第五章　虚拟存储器

§5.1 虚拟存储器概述

5.1.1 常规存储管理方式的特征和局部性原理

前面所介绍的各种存储管理方式，有一个共同的特点，即它们都要求将一个作业全部装入内存后才能运行，于是，就可能出现以下情况：

(1) 有的作业很大，其所要求内存空间超过了内存容量，从而导致作业不能全部被装入内存，以至于该作业无法运行。

(2) 有多个作业要求运行，但可用的内存空间不足以容纳所有的作业，只能将少数的作业装入内存让它们先运行，而将其他的作业留在外存等待。

§5.1 虚拟存储器概述

- 1 常规存储器管理方式的特征

(1) 一次性：作业在运行前需一次性地全部装入内存。将导致上述两问题。

(2) 驻留性：作业装入内存后，便一直驻留内存，直至作业运行结束。

- 2 局部性原理

指程序在执行时呈现出局部性规律，即在一较短时间内，程序的执行仅限于某个部分，相应地，它所访问的存储空间也局限于某个区域。

局部性又表现为时间局部性(由于大量的循环操作，某指令或数据被访问后，则不久可能会被再次访问)和空间局部性（如顺序执行，指程序在一段时间内访问的地址，可能集中在一定的范围之内）。

§5.1 虚拟存储器概述

5.1.2 虚拟内存的定义和特征

1 虚拟存储器的定义

- 基于局部性原理，程序在运行之前，没有必要全部装入内存，仅须将当前要运行的页（段）装入内存即可。
- 运行时，如访问的页（段）在内存中，则继续执行，如访问的页未在内存中（缺页或缺段），则利用OS的请求调页（段）功能，将该页（段）调入内存。如内存已满，则利用OS的页（段）置换功能，按某种置换算法将内存中的某页（段）调至外存，从而调入需访问的页。
- **虚拟内存（Virtual Memory）**是指在具有层次结构存储器的计算机系统中，采用自动实现部分装入和部分对换功能，为用户提供一个比物理主存容量大得多的可寻址的一种“主存储器”。它使用户逻辑存储器与物理存储器分离，是操作系统给用户提供的一个比真实内存空间大得多的地址空间 。

§4.6 虚拟存储器

- 实现虚拟存储器的物质基础是二级存储器结构和动态地址转换机构。经过操作系统的改造，把计算机的内存与外存有机的结合起来使用，从而得到一个容量很大的“内存”，这就是虚存。
- 虚拟存储器实质上是把用户地址空间和实际的存储空间区分开来，当作两个不同的概念。它的容量主要受到两方面的限制：

（1）指令中表示地址的字长。一个虚拟存储器的最大容量是由计算机的地址结构确定的。

(2)外存的容量。虚拟存储器的容量与主存的实际大小没有直接的关系，而是由主存与辅存的容量之和所确定。

§5.1 虚拟存储器概述

- **2 虚拟存储器的特征**
- 虚拟性。虚拟内存不是扩大实际的物理内存，而是扩充逻辑内存的容量。
- 部分装入。每个进程不是全部装入内存，而是分成若干个部分。当进程需要执行时，才将当前运行所需要的程序和数据装入内存。
- 对换性。在一个进程运行期间，它所需要的程序和数据可以分多次调入。每次仅仅调入一部分，以满足当前程序执行的需要。而且，在内存中那些暂时不使用的程序和数据可以换到外存的交换区存放，以腾出尽量多的内存空间供可运行进程使用。

§5.1 虚拟存储器概述

2 虚拟存储器的实现方法

- ◆分页请求系统
- ◆请求分段系统

§5.2 请求分页技术存储管理方式

5.2.1 请求分页存储管理的基本思想

- 请求式分页也称虚拟页式存储管理，它的基本思想是：在进程开始运行之前，不是装入全部页面，而是装入一个或零个页面，之后根据进程运行的需要，动态装入其它页面；当内存空间已满，而又需要装入新的页面时，则根据某种算法淘汰某个页面，以便装入新的页面。
- 为了实现页式虚存，系统需要解决下面三个问题：

