

第五章 虚拟存储器

5.1 虚拟存储器概述

5.2 请求分页存储管理方式

5.3 页面置换算法

5.4 抖动

5.5 请求分段存储管理方式



5.1 虚拟存储器的基本概念

5.1.1 虚拟存储器的引入

1. 常规存储器管理方式的特征

- ((1)) 一次性（指全部装入）。
- (2) 驻留性（指驻留在内存不换出）。



2. 局部性原理

- 时间局部性：如循环执行
- 空间局部性：如顺序执行。

3. 虚拟存储器

- 具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储系统。
- 实质：以时间换空间，但时间牺牲不大。



5.1.2 虚拟存储器的实现方式

❖ 需要动态重定位

一、请求分页系统

- * 以页为单位转换

- * 需硬件：

 - (1) 请求分页的页表机制

 - (2) 缺页中断

 - (3) 地址变换机构

- * 需实现请求分页机制的软件（置换软件等）



二、请求分段系统

*以段为单位转换:

(1) 请求分段的段表结构

(2) 缺段中断

(3) 地址变换机构

*需实现请求分段机制的软件（置换软件等）



5.1.3 虚存特征

1. 离散性：部分装入

（若连续则不可能提供虚存），无法支持大作业小内存运行

2. 多次性：局部装入，多次装入。

3. 对换性。

4. 虚拟性。



5.2 请求分页存储管理方式

5.2.1 请求分页中的硬件支持

1. 页表机制

页号	物理块号	状态位P	访问字段A	修改位M	外存地址
----	------	------	-------	------	------



2. 缺页中断机构

缺页中断机构：可在指令执行期间产生，转入缺页中断处理程序。

页面

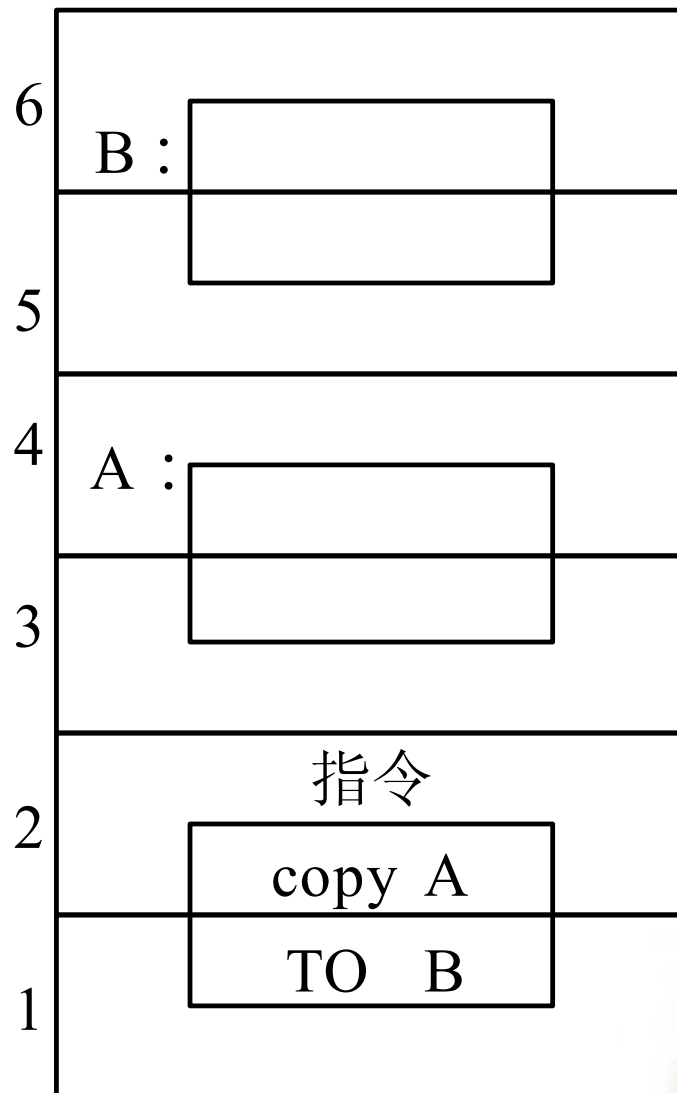
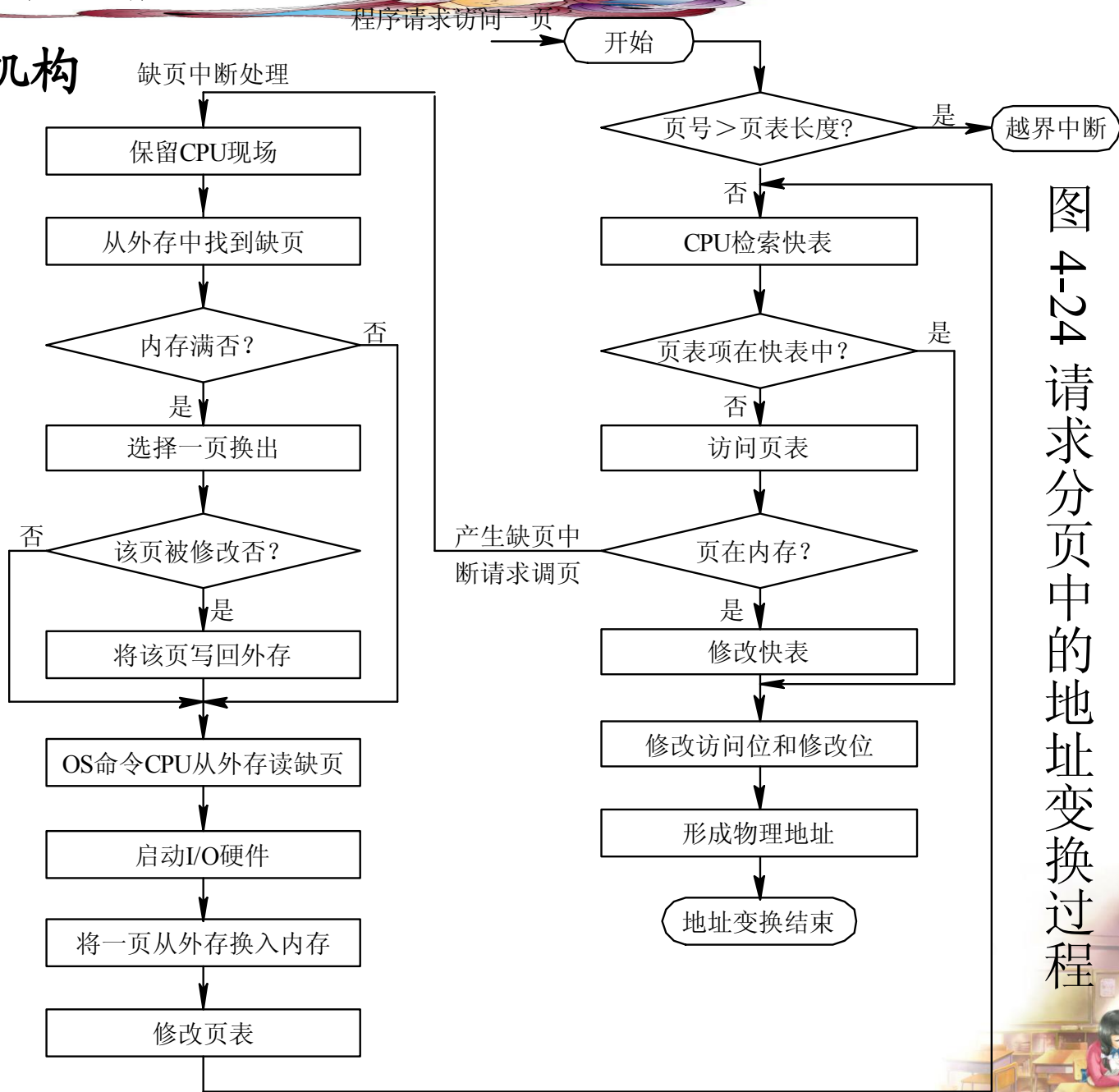


