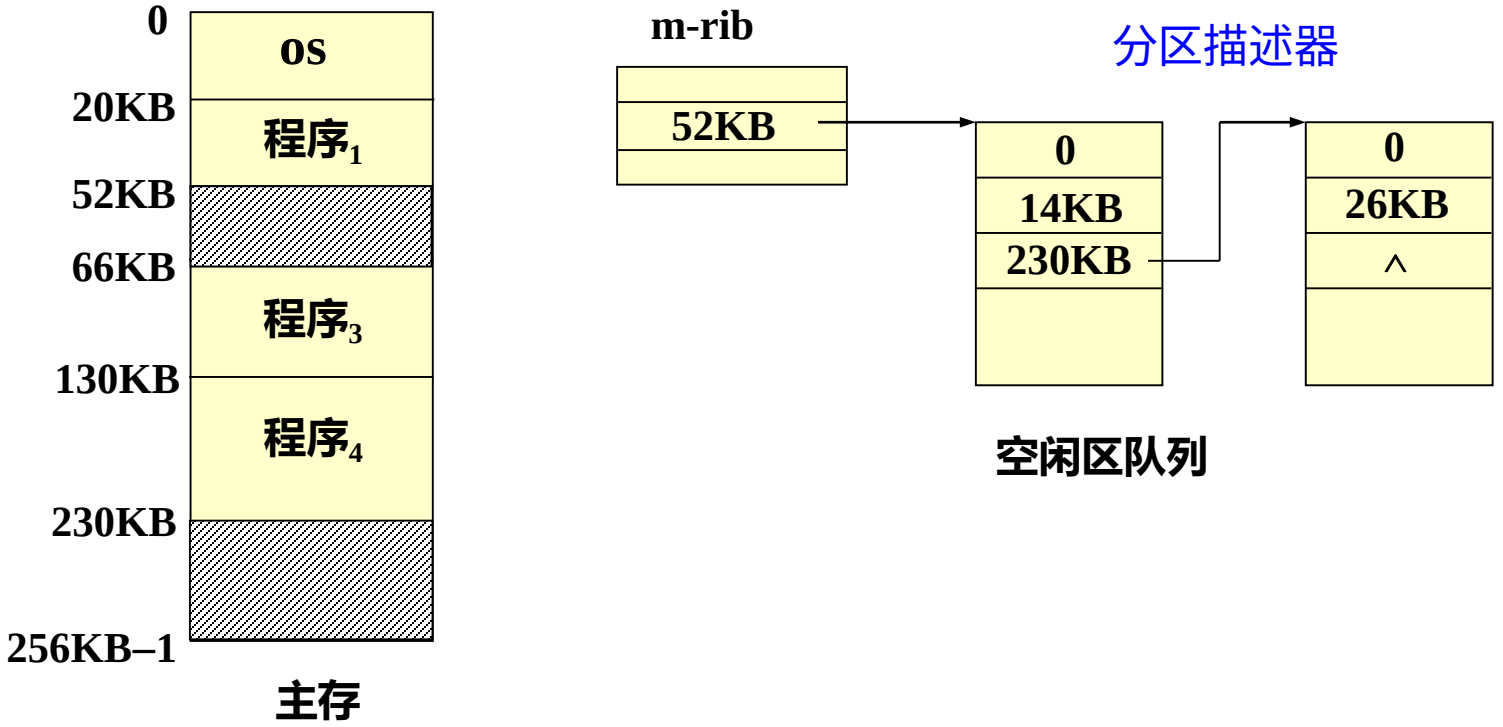
等待队列头指针
空闲区队列头指针
主存分配程序入口地址

PD

分配标志	flag
大小	size
勾链字	next

主存管理——分区存储管理

③ 空闲区队列结构



动态分区的空闲区队列结构

主存管理——分区存储管理
2. 分区的分配与回收

(1) 分区分配思路

① 寻找空闲块

依申请者所要求的主存区的大小，分区分配程序在自由主存队列中找一个满足用户需要的空闲块；

② 若找到了所需的空闲区，有两种情况

i 空闲区与要求的大小相等，将该空闲区分配并从队列中摘除；

ii 空闲区大于所要求的的大小，将空闲区分为两部分：一部分成为已分配区，建立已分配区的描述器；剩下部分仍为空闲区。返回所分配区域的首址；

③ 否则，告之不能满足要求。

(2) 分区回收思路

① 检查释放分区 (即为回收分区)在主存中的邻接情况

若上、下邻接空闲区，则合并，成为一个连续的空闲区

② 若回收分区不与任何空闲区相邻接

建立一个新的空闲区，并加入到空闲区队列中。

3. 放置策略

(1) 什么是放置策略

选择空闲区的策略，称为放置策略。

常用的放置策略——

首次匹配 (首次适应算法)

最佳匹配 (最佳适应算法)

最坏匹配 (最坏适应算法)