（1）系统如何获知进程当前所需页面不在主存。

（2）当发现缺页时，如何把所缺页面调入主存。

（3）当主存中没有空闲的页框时，为了要接受一个新页，需要把老的一页淘汰出去，根据什么策略选择欲淘汰的页面。

页面置换算法

扩充页表

缺页中断

§5.2 请求分页技术

1. 请求页表机制

页号	块号	状态位	访问字段	修改位	外存地址

（1）状态位P：指示该页是否已调入内存。

（2）访问字段A：记录本页在一段时间内被访问的次数或最近未被访问的时间。

（3）修改位M：表示该页在调入内存后是否被修改过。若修改过，则换出时需重写至外存。

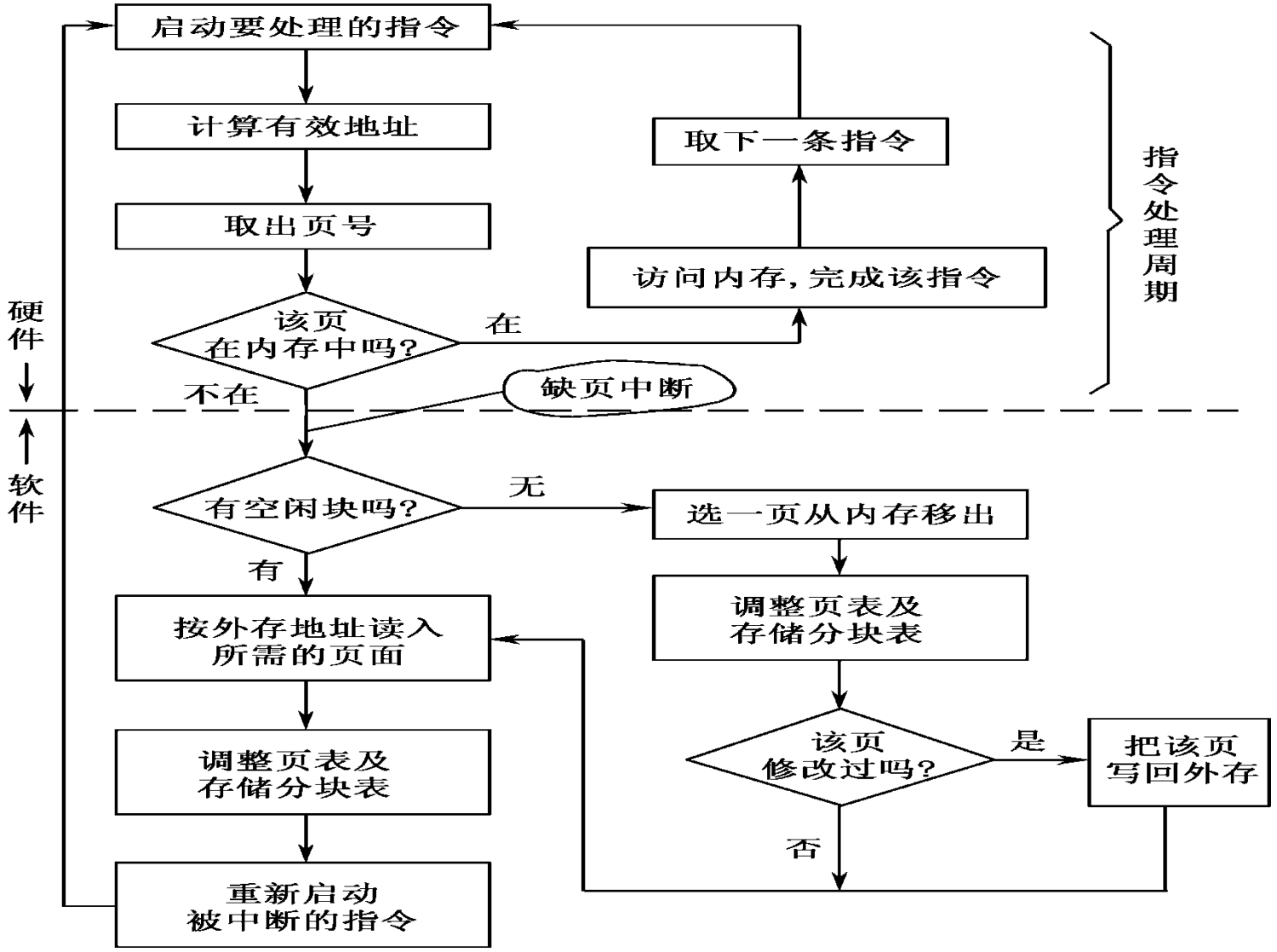
（4）外存地址：指出该页在外存上的地址。

§5.2 请求分页技术

2. 缺页中断机构

- 程序在执行时，首先检查页表，当状态位指示该页不在主存时，则引起一个缺页中断发生，相应的中断处理程序把控制转向缺页中断子程序。执行此子程序，即把所缺页面装入主存。然后处理机重新执行缺页时打断的指令。这时，就将顺利形成物理地址。
- 缺页中断的处理过程是由硬件和软件共同实现的。

§5.2 请求分页技术


启动要处理的指令
计算有效地址
取出页号
该页
在内存中吗?
在
不在
缺页中断
有空闲块吗?
无
选一页从内存移出
调整页表及
存储分块表
该页
修改过吗?
是
把该页
写回外存
否
有
按外存地址读入
所需的页面
调整页表及
存储分块表
重新启动
被中断的指令
取下一条指令
访问内存,完成该指令
指令处理周期
硬件
软件