图 4-23 涉及6次缺页中断的指令

3. 地址变换机构



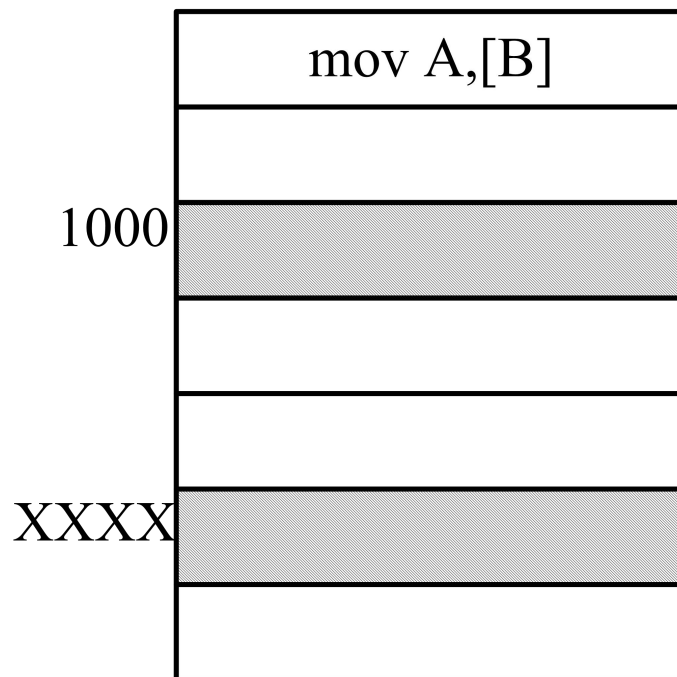
5.2.2 内存分配策略和分配算法

一、最小物理块数

不同的作业要求不同。

如：允许间接寻址：则至少
要求3个物理块。

Mov A, [B]



二、页面分配和置换策略。

1. 固定分配局部置换。

- 缺点：难以确定固定分配的页数。

(少：置换率高； 多：浪费)

2. 可变分配全局置换

3. 可变分配局部置换

- 根据进程的缺页率进行物理块数调整，进程之间相互不会影响。



三、分配算法

1. 平均分配算法
2. 按进程大小比例分配算法:

$$S = \sum_{i=1}^n S_i \quad b_i = \frac{S_i}{S} \times m$$

3. 考虑优先权分配算法



5.2.3 页面调入策略

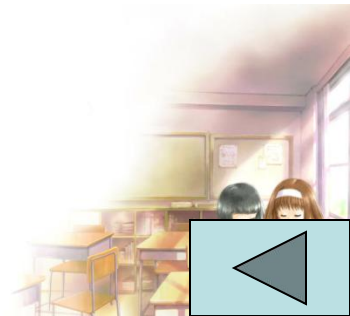
1. 调入时机:

- * 预调: (根据空间局部性)
- * 目前: 成功率 $\leq 50\%$
- * 请求调入: 较费系统开销
- * 各有优劣

2. 从何处调页:

- * 对换区: 全部从对换区调入所需页面, 快
- * 文件区: 修改过的页面换出到对换区, 稍慢
- * UNIX方式: 未运行过的页面, 都应从文件区调入。曾经运行过但又被换出的页面, 从对换区调入。对共享页, 应判断其是否在内存区。

3. 页面调入过程



5.3 页面置换算法

5.3.1 最佳置换算法和先进先出置换算法

1. 最佳 (Optimal) 置换算法

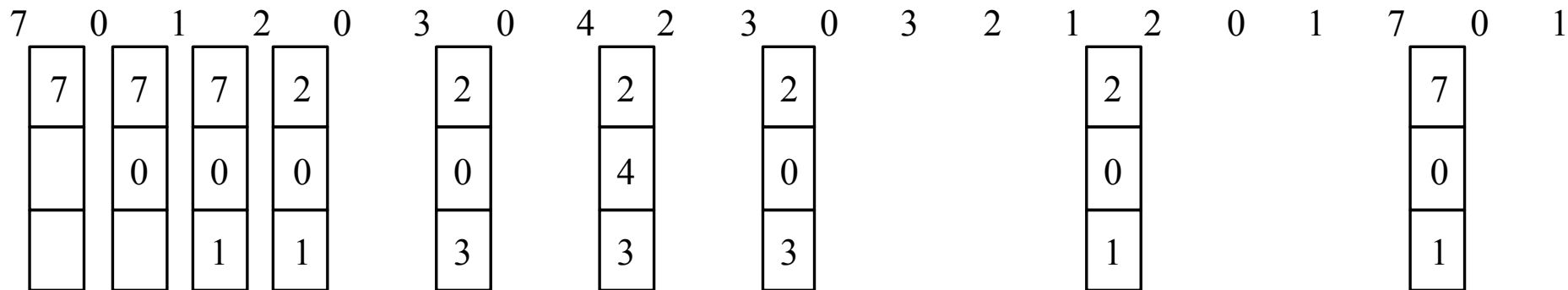
最佳置换算法是由Belady于1966年提出的一种理论上的算法。其所选择的被淘汰页面，将是以后永不使用的，或许是在最长(未来)时间内不再被访问的页面。



假定系统为某进程分配了三个物理块， 并考虑有以下的页面号引用串：

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

引用率



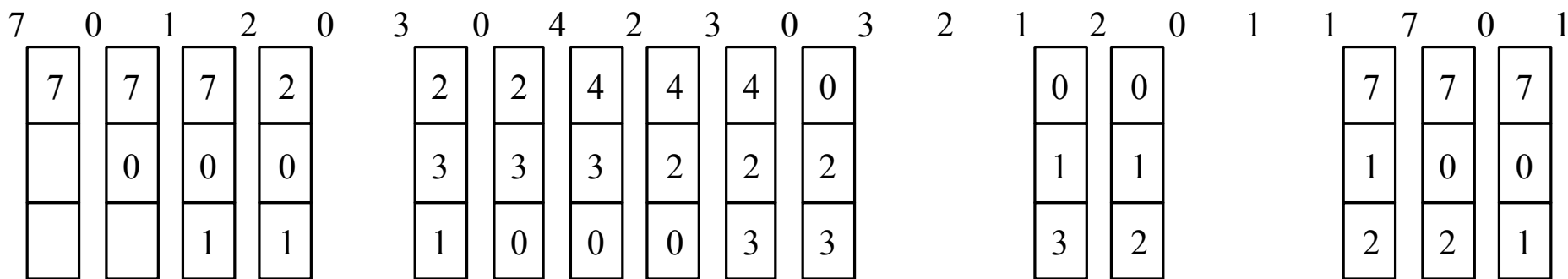
页框(物理块)