(2) 首次适应算法

① 什么是首次适应算法

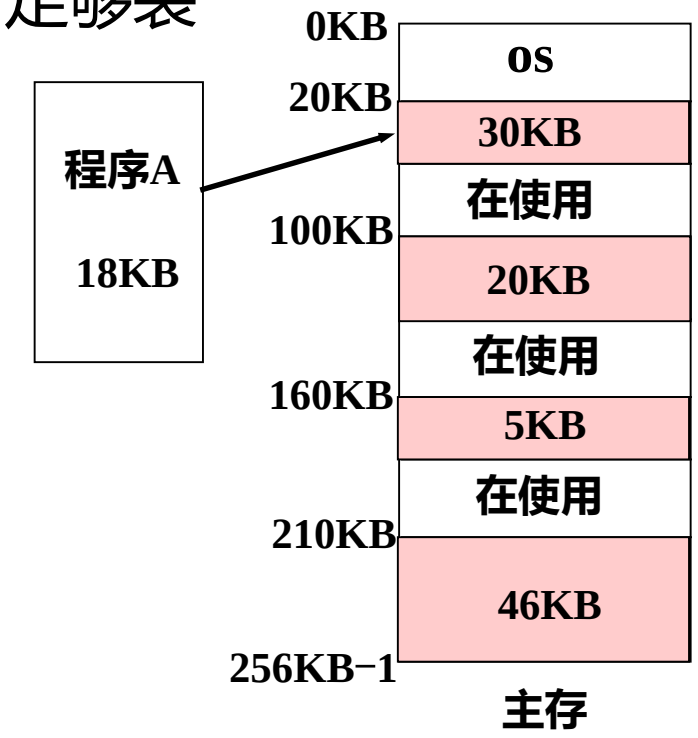
首次适应算法是将输入的程序放置到主存里第一个足够装入它的地址最低的空闲区中。

② 空闲区队列结构

空闲区地址由低到高排序

③ 首次适应算法的特点

尽可能地利用存储器中低地址的空闲区，而尽量保存高地址的空闲区。



三种放置策略的图解

首次适应算法

(3) 最佳适应算法

① 什么是最佳适应算法

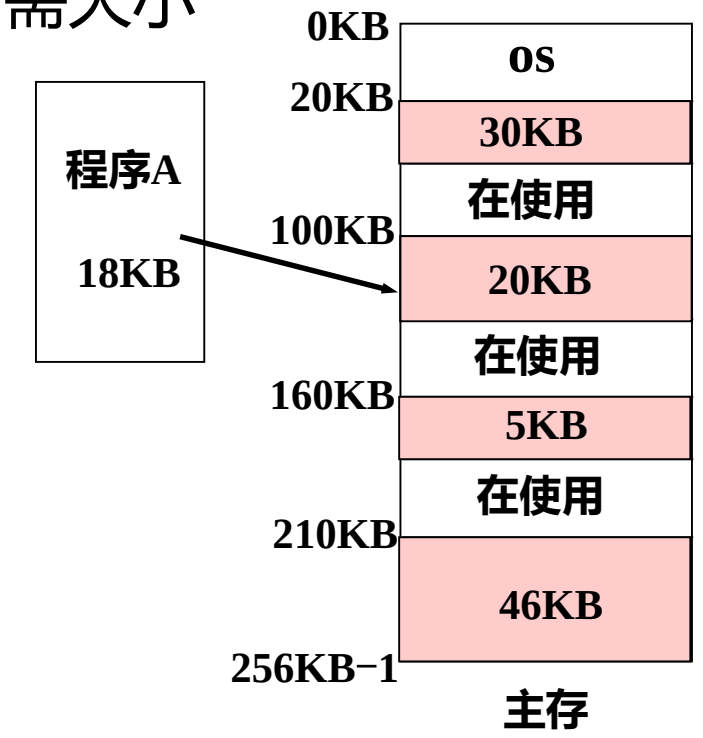
最佳适应算法是将输入的程序放置到主存中与它所需大小最接近的空闲区中。

② 空闲区队列结构

空闲区大小由小到大排序

③ 最佳适应算法的特点

尽可能地利用存储器中小的空闲区，而尽量保存大的空闲区。



三种放置策略的图解

最佳适应算法

(4) 最坏适应算法

① 什么是最坏适应算法

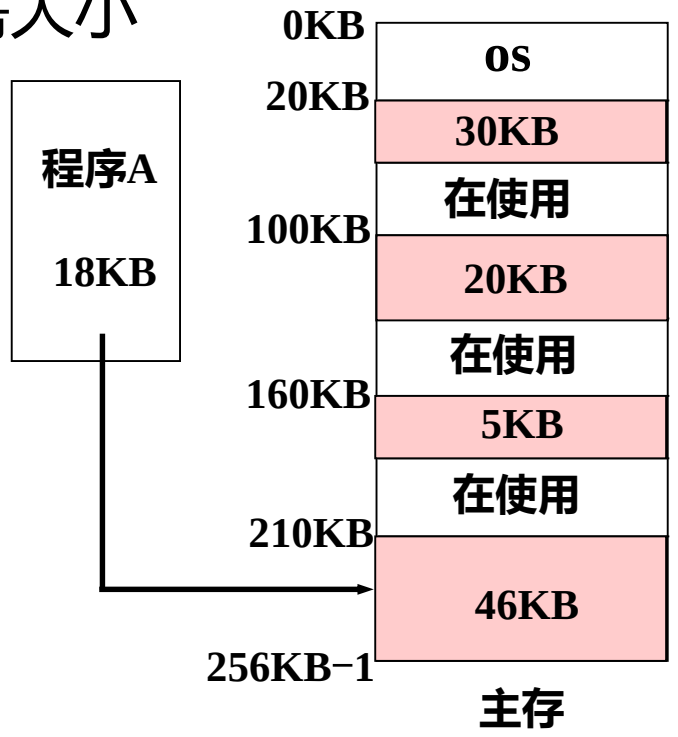
最坏适应算法是将输入的程序放置到主存中与它所需大小差距最大的空闲区中。

② 空闲区队列结构

空闲区大小由大到小排序

③ 最坏适应算法的特点

尽可能地利用存储器中大的空闲区。



三种放置策略的图解

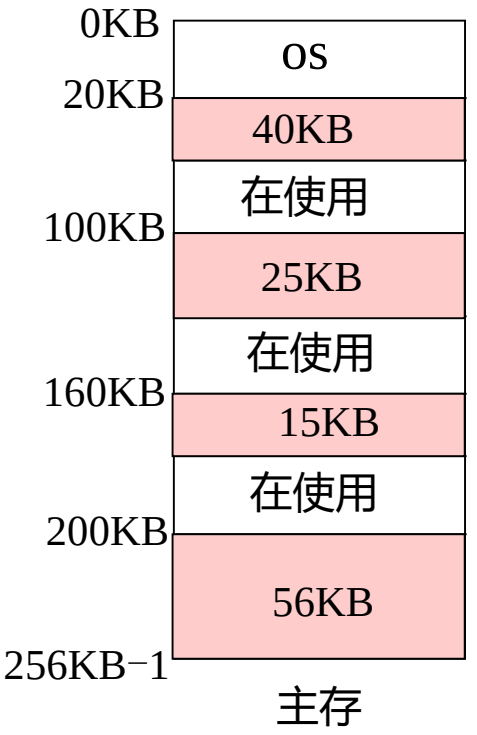
最坏适应算法

(5) 三种放置策略的讨论

① 题例

程序A要求23KB；程序B要求28KB；程序C要求42KB。

1. 用首次适应算法、最佳适应算法、最坏适应算法来处理程序序列，看哪种算法合适。
2. 分析三种算法的应用效果、可能存在的问题和程序的运行开销。



4. 采用分区存储管理的OS如何存储空闲区链表（自由主存队列）？

（1）集中存储

在OS空间中开辟一块专有的空间来存储所有PD结构。

优缺点?

（2）分散存储

将PD分散存储到它们所对应的存储区域。

如何实现?

优缺点?

5. 碎片问题及拼接技术

(1) 什么是碎片问题

在已分配区之间存在着的一些没有被充分利用的空闲区。

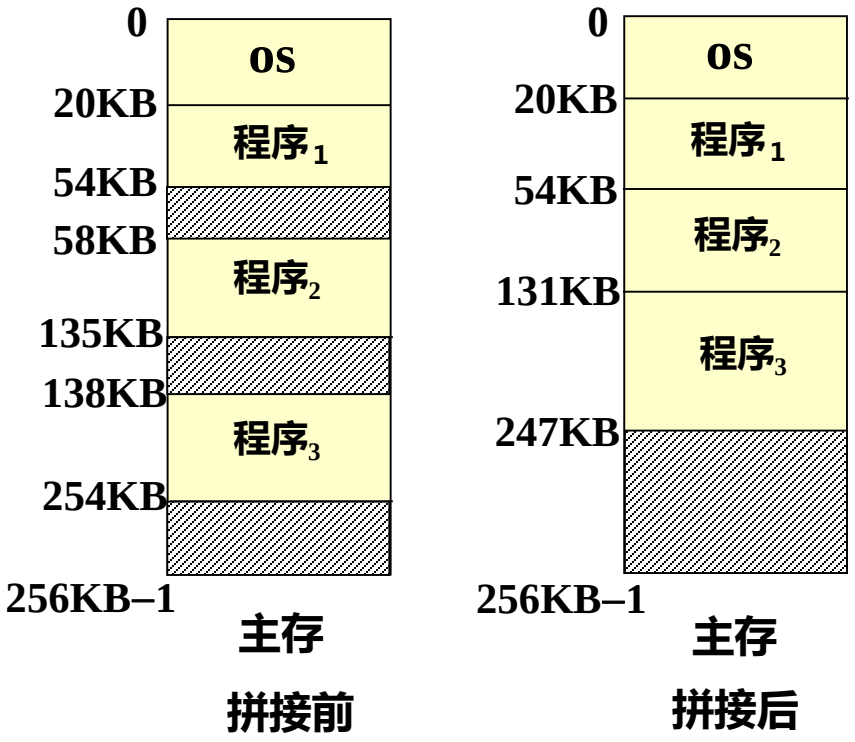
外部碎片。如何解决碎片问题?

(2) 什么是拼接技术

所谓拼接技术是指移动存储器中某些已分配区中的信息，使本来分散的空闲区连成一个大的空闲区。

问题:

- 系统开销太大
- 不能实现对主存的扩充



分区分配中的存储区拼接

主存管理——分区存储管理

(3) 伙伴系统

把一个大的存储块分为大小相等的两个小存储区（互为伙伴，大小均为2的K次幂）

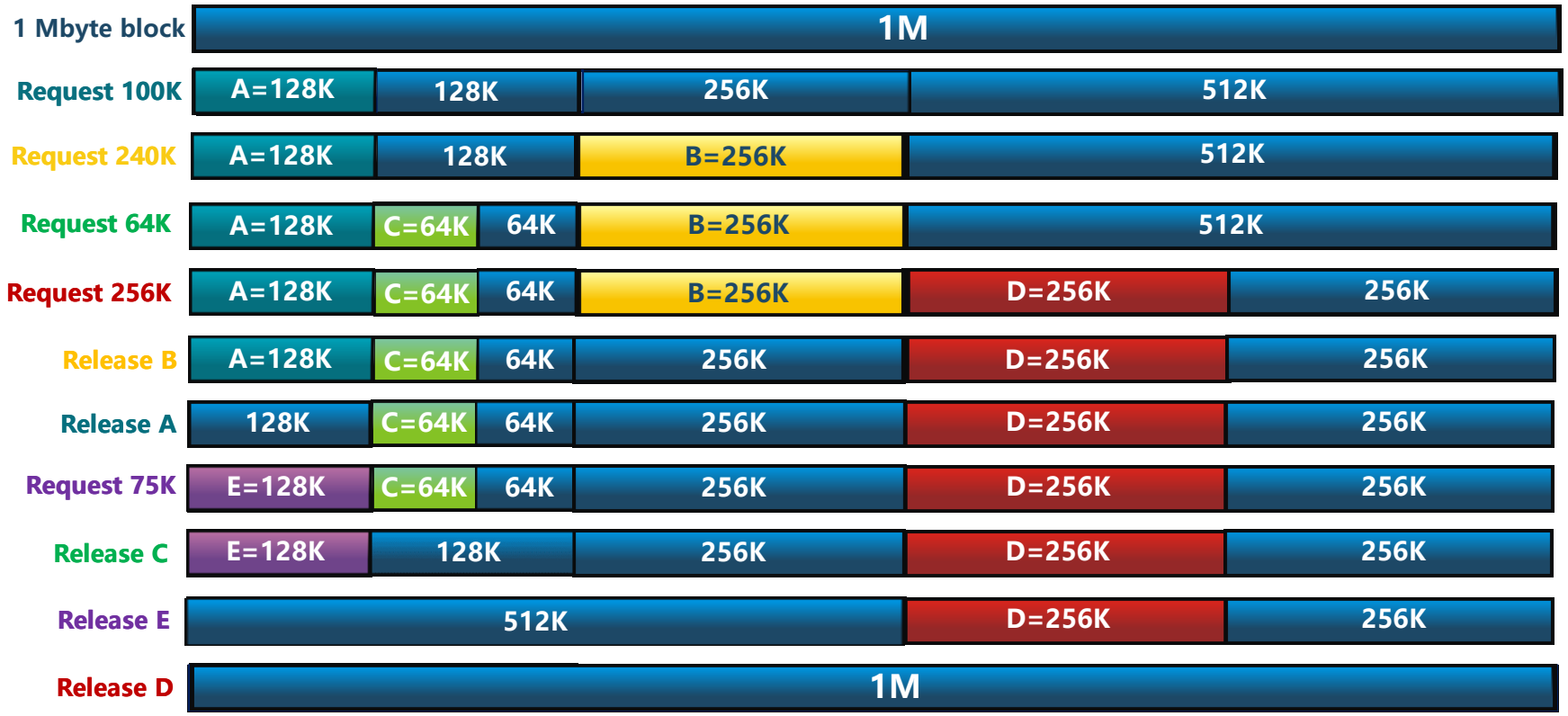
- 整个可分配的分区大小2^K
- 需要的分区大小(s)为$2^K-1 < s \leq 2^K$ 时，把整个块分配给该进程；
  - 如$s \leq 2^{i-1}$，将大小为2^i的当前空闲分区划分成两个大小为2^{i-1}的空闲分区
  - 重复划分过程，直到$2^i-1 < s \leq 2^i$，并把一个空闲分区分配给该进程

主存管理——分区存储管理

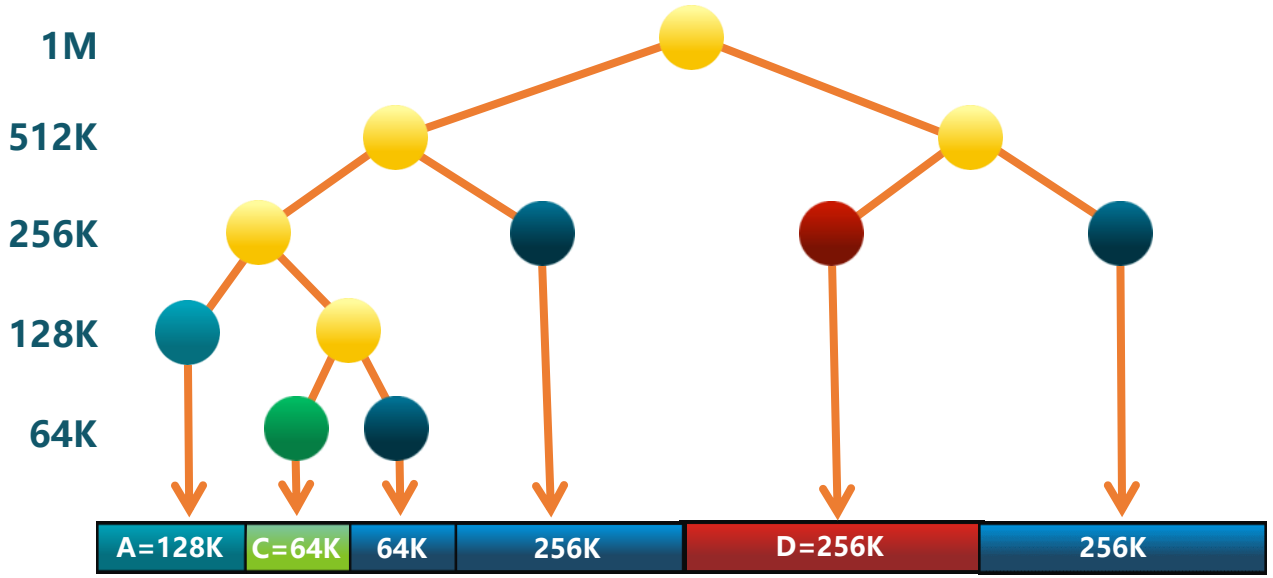
伙伴系统的实现

- **数据结构**
  - 空闲块按大小和起始地址组织成二维数组
  - 初始状态：只有一个大小为2^K的空闲块
- **分配过程**
  - 由小到大在空闲块数组中找最小的可用空闲块
  - 如空闲块过大，对可用空闲块进行二等分，直到得到合适的可用空闲块
- **释放过程**
  - 把释放的块放入空闲块数组
  - 合并满足合并条件的空闲块
- **合并条件**
  - 大小相同2^i
  - 地址相邻
  - 低地址空闲块起始地址为2^{i+1}的位数

伙伴系统中的内存分配

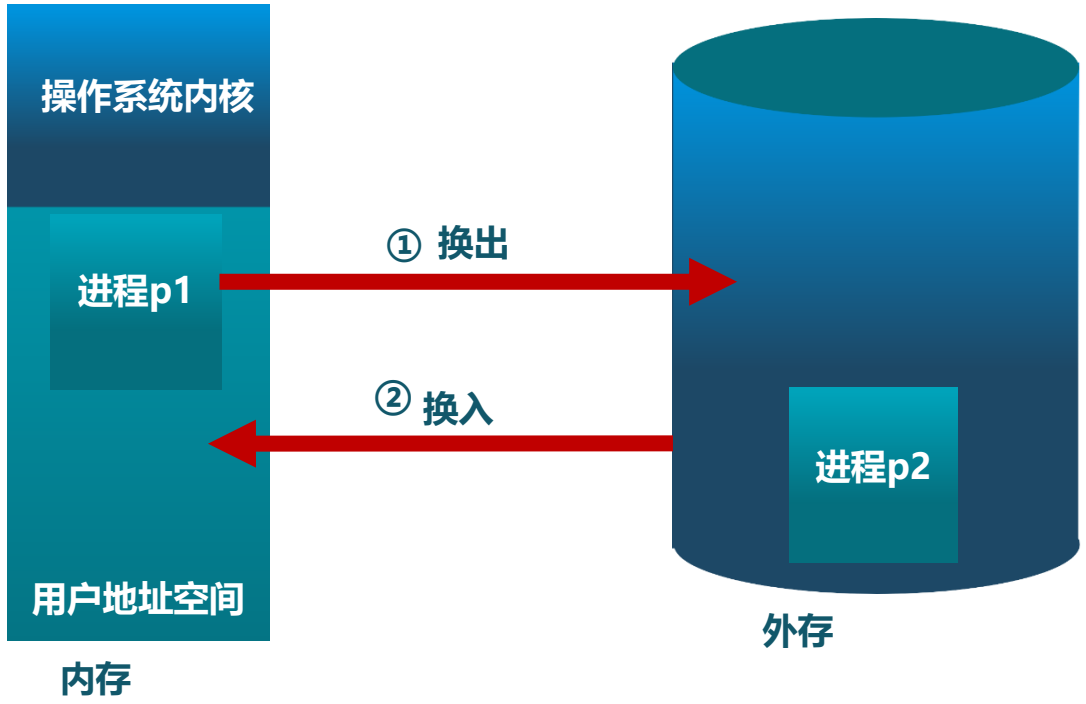

1 Mbyte block
1M
Request 100K
A=128K
128K
256K
512K
Request 240K
A=128K
128K
B=256K
512K
Request 64K
A=128K
C=64K
64K
B=256K
512K
Request 256K
A=128K
C=64K
64K
B=256K
D=256K
256K
Release B
A=128K
C=64K
64K
256K
D=256K
256K
Release A
128K
C=64K
64K
256K
D=256K
256K
Request 75K
E=128K
C=64K
64K
256K
D=256K
256K
Release C
E=128K
128K
256K
D=256K
256K
Release E
512K
D=256K
256K
Release D
1M

伙伴系统中的内存分配



(4) 对换技术

选择内存中的某个进程暂时移出到磁盘，腾出空间给其他进程，同时把磁盘中的某个进程换进主存使其投入运行



6. 碎片问题（内存利用率较低）如何解决?

能否将进程分配到几个不连续的内存区域，进程仍能正确执行

连续分配——〉离散分配（非连续分配）

如何确定离散分配中的内存分块大小?

分页存储管理

分段存储管理

动态分区存储管理的分析

- 有助于多道程序设计
- 不需过多硬件，仅需界地址寄存器进行存储保护
- 算法简单，易于实现
- 碎片问题
- 分区大小受主存容量限制，无法扩充主存容量

页式存储管理

主存管理——页式存储管理

1. 页式系统的基本概念

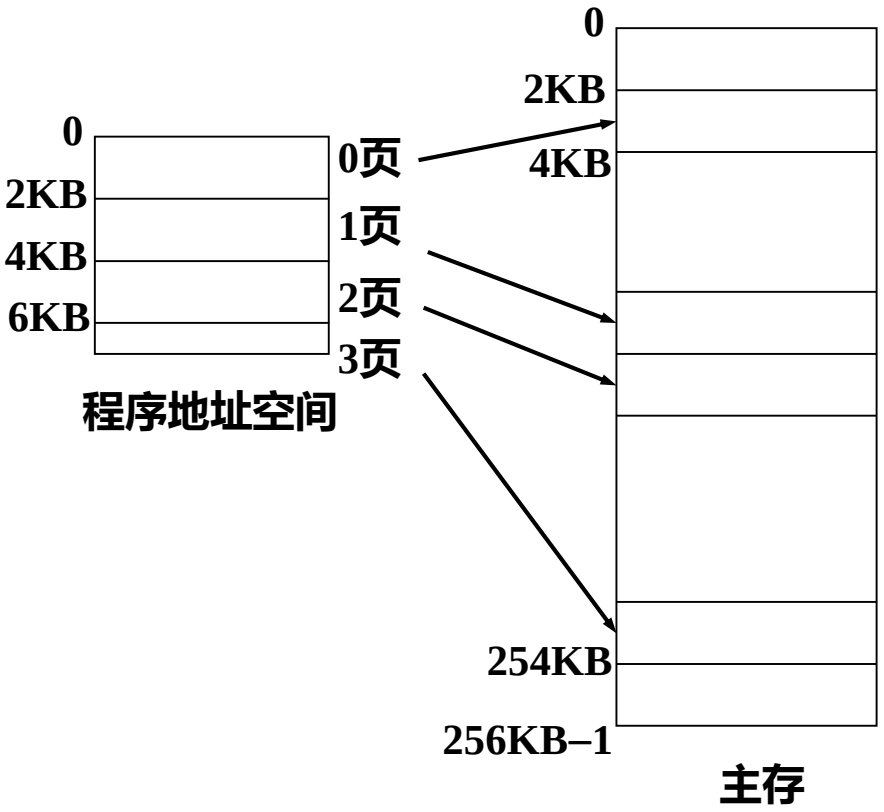
(1) 页面

程序的地址空间被等分成大小相等的片，称为页面，又称为虚页。

(2) 主存块

主存被等分成大小相等的片，称为主存块，又称为实页。

(3) 页面与主存块的关系



等分主存和虚地址空间

(4) 页表

① 什么是页表

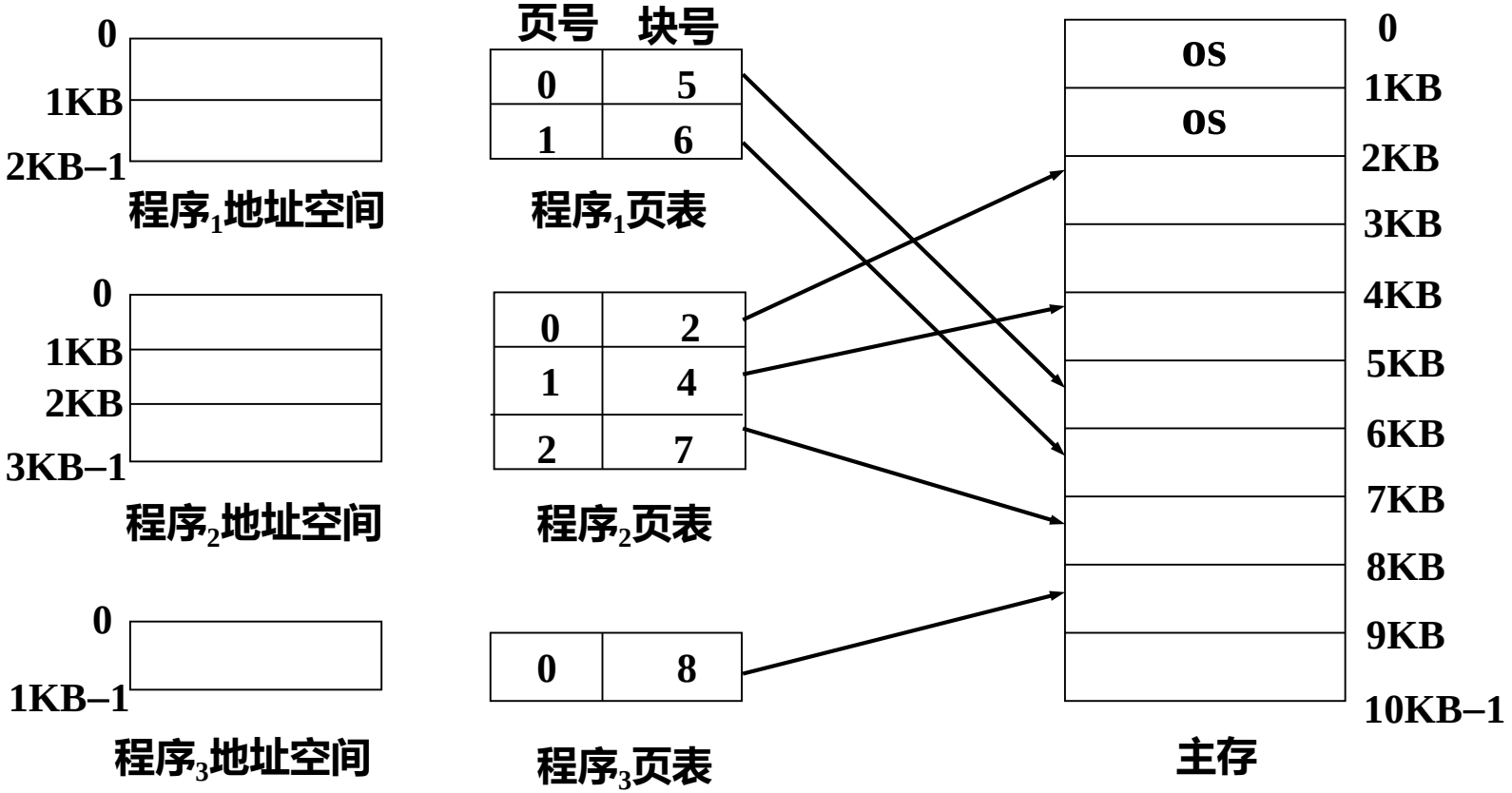
为了实现从地址空间到物理主存的映象，系统建立的记录页与内存块之间对应关系的地址变换的机构称为页面映像表，简称页表。