图 4-25 利用最佳页面置换算法时的置换图



2. 先进先出 (FIFO) 页面置换算法

引用率



页框

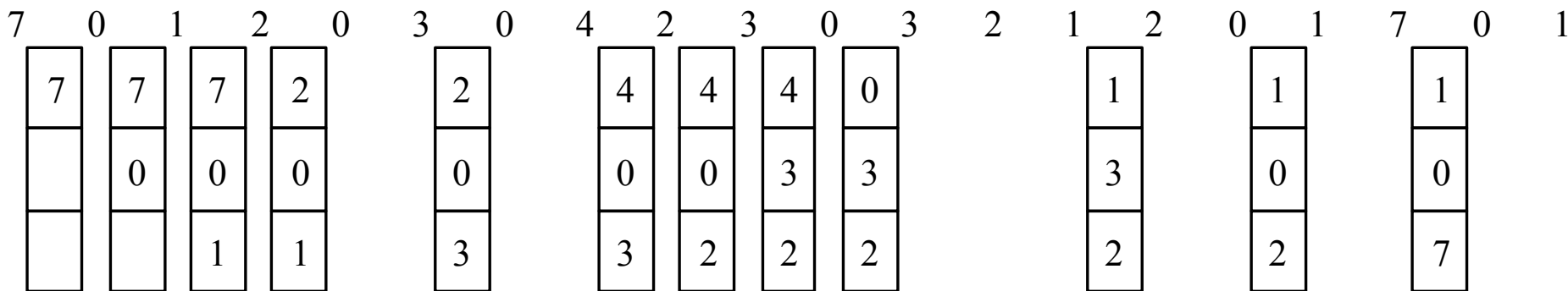
图 4-26 利用FIFO置换算法时的置换图



5.3.2 最近最久未使用 (LRU) 置换算法

1. LRU (Least Recently Used) 置换算法的描述

引用率



页框

图 4-27 LRU 页面置换算法



2. LRU置换算法的硬件支持

1) 寄存器

为了记录某进程在内存中各页的使用情况，须为每个在内存中的页面配置一个移位寄存器，可表示为

$$R=R_{n-1}R_{n-2}R_{n-3} \dots R_2R_1R_0$$



实页 \ R	R ₇	R ₆	R ₅	R ₄	R ₃	R ₂	R ₁	R ₀
1	0	1	0	1	0	0	1	0
2	1	0	1	0	1	1	0	0
3	0	0	0	0	0	1	0	0
4	0	1	1	0	1	0	1	1
5	1	1	0	1	0	1	1	0
6	0	0	1	0	1	0	1	1
7	0	0	0	0	0	1	1	1
8	0	1	1	0	1	1	0	1

图 4-28 某进程具有8个页面时的LRU访问情况



2) 栈

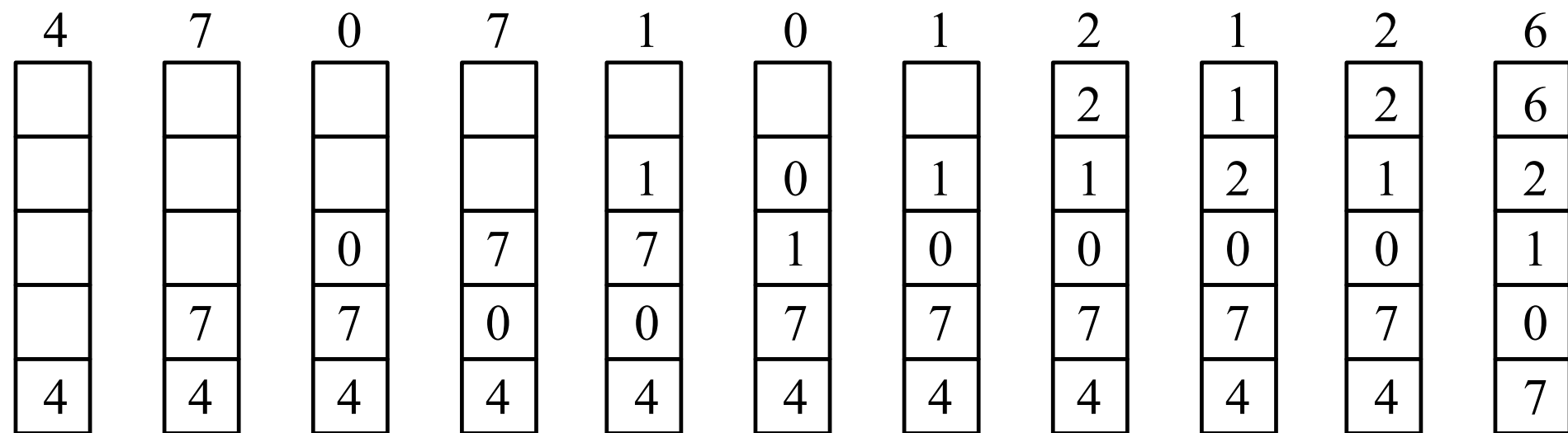
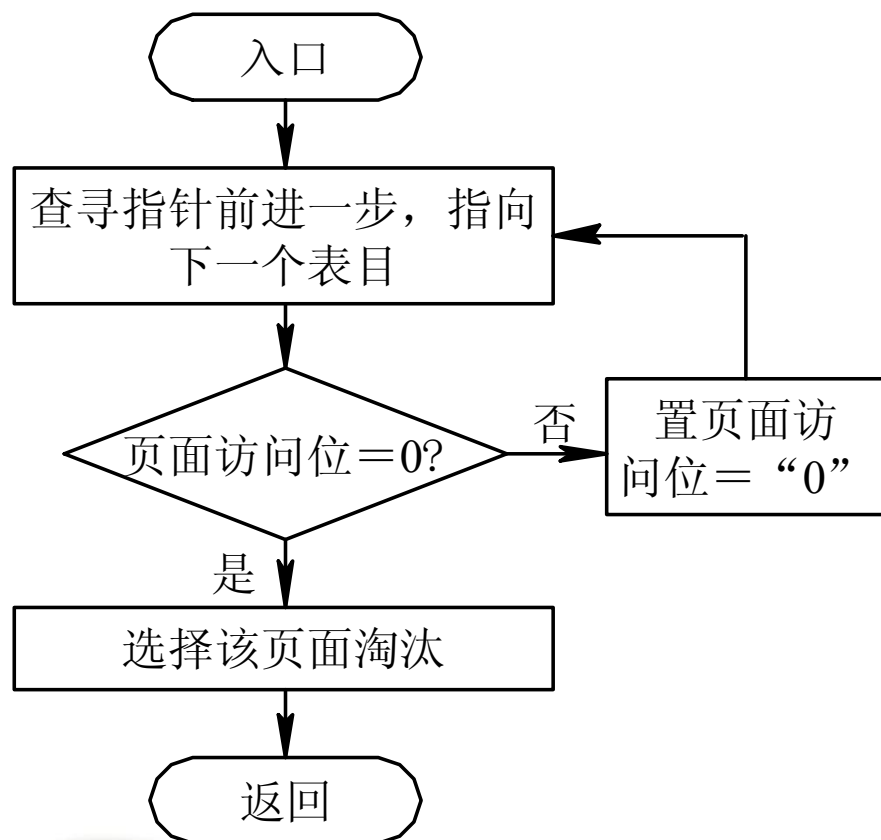