② 页表的组成

i 高速缓冲存储器： 地址变换速度快，但成本较高

ii 主存区域： 地址变换速度比硬件慢，成本较低

(5) 分页映像存储的例子



分页映像存储

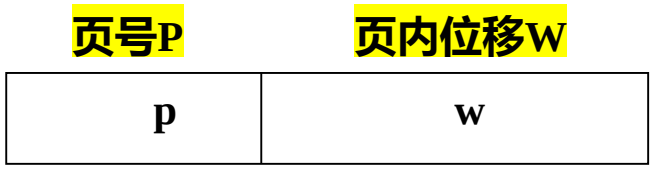
2. 页式地址变换

(1) 页表

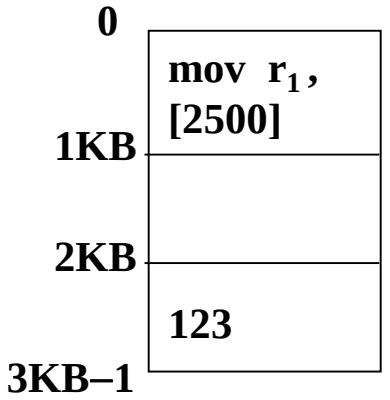
记录页与块之间对应关系的**地址变换的机构。**

(2) 虚地址结构

当CPU给出的虚地址长度为16位，页面大小为1KB时，在分页系统中地址结构的格式如下：



虚地址结构



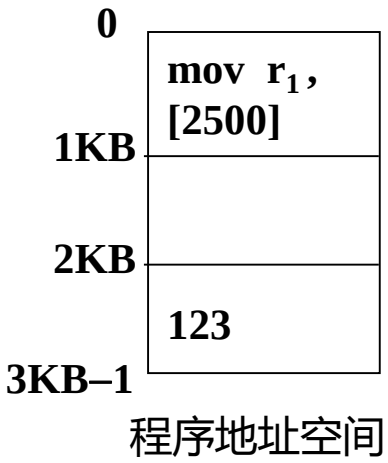
程序地址空间

(3) 页式地址变换

① 页式地址变换的例

程序地址空间中，设100号单元处有如下指令：

mov r1,[2500]。当这条指令执行时，如何进行正确的地址变换。



2500 → 2×1024 + 452

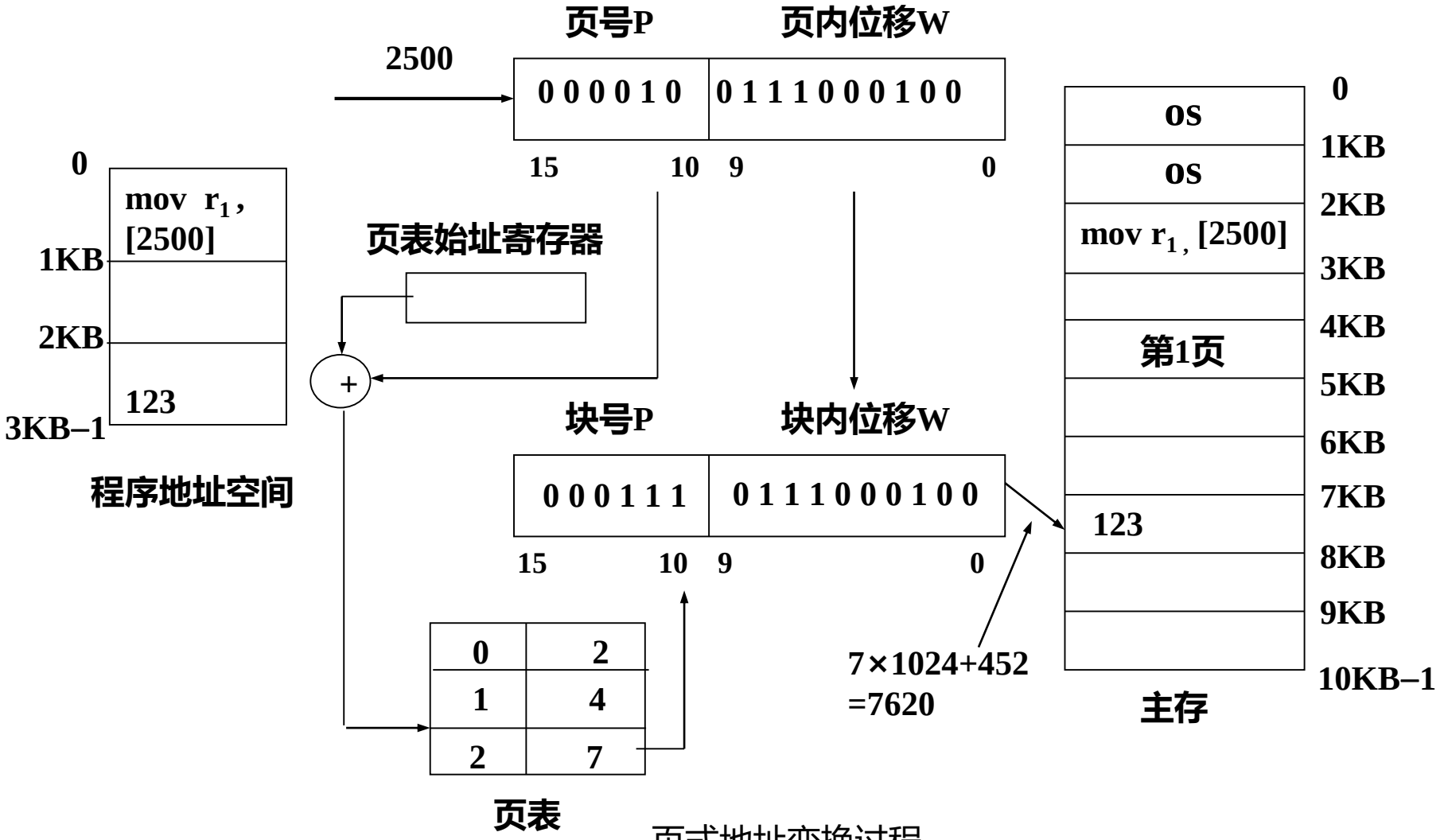
p=2 w=452

0000100111000100 000010 0111000100

页号 页内位移

主存管理——页式存储管理

② 页式地址变换过程



页式地址变换过程

主存管理——页式存储管理

③ 页式地址变换步骤

i CPU给出操作数地址 (为2500)；

ii 由分页机构自动地把逻辑地址分为两部分，得到页号p和页内相对位移w (p =2， w =452)；

iii 根据页表始址寄存器指示的页表始地址，以页号为索引，找到第2页所对应的块号 (为7)；

iv 将块号b和页内位移量w拼接在一起，就形成了访问主存的物理地址 (7×1024+452=7620)

(4) 页式存储管理的性能问题

① 内存访问的性能问题

一个页面的访问开销?

② 页表大小问题

假设64位机器中每页1024字节，页表会有多大?

③ 如何处理?

缓存

间接访问

(5) 采用联想存储器加快查表速度

① 什么是联想存储器

高速、小容量半导体存储部件，又称缓冲存储器。

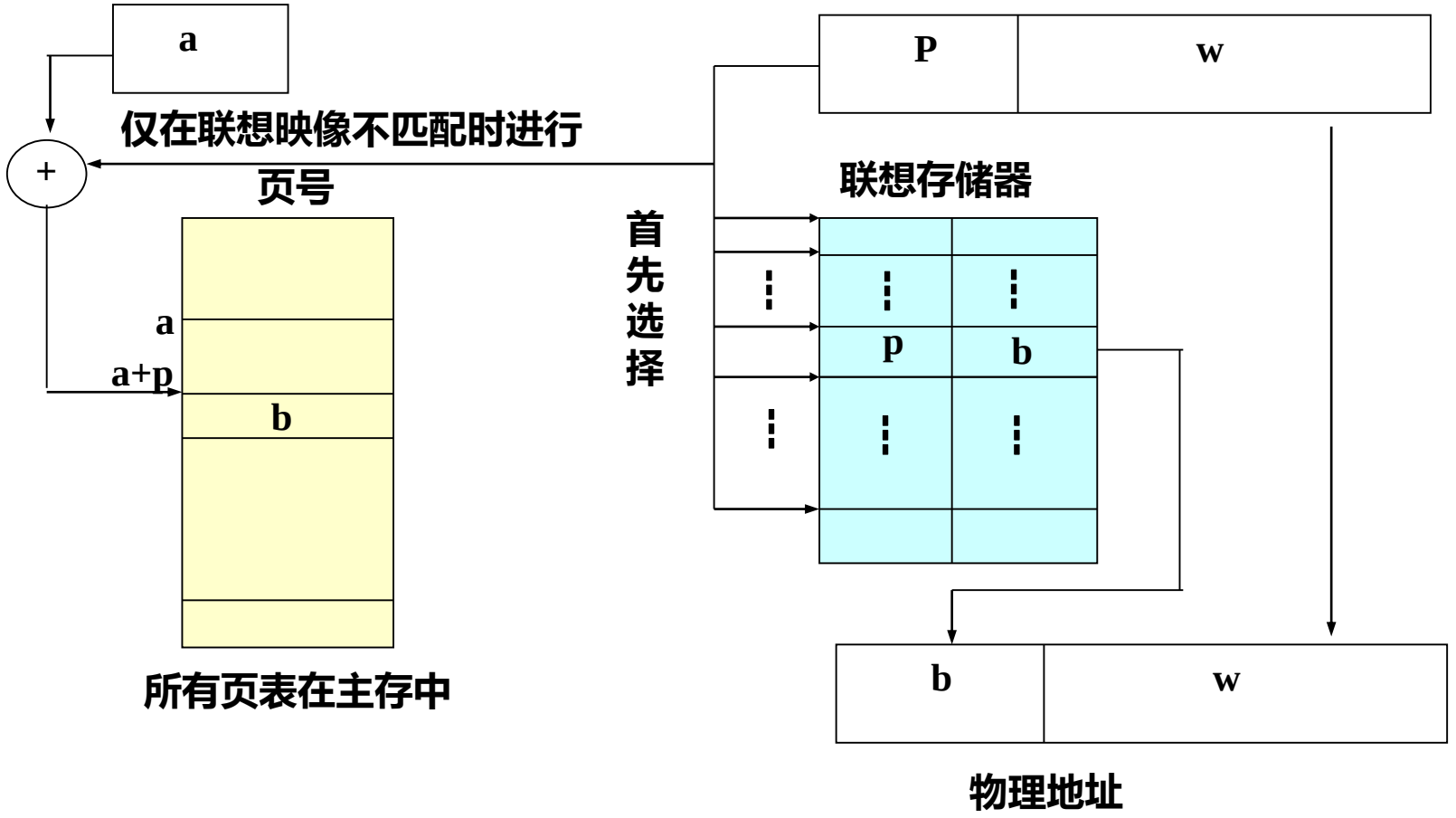
② 快表(Translation Look-aside Buffer, TLB)