图 4-29 用栈保存当前使用页面时栈的变化情况



5.3.3 Clock置换算法

1. 简单的Clock置换算法



块号	页号	访问位	指针
0			
1			
2	4	0	
3			
4	2	1	
5			
6	5	0	
7	1	1	

替换
指针

图 4-30 简单Clock置换算法的流程和示例

2. 改进型Clock置换算法

由访问位A和修改位M可以组合成下面四种类型的页面：

1类 ($A=0, M=0$): 表示该页最近既未被访问, 又未被修改, 是最佳淘汰页。

2类 ($A=0, M=1$): 表示该页最近未被访问, 但已被修改, 并不是很好的淘汰页。

3类 ($A=1, M=0$): 最近已被访问, 但未被修改, 该页有可能再被访问。

4类 ($A=1, M=1$): 最近已被访问且被修改, 该页可能再被访问。



其执行过程可分成以下三步：

(1) 从指针所指示的当前位置开始，扫描循环队列，寻找 $A=0$ 且 $M=0$ 的第一类页面，将所遇到的第一个页面作为所选中的淘汰页。在第一次扫描期间不改变访问位 A 。

(2) 如果第一步失败，即查找一周后未遇到第一类页面，则开始第二轮扫描，寻找 $A=0$ 且 $M=1$ 的第二类页面，将所遇到的第一个这类页面作为淘汰页。在第二轮扫描期间，将所有扫描过的页面的访问位都置0。

(3) 如果第二步也失败，亦即未找到第二类页面，则将指针返回到开始的位置，并将所有的访问位复0。然后重复第一步，如果仍失败，必要时再重复第二步，此时就一定能找到被淘汰的页。



5.3.4 请求分页系统的性能分析

请求分页系统是目前最常用的一种存储方式，但运行中产生的缺页情况会影响速度和系统性能，而缺页率的高低往往与进程所占用的物理块数有关。因此本节分析缺页率对系统性能的影响，以及应为每个进程所分配的物理块数目。

1. 缺页率与有效访问时间

$$\text{有效访问时间} = (1-p) * t + p * f$$

其中：p为缺页率，t为内存访问时间，f为缺页中断时间



说明:

- 现代计算机系统，内存访问时间在10ns到数百ns之间。
(以100ns为例计算)
- 缺页中断时间包括三部分：(1) 缺页中断服务时间；(2) 将缺页读入的时间；(3) 进程重新执行时间。由于CPU时间很快，所以(1)(3)可以不超过1ms；(2)则包括寻道时间、旋转时间和数据传送时间，大体需要24ms。代入上式得：

$$\begin{aligned}\text{有效访问时间} &= (1-p) * 0.1 (\mu s) + p * 25000 (\mu s) \\ &= 0.1 + 24999.9 * p\end{aligned}$$

如果缺页率 $p=0.001$ (即在1000次的页面访问中，仅发生一次缺页)

则有效访问时间约为 $25 \mu s$ ，与无缺页相比，速度降低至 $1/250$ 。



❖ 如果希望在缺页时有效访问时间延长不超过10%，则有

$$0.11 > 0.1 + 24999.9 * p$$

$$\text{则 } p < 0.01 / 24999.9 = 0.0000004$$

结论：有效访问时间直接比例于缺页率，改善请求分页系统的性能，需要保持非常低的缺页率，同时提高I/O的速度。



2. 抖动——是这样一种系统状态，即系统花在页面替换上的时间远远大于执行进程的时间。

抖动产生的原因：由于分配给进程的页面数大小少于进程所需要的最低页面数，导致出现接连不断的缺页中断，引起抖动。

CPU利用率与多道程序度的关系：多道程序度指在内存中并发执行的程序数目。两者关系如下：在低度情况下，CPU利用率呈线性变化关系。随着度的上升，CPU利用率也逐渐上升，最终上升到一个最大值，若在这种情况下，进一步增加度，则系统发生抖动，且CPU利用率将迅速恶化。

结论：系统可以利用CPU利用率与多道程序度进行比较的方法检测抖动，一旦发生抖动，可以通过减少多道程序度的方法来消除。



例：考虑一个请求分页系统，它采用全局置换策略和平均分配内存块的算法（即若有 m 个内存块和 n 个进程，则每个进程分得 m/n 个内存块）。如果该系统测得如下CPU和对换盘利用率，请问能否用增加多道程序度数来增加CPU的利用率？为什么？

(1) CPU利用率为13%，盘利用率为97%；

(2) CPU利用率为87%，盘利用率为3%；

(1) CPU利用率为13%，盘利用率为3%；



答：

(1) 此时发生抖动现象。增加多道程序度会进一步增加缺页率，使系统性能进一步恶化，所以不能用增加多道程序度数来增加CPU的利用率。

(2) CPU利用率已经相当高，盘利用率却相当低，即进程的缺页率很低，此时应适当增加多道程序度数来增加CPU的利用率。

(3) CPU利用率相当低，盘利用率也相当低，表示内存中可运行的程序数不足，此时应增加多道程序度数来增加CPU的利用率。



5.4 请求分段存储管理方式

5.4.1 请求分段中的硬件支持

1. 段表机制

段名	段长	段的基址	存取方式	访问字段A	修改位M	存在位P	增补位	外存始址
----	----	------	------	-------	------	------	-----	------



在段表项中，除了段名(号)、段长、段在内存中的起始地址外，还增加了以下诸项：

- (1) 存取方式。
- (2) 访问字段A。
- (3) 修改位M。
- (4) 存在位P。
- (5) 增补位。
- (6) 外存始址。



2. 缺段中断机构

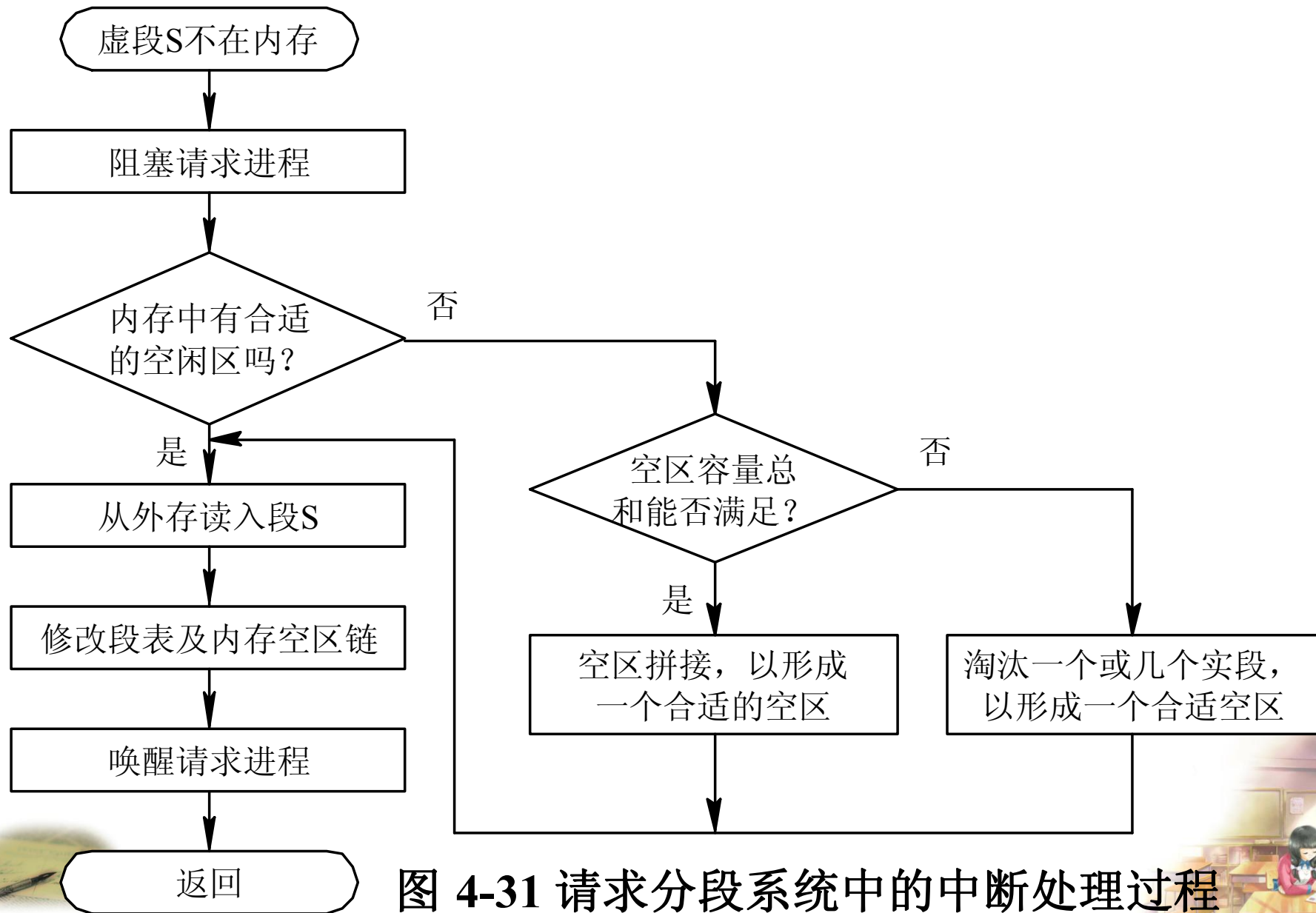


图 4-31 请求分段系统中的中断处理过程

3. 地址变换机构

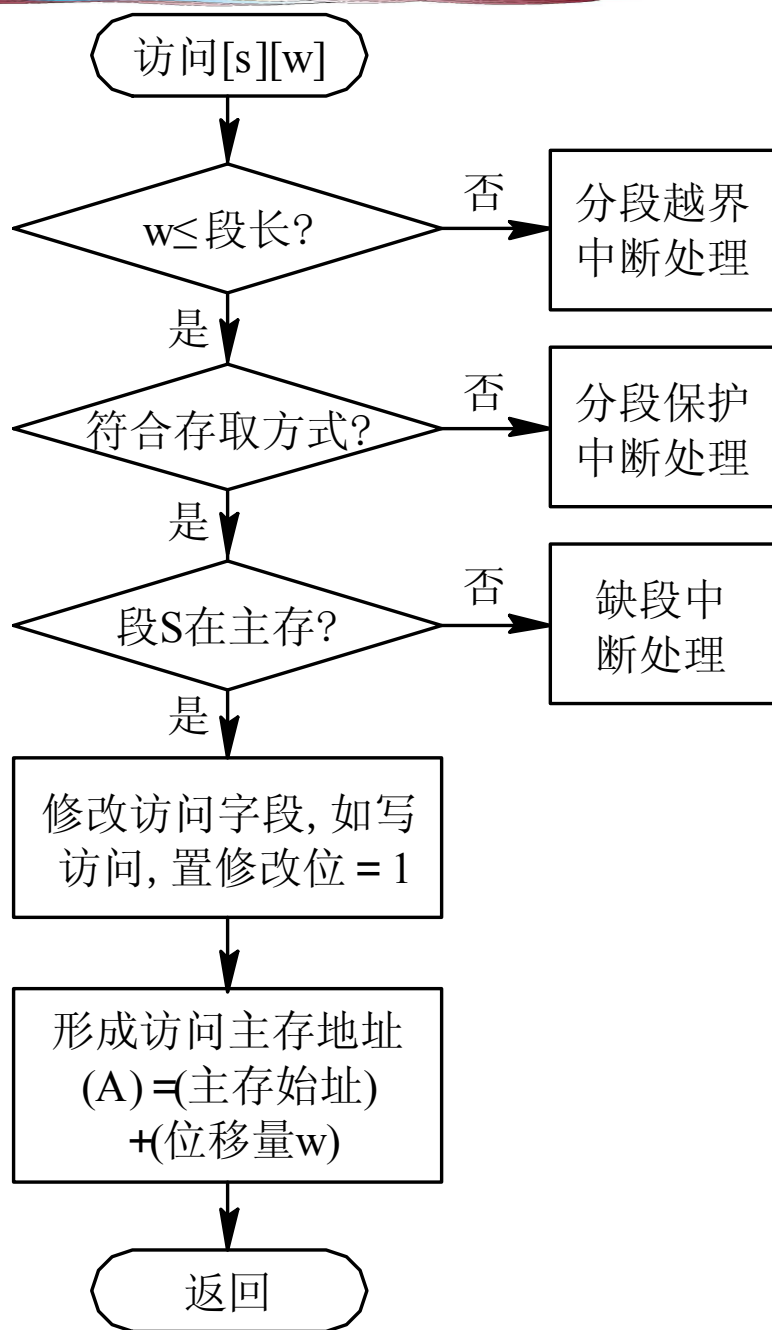
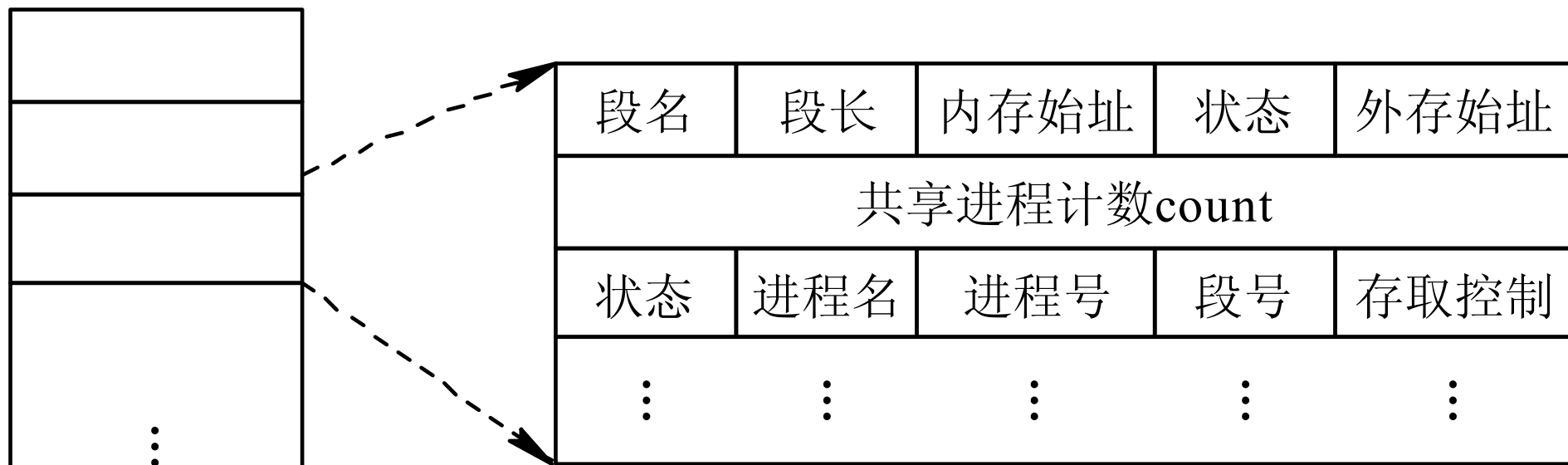


图 4-32 请求分段系统的地址变换过程



5.4.2 分段的共享与保护

1. 共享段表



共享段表

图 4-33 共享段表项



2. 共享段的分配与回收

1) 共享段的分配

在为共享段分配内存时，对第一个请求使用该共享段的进程，由系统为该共享段分配一物理区，再把共享段调入该区，同时将该区的始址填入请求进程的段表的相应项中，**还须在共享段表中增加一表项，填写有关数据，把count置为1**；之后，当又有其它进程需要调用该共享段时，由于该共享段已被调入内存，故此时无须再为该段分配内存，而只需在调用进程的段表中，增加一表项，填写该共享段的物理地址；在共享段的段表中，填上调用进程的进程名、存取控制等，再执行 $\text{count} := \text{count} + 1$ 操作，以表明有两个进程共享该段。