在缓冲存储器中存放正在运行的进程当前用到的页号和对应的块号，又称为快表。

- TLB 使用关联存储实现，具备快速访问性能
- 如果TLB命中，物理页号可以很快被获取
- 如果TLB未命中，对应的表项被更新到TLB中

主存管理——页式存储管理

③ 利用快表进行地址映射

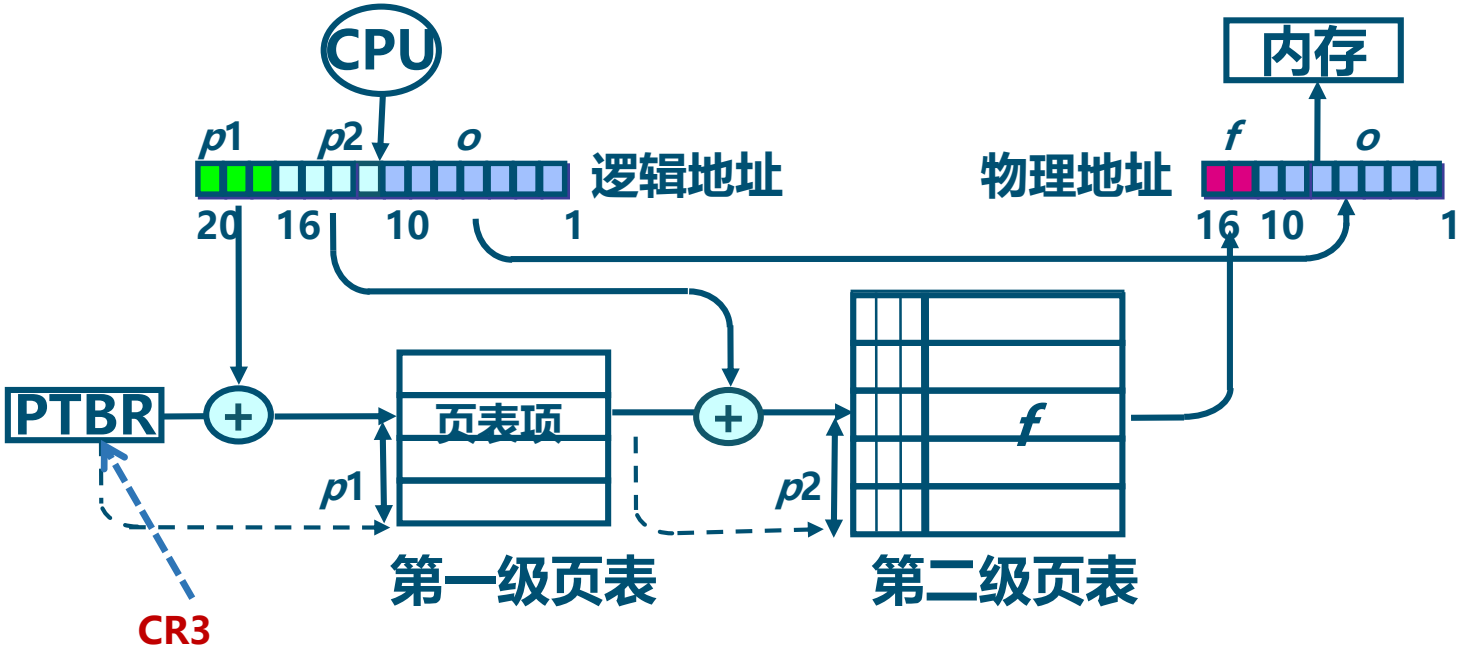


联想寄存器和主存页表相结合的分页地址变换

(6) 多级页表

通过间接引用，将页号分成k级

建立页表“树”，减少每级页表的长度



（7）大地址空间问题

- 大地址空间系统，多级页表也很繁琐
- 逻辑（虚拟）地址空间增长速度快于物理地址空间

反置页表：

- 页表不再与逻辑地址空间的大小相对应
- 页表与物理地址空间的大小相对应

页表大小与逻辑地址空间无关，相对于物理内存很小

页表对调后，根据内存块号找页号

如何进行地址映射？如何查找页号？

3. 请调页面的机制

(1) 两种页式系统

① 简单页式系统： 装入一个程序的全部页面才能投入运行。

② 请求页式系统： 装入一个程序的部分页面即可投入运行。

请求页式系统需解决的问题

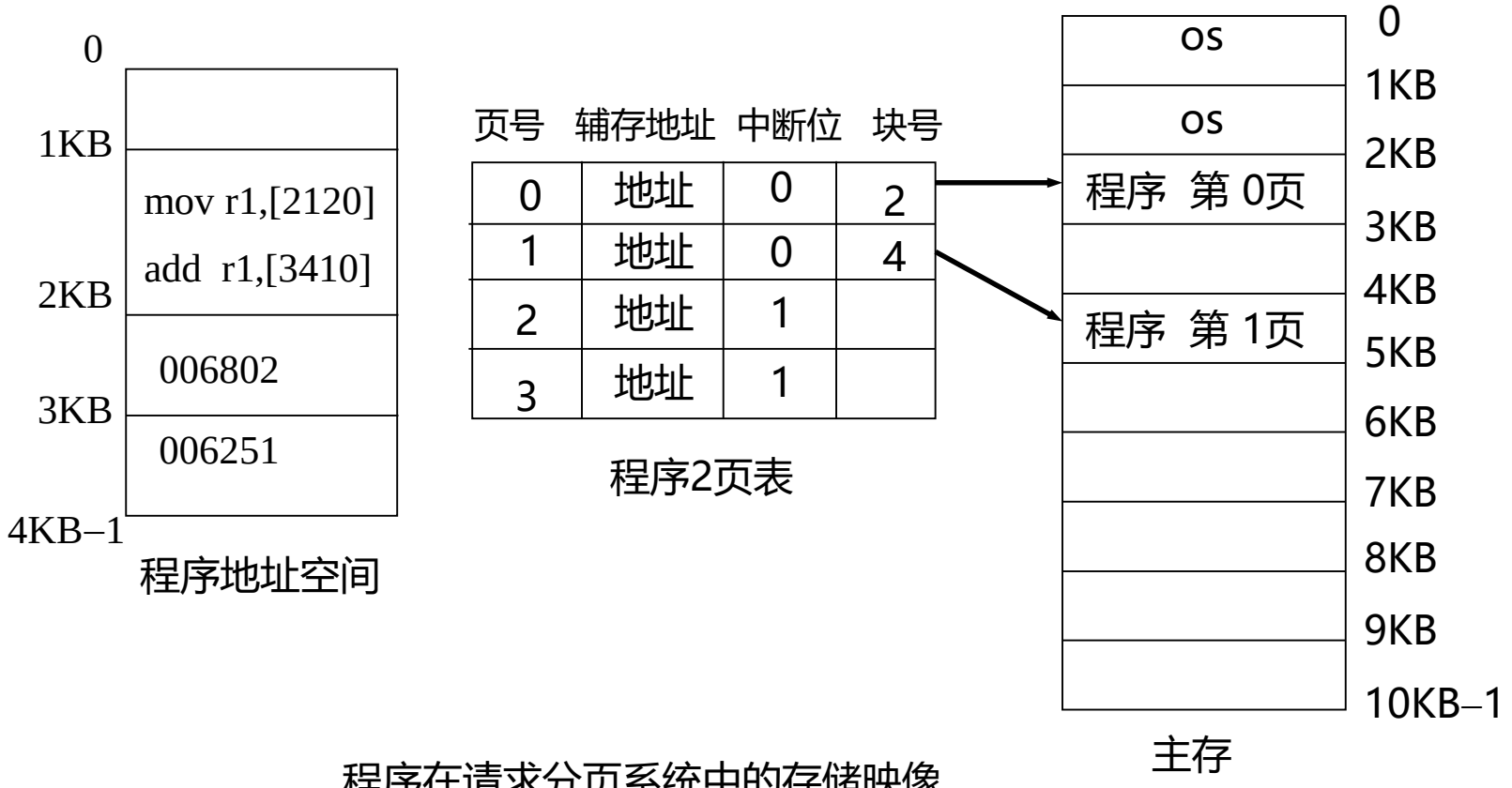
(2) 扩充页表功能

页号	主存块号	中断位	辅存地址

- 中断位i ： 标识该页是否在主存若i=1，表示此页不在主存；若i=0，表示该页在主存
- 辅存地址： 该页面在辅存的位置

(3) 缺页处理

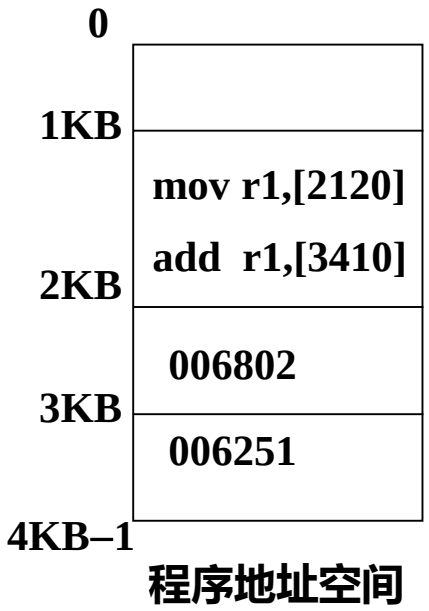
① 程序在请求分页系统中的存储映像



程序在请求分页系统中的存储映像

② 缺页处理的例子

程序的主存块数为 $m_2=3$，讨论程序执行“mov r_1，[2120]”指令时的情况。

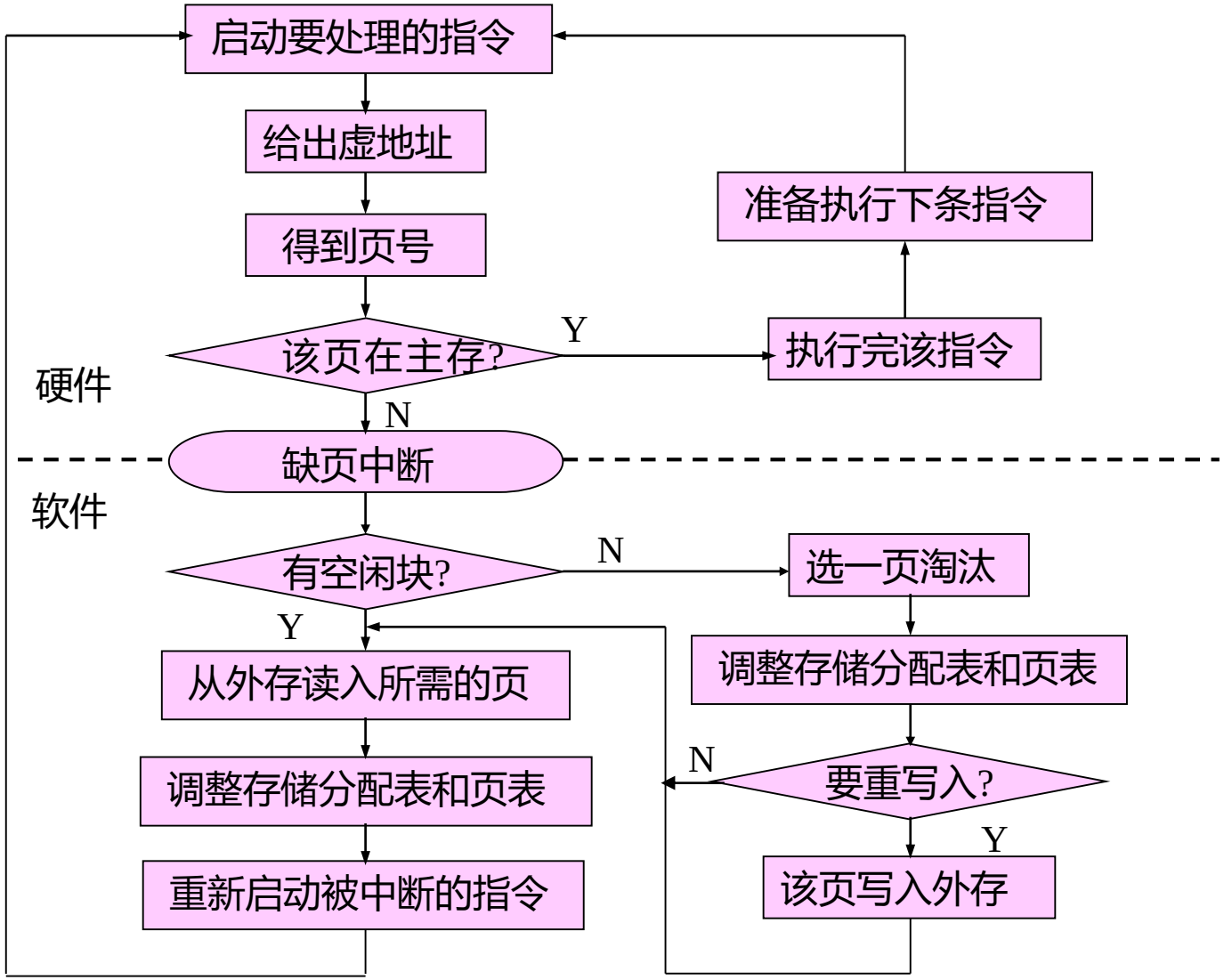


页号	辅存地址	中断位	块号
0	地址	0	2
1	地址	0	4
2	地址	1	
3	地址	1	

- CPU产生的虚地址为2120
- 分页机构得 p=2，w=72
- 查页表。该页中断位i=1
- 发生缺页中断！
- 如主存中有空白块，且n<m 则直接调入
- 如主存中无空白块，或n ≥ m ，则需淘汰主存中的一页

主存管理——页式存储管理

③ 缺页处理过程图示



指令执行步骤和缺页中断处理过程

4. 淘汰机制与策略

(1) 什么是淘汰策略

用来选择淘汰哪一页的规则叫做置换策略，或称淘汰算法。

如何决定淘汰哪一页?

(2) 扩充页表功能

页号	主存块号	中断位	辅存地址	引用位	改变位

① 引用位 —— **标识该页最近是否被访问**