2) 共享段的回收

当共享此段的某进程不再需要该段时，应将该段释放，包括撤在该进程段表中共享段所对应的表项，以及执行 $\text{count} := \text{count} - 1$ 操作。若结果为0，则须由系统回收该共享段的物理内存，以及取消在共享段表中该段所对应的表项，表明此时已没有进程使用该段；否则（减1结果不为0），则只是取消调用者进程在共享段表中的有关记录。



3. 分段保护

1) 越界检查

2) 存取控制检查

※只读 ※只执行 ※读/写

3) 环保护机构

※一个程序可以访问驻留在相同环或较低特权环中的数据。

※一个程序可以调用驻留在相同环或较高特权环中的服务。



典型问题分析:

1. 设作业 A 的页面映象表如下图所示: (一页=1KB)

页号	块号	中断位	访问位	修改位	辅存地址
0	8	1	1	1	1 0 0 0
1	5	1	0	0	3 0 0 0
2	7	1	1	0	5 0 0 0
3		0	0	0	8 0 0 0

问: ①指出页表中中断位、访问位、修改位、辅存地址的含义?

② 当执行到 1 0 0 0 单元的指令 “LOAD 1, 1 8 0 0” 时, 系统是怎样进行地址变换 (即 1 8 0 0 在主存的哪个单元中)?

③ 当执行到 1 5 0 0 单元指令 (LOAD 1, 3 6 0 0) 时, 会发生什么现象?



(1) 中断位：也称状态位，表示该页是否已调入内存；

访问位：记录本页在一段时间内被访问次数；

修改位：表示该页调入内存后是否修改过；

辅存地址：指出该页在辅存上的地址。

(2) 设页号为P，页内地址为d，逻辑地址为A，页面大小为L，则：

$$P = \text{INT}[A/L] \quad d = [A] \bmod L$$

当执行到 1 0 0 0 单元的指令 “LOAD 1, 1 8 0 0” 时，系统地址变换如下：

$$L = 1024\text{B}, A = 1800,$$

$$\text{则 } P = \text{INT}[1800/1024] = 1, d = [1800] \bmod 1024 = 776$$

$$\text{故 } A = 1800 \rightarrow (1, 776)$$

查页表第1页在第5块，所以物理地址为：5896



(3) 当执行到 1 5 0 0 单元指令 (L O A D 1, 3 6 0 0) 时, 系统地址变换如下:

$L=1024B$, $A=3600$,

则 $P=INT[3600/1024]=3$, $d=[3600] \bmod 1024=528$

故 $A=3600 \rightarrow (3, 528)$

查页表第3页未调入内存, 所以产生缺页中断, 从辅存8000位置将该页调入。



2. 有一个二维数组:

```
Var  A: ARRAY [1..100, 1..100] OF integer;
```

按先行后列的次序存储。对一采用LRU置换算法的页式虚拟存储器系统，假设每页可存放200个整数。若分配给一个进程的内存块数为3，其中一块用来装入程序和变量i、j，另外两块专门用来存放数组（不作它用），且程序段已在内存，但数据页尚未装入内存，但数据页尚未装入内存。请分别就下列程序计算执行过程中的缺页次数。

程序1

```
FOR i:=1 TO 100 DO  
  FOR j:=1 TO 100 DO  
    A[i, j]:=0;
```

程序2

```
FOR j:=1 TO 100 DO  
  FOR i:=1 TO 100 DO  
    A[i, j]:=0;
```



2. 答：对程序1，首次缺页中断（访问A[0, 0]时产生）将装入数组的第1、2行共200个整数，由于程序是按行对数组进行访问，只有在处理完200个整数后才会再次产生缺页中断；以后每调入一页，也能处理200个整数，因此处理 100×100 个整数共将发生50次缺页。

对程序2，首次缺页中断同样将装入数组的第1、2行共200个整数，但由于程序是按列队数组进行访问，因此在处理完2个整数后又会再次产生缺页中断；以后每调入一页，也只能处理2个整数，因此，处理 100×100 个整数共将产生5000次调页。



3. 现有一请求调页系统，页表保存在寄存器中，若有一个被替换的页未被修改，则处理一个缺页中断需要8ms，若被替换页修改过，则处理一个缺页中断需要20ms。内存存取时间为 $1\mu\text{s}$ ，访问页表时间可忽略不计。假定70%被替换的页被修改过，为保证有效存取时间不超过 $2\mu\text{s}$ ，可接受最大的缺页率是多少？



3. 答：用 p 表示缺页率，则有效时间不超过 $2\ \mu s$ 可表示为：

$$(1-p) * 1\ \mu s + p * (0.7 * 20ms + 0.3 * 8ms + 1\ \mu s) \leq 2\ \mu s$$

$$p \leq 1/16400 = 0.00006$$

即可接受的最大缺页率为0.00006。



本章作业：

1. 存储器的用户空间共有32个页面，每页1K，主存16K。假定某时刻.某虚拟系统为用户的第0、1、2、3页分配的物理块号为5、10、4、7。而该用户作业的长度为6页，试将十六进制的虚拟地址0A5C、103C、1A5C转换成物理地址。



2.假如一个程序的段表如下，其中存在位为1表示段在内存，对于下面指令，在执行时会产生什么样的结果。

段号	存在位	内存始址	段长	存取控制
0	0	500	100	W
1	1	1000	30	R
2	1	3000	200	E
3	1	8000	80	R
4	0	5000	40	R

(1)STORE R1,[0,70]

(2)STORE R1,[1,20]

(3)LOAD R1,[3,20]

(4)LOAD R1,[3,100]

(5)JMP [2,100]