为“0”—— 该页没有被访问；为“1”—— 该页已被访问

② 改变位 —— **表示该页是否被修改**

为“0”—— 该页未被修改；为“1”—— 该页已被修改

(3) 颠簸

颠簸(thrashing)，又称为“抖动”。

简单地说，导致系统效率急剧下降的主存和辅存之间的频繁页面置换现像称为“抖动”。

(4) 缺页中断率

- 假定程序p共有n页，系统分配m块，有 1≤m≤n；
- 若程序p在运行中：成功的访问次数为s，不成功的访问次数为f；
- **缺页中断率： f′=f/ (s+ f)**

f′= f (r，m，p)；

r：置换算法；m：系统分配的块数； p：程序特征

(5) 常用的置换算法

- 局部页面置换：
  - 置换页面的选择范围仅限于当前进程占用的物理页面内
- 全局页面置换：
  - 置换页面的选择范围是所有可换出的物理页面
  - 为进程分配可变数目的物理页面

最佳算法、先进先出淘汰算法、最久未使用淘汰算法

(5) 常用的置换算法

① 最佳算法(OPT算法)

当要调入一新页而必须先淘汰一旧页时，所淘汰的那一页应是以后不再要用的，或者是在最长的时间以后才会用到的那页。

主存管理——页式存储管理

② 先进先出淘汰算法(FIFO算法)

i 什么是先进先出淘汰算法

总是选择在主存中居留时间最长 (即最早进入主存)的一页淘汰。

ii 先进先出淘汰算法的实现

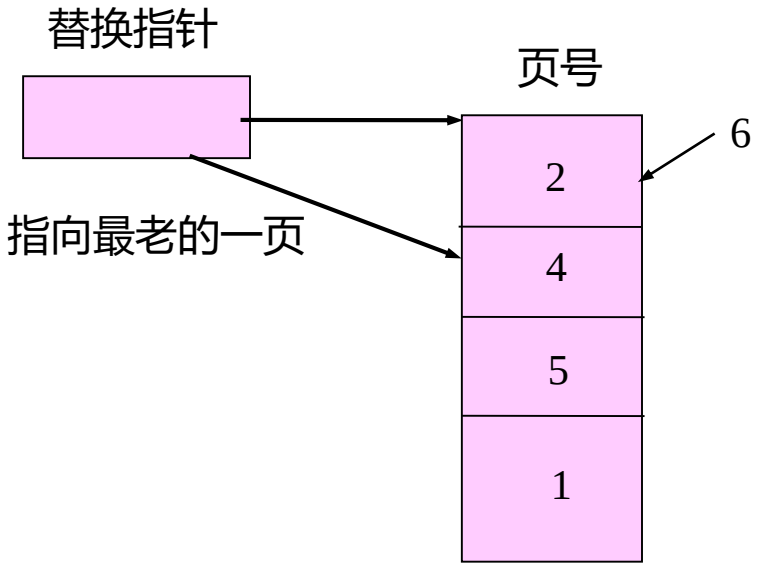
- 建立一个页面进入主存的先后次序表；
- 建立一个替换指针，指向最早进入主存的页面；
- 当需要置换一页时，选择替换指向的那一页，然后调整替换指针的内容。

主存管理——页式存储管理

iii 页号表

页号表记录页面进入主存的先后次序：

2→4→5→1



先进先出淘汰算法图例子

当要调入第6页时：

- 置换第2页
- 将第2页改为6
- 替换指针指向第4页

iv **在存储分块表中建立次序表**

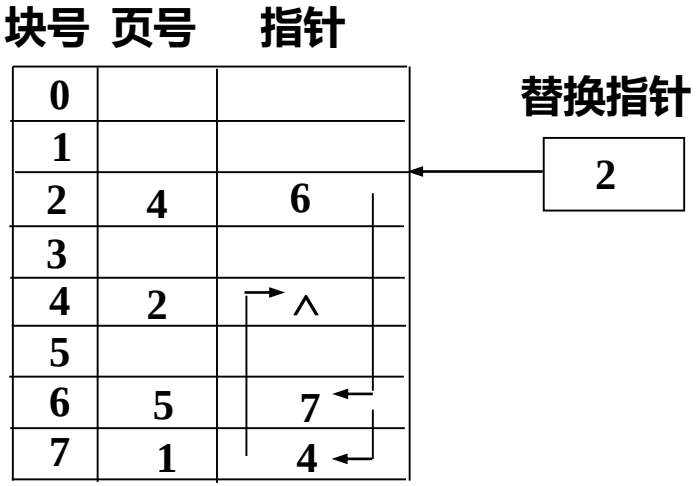
在存储分块表中记录页面进入主存的先后次序：

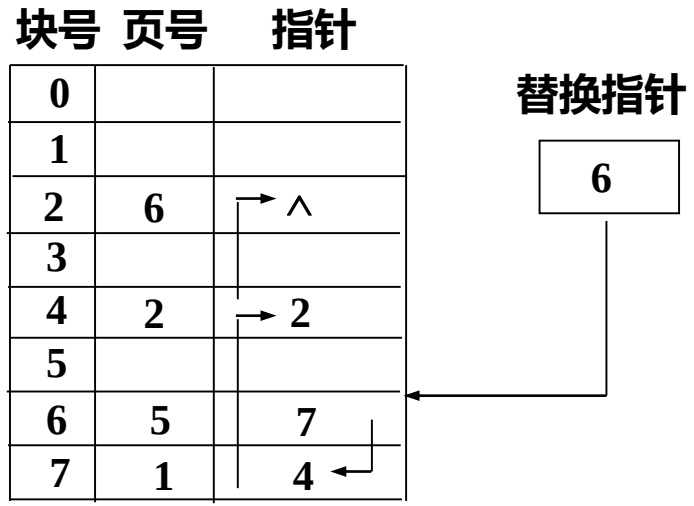
4→5→1→2

当要调入第6页时：

如何处理？

5→1→2 → 6





先进先出淘汰算法存储块构造

③ 最久未使用淘汰算法(LRU算法)

i 什么是最久未使用淘汰算法

总是选择最长时间未被使用的那一页淘汰。

ii 最久未使用淘汰算法的实现

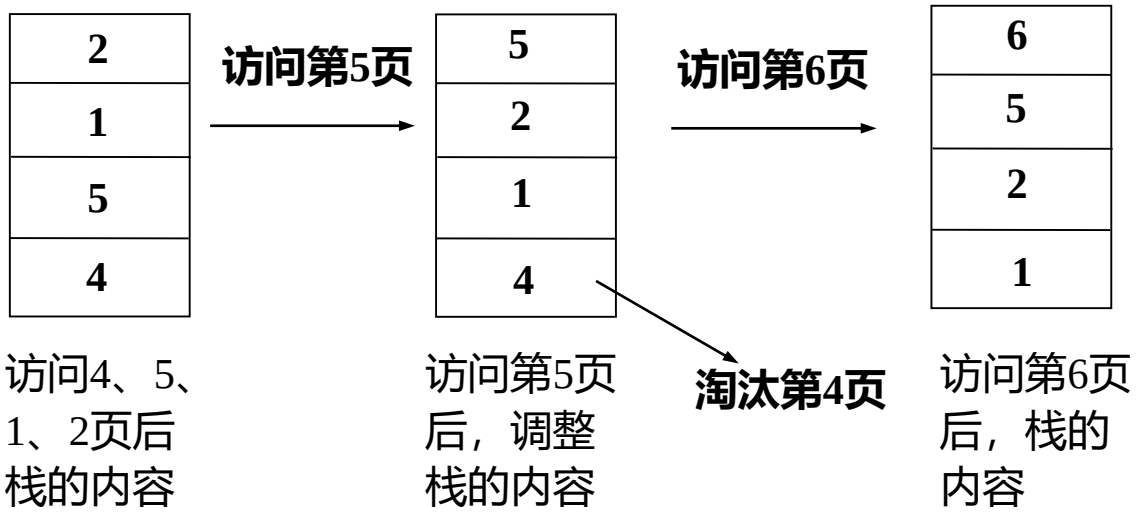
- 用引用位考察页面的使用情况；
- 当访问页面时，将引用位置1，并记时；
- 当要淘汰一页时，选择时间最长的一页淘汰。

要精确实现很困难

- **硬件方法：采用计数器**
- **软件方法：采用页号栈**

iii 软件方法：采用页号栈

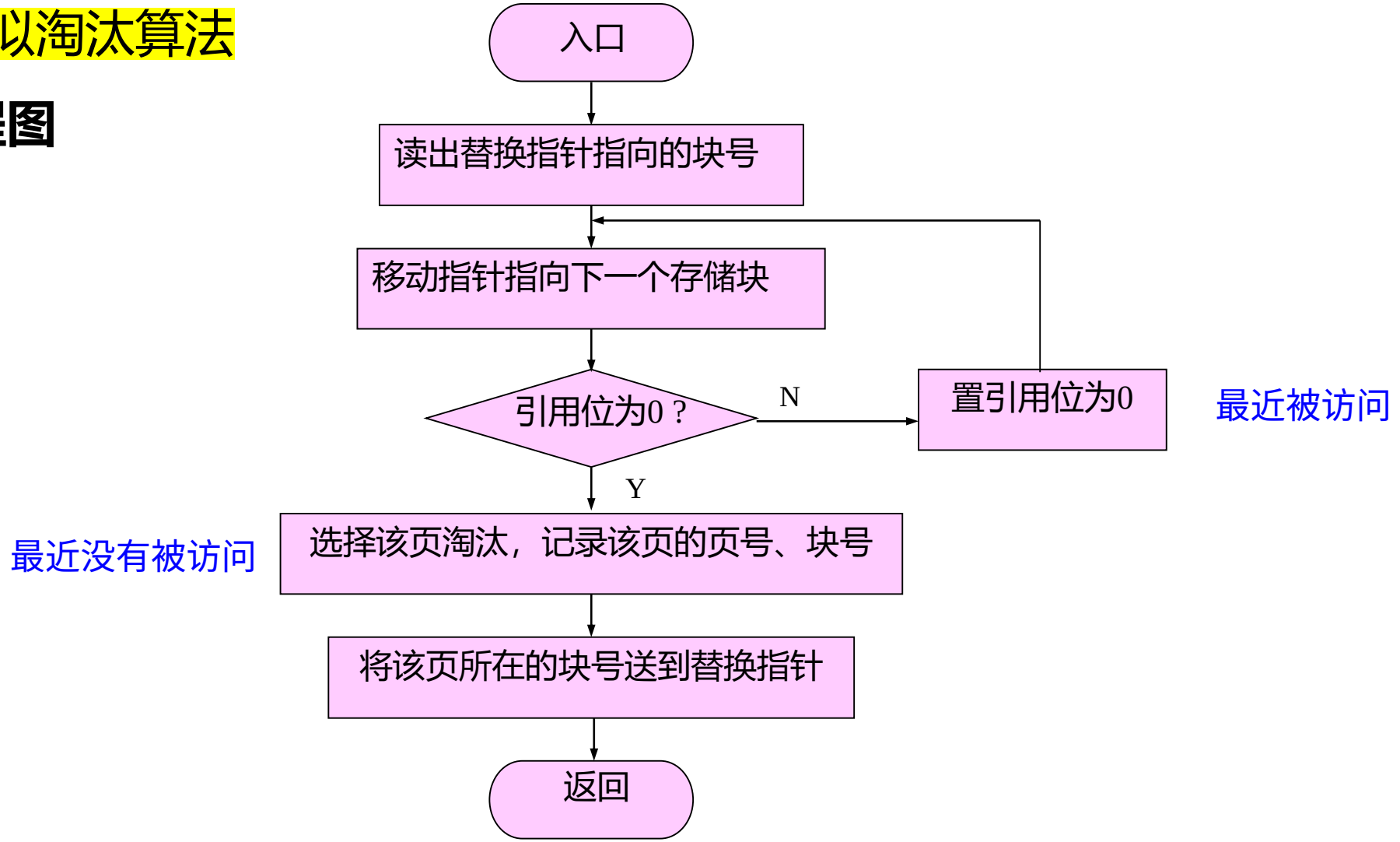
页面访问轨迹：**4→5→1 →2 →5 →6**



用页号栈记录最近访问的页

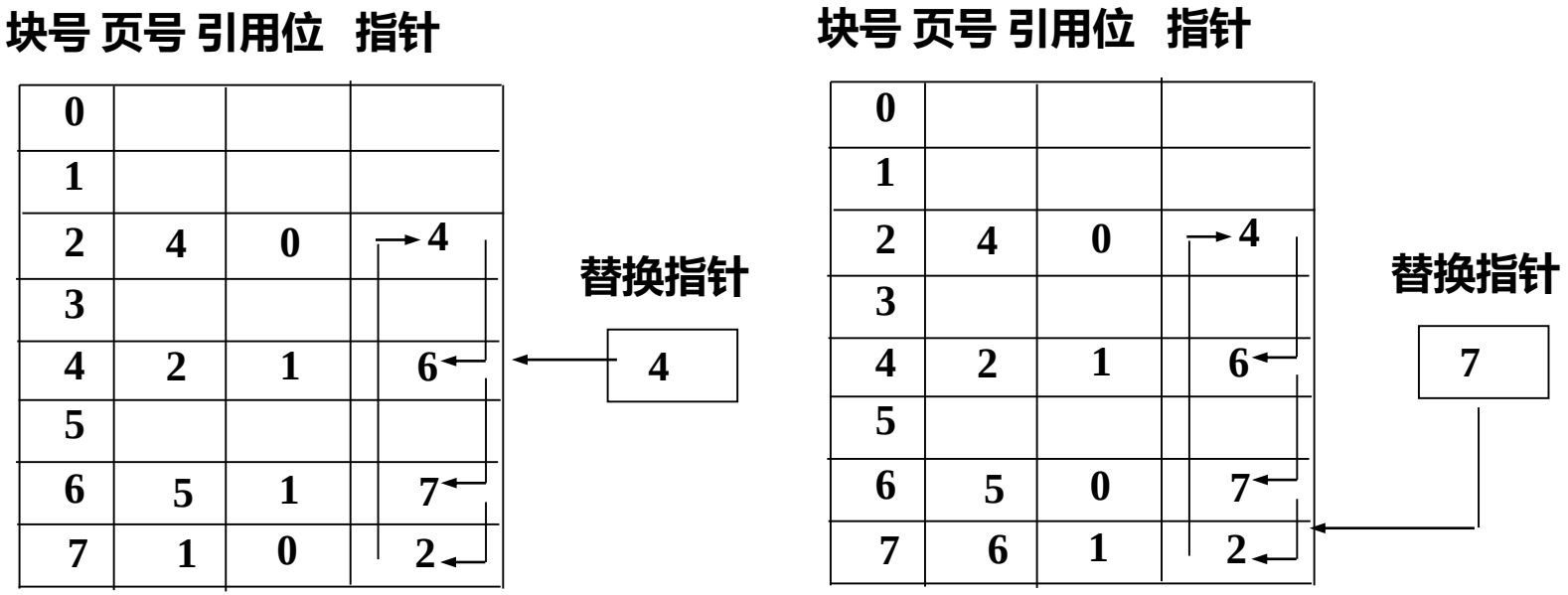
④ LRU近似淘汰算法

i 流程图



LRU近似算法流程图

ii LRU近似淘汰算法举例



LRU近似算法举例

当要调入第6页时，如何处理？

ii LRU近似淘汰算法举例

某请求分页系统的局部页面置换策略为：系统从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集(扫描时间忽略不计)，本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次被分配之前不被清空。当发生缺页时，如果该页曾被使用过且还在空闲页框链表中，则重新放回进程的驻留集中；否则从空闲页框链表头部取出一个页框。

假设不考虑其它进程的影响和系统开销，初始时进程驻留集为空。目前系统空闲页框链表中页框号依次为32,15,21,41。进程P依次访问的<虚拟页号，访问时刻>是：<1,1>，<3,2>，<0,4>，<0,6>，<1,11>, <0,13>,<2,14>。请回答下列问题：

（1）访问<0，4>时，对应的页框号是什么？

（2）访问<1，11>时，对应的页框号是什么？说明理由。

（3）访问<2，14>时，对应的页框号是什么？说明理由。

（4）该策略是否适合于时间局部性好的程序？说明理由。

段页式存储管理

主存管理——段页式存储管理

1. 段式地址空间

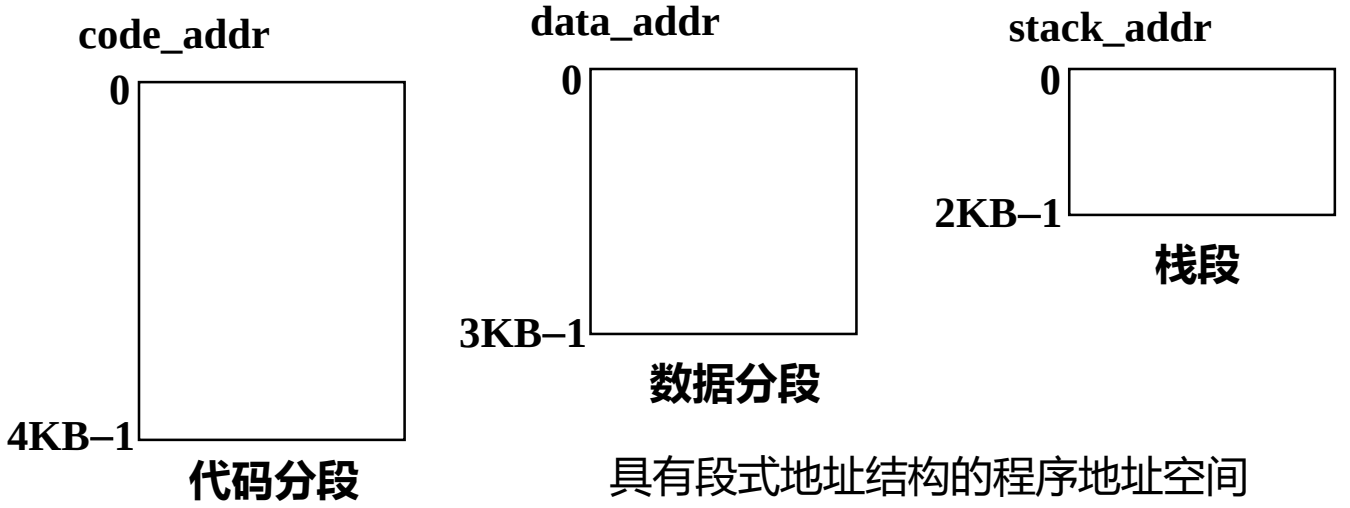
(1) 什么是段

分段是程序中自然划分的一组逻辑意义完整的信息集合。

分段的例：代码分段、数据分段、栈段页。

(2) 程序地址空间

由若干个逻辑分段组成，每个分段有自己的名字，对于一个分段而言，它是一个连续的地址区。



具有段式地址结构的程序地址空间

(3) 段式地址结构

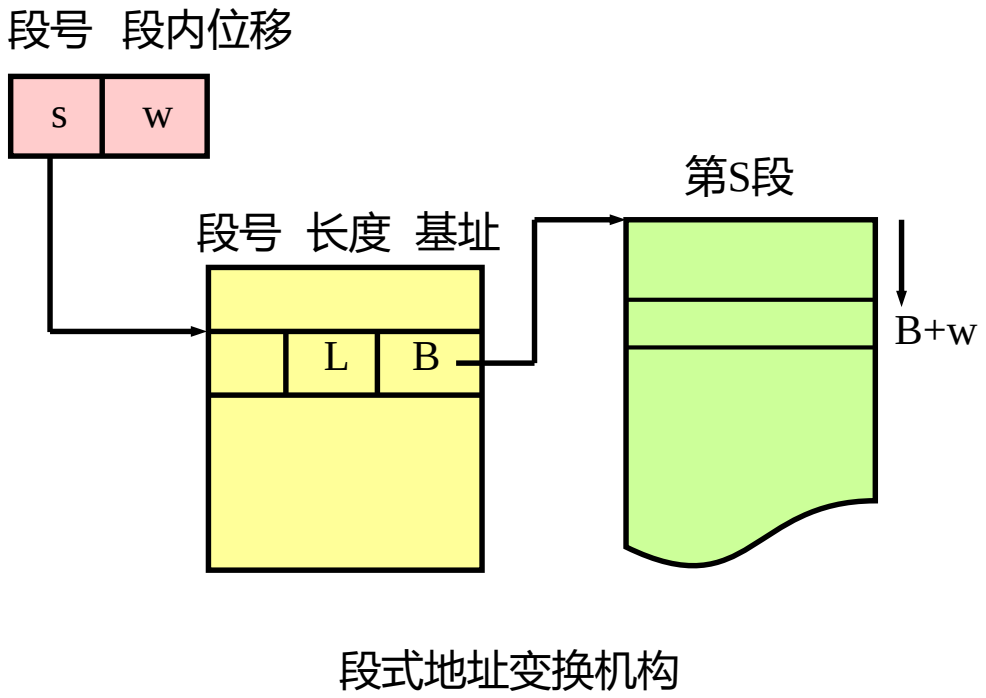
段号 s	段内位移 w

段式地址结构

2. 段式地址变换

段式地址步骤

- 取出程序地址(s，w)；
- 用s检索段表；
- 如w < 0或w≥L则主存越界；
- (B + w)即为所需主存地址



段式地址变换机构

主存管理——段页式存储管理

3. 页式系统与段式系统的区别

(1) 用户地址空间的区别

① 页式系统中用户地址空间： **一维地址空间**

② 段式系统中用户地址空间： **二维地址空间**

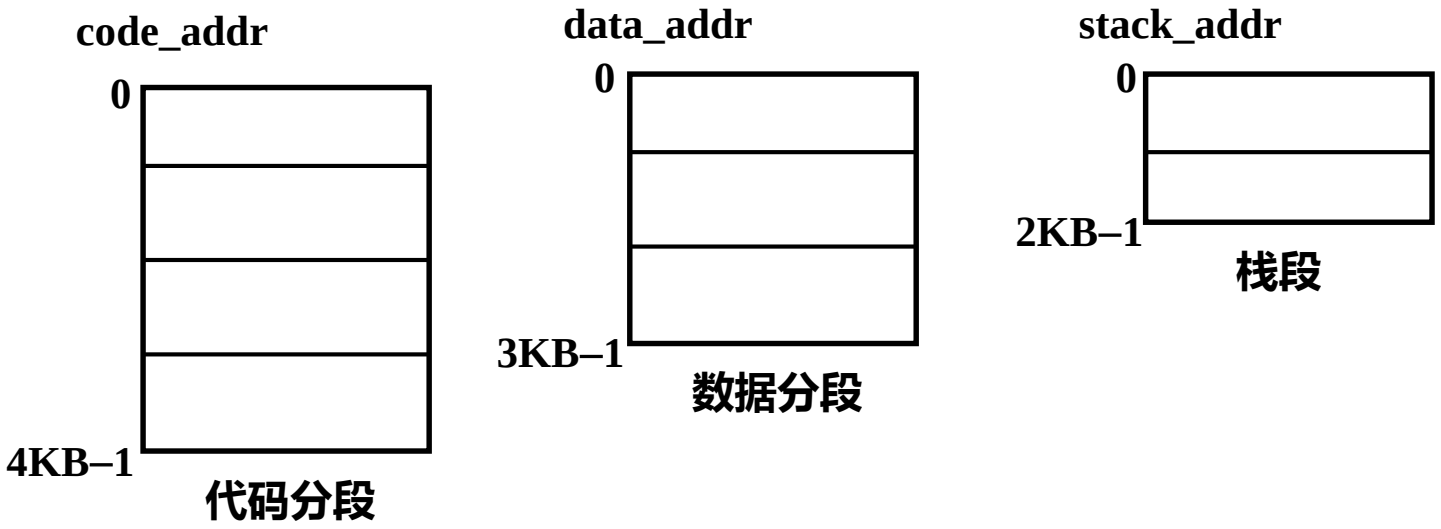
(2) 分段和分页的区别

分段	分页
信息的逻辑划分	信息的物理划分
段长是可变的	页的大小是固定的
用户可见	用户不可见
w字段的溢出将产生越界中断	w字段的溢出自动加入到页号中

4. 段页式系统

在段式存储管理中结合分页存储管理技术，在一个分段内划分页面，就形成了段页式存储管理。

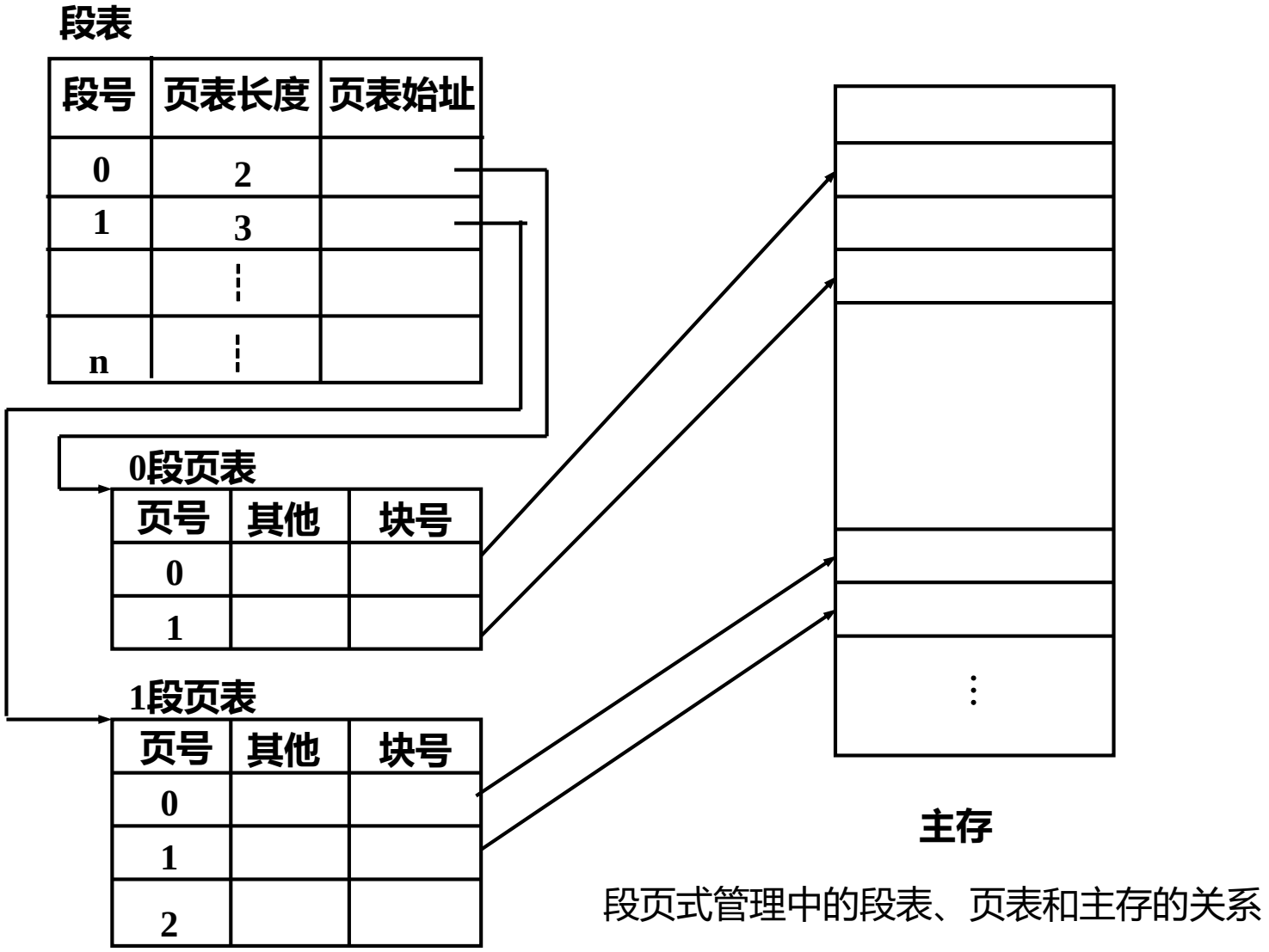
(1) 段页式地址结构的程序地址空间



段页式地址结构

主存管理——段页式存储管理

(2) 段页式系统中段表、页表与主存的关系



段页式管理中的段表、页表和主存的关系

第7章 主存管理

小结

主存管理——小结

基本概念

- 虚、实分离
  - 逻辑地址 作业地址空间
  - 物理地址 存储空间
- 地址映射
  - 定义
  - 类型——静态地址重定位 定义 实现
    动态地址重定位 定义 实现
- 虚存
- 存储保护
  - 定义
  - 界地址保护的实现方法

主存管理——小结

分区存储管理

- 分区分配中的数据结构
  - 空闲区队列结构
- 放置策略
  - 首次适应算法、最佳适应算法、最坏适应算法的定义、特点
  - 三种放置策略的讨论
- 分区的缺点及解决
  - 碎片与拼接

页式存储管理

- 页式地址变换
  - 页表　虚地址结构
  - 页式地址变换过程
- 请调策略
  - 扩充页表功能 —— 中断位　辅存地址
  - 缺页处理
- 淘汰策略
  - 扩充页表功能 —— 引用位 改变位
  - 抖动
  - 置换算法
    - 定义
    - 先进先出淘汰算法、最久未使用淘汰算法

主存管理——小结

- **段页式存储管理**
  - 段式系统的二维地址结构
  - 段页式系统中段表、页表与主存的关系