§5.2 请求分页技术

缺页中断同一般中断都是中断，相同点是：

- 保护现场　中断处理　恢复现场

不同点：

- 一般中断是一条指令完成后中断，缺页中断是一条指令执行时中断
- 一条指令执行时可能产生多个缺页中断。如指令可能访问多个内存地址，这些地址在不同的页中。

§5.2 请求分页技术

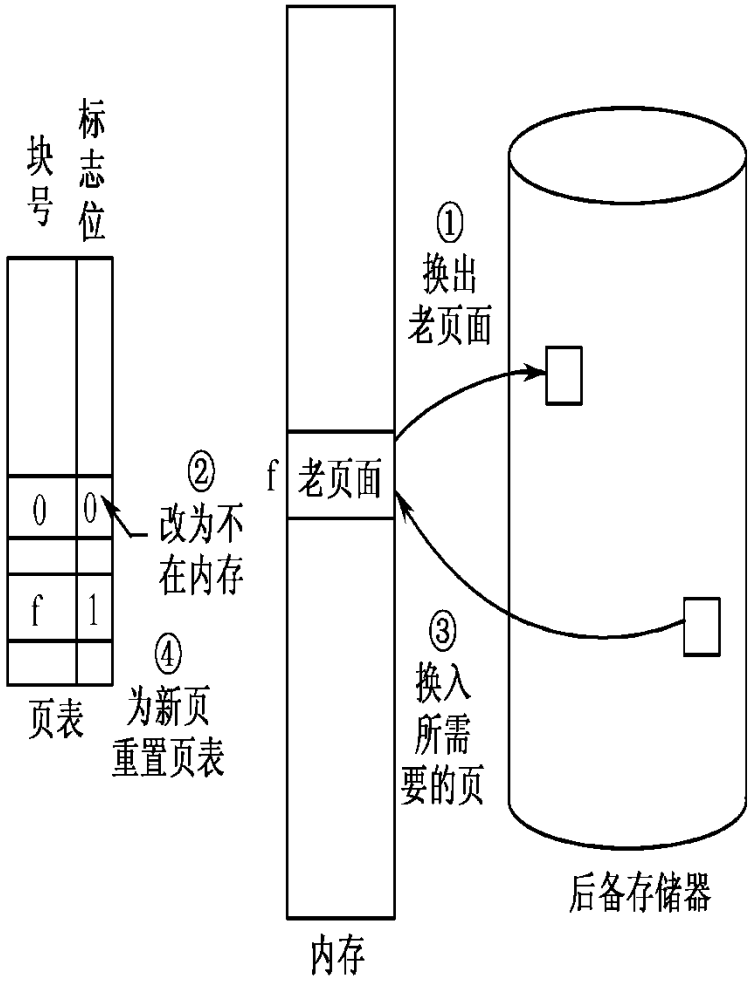
3. 页面置换的过程

（1）找出所需页面在磁盘上的位置；

（2）找出可用空闲内存块。如果有，就立即使用，否则，就进行页面置换，选择一个老的页面置换到外存磁盘。

（3）将所需页面装入内存，修改相应的数据结构。

（4）继续执行用户进程。



§5.2 请求分页技术

4. 地址变换机构

- 调页+基本分页地址映射过程

§5.2 请求分页技术

5.2.2 请求分页中的内存分配

1. 最小物理块数的确定

分给每个进程的最少内存块数是指保证进程正常运行所需的最少内存块数，它是由指令集结构决定的。

2. 内存分配策略

（1）固定分配策略分配给进程的内存块数是固定的，且在最初装入时（即进程创建时）确定块数。

（2） 可变分配策略允许分给进程的内存块数随进程的活动而改变。

（3）页面置换范围

- 全局置换允许一个进程从全体存储块的集合中选取淘汰块，尽管该块当前分配给其他进程，还是能强行占用。
- 局部置换是每个进程只能从分给它的一组内存块中选择淘汰块。

§5.2 请求分页技术

- **固定分配局部置换：**为每个进程分配固定数目n的物理块，在整个运行中都不改变。如出现缺页，则从中置换一页。
- **可变分配全局置换：**分配固定数目的物理块，但OS自留一空闲块队列，若发现缺页，则从空闲块队列中分配一空闲块与该进程，并调入缺面于其中。当空闲块队列用完时，OS才从内存中任选择一页置换。
- **可变分配局部置换：**分配一定数目的物理块，若发现缺页，则从该进程的页面中置换一页，根据该进程缺页率高低，则可增加或减少物理块。

§5.2 请求分页技术

3. 物理块分配算法

在采用固定分配策略时，将系统中可供分配的所有物理块分配给各个进程，可采用以下几种算法：

（1）平均分配算法：平均分配给各个进程。

（2）按比例分配算法：根据进程的大小按比例分配给各个进程。

（3）考虑优先权的分配算法：将系统提供的物理块一部分根据进程大小先按比例分配给各个进程，另一部分再根据各进程的优先权适当增加物理块数。

§5.2 请求分页技术

5.2.3 页面调入策略

1. 何时调入页面

预调页策略

请求调页策略

2. 从何处调入页面

有足够的对换空间

缺少足够的对换空间

unix方式

3. 页面调入过程

4. 缺页率

缺页率=缺页次数/总的页面访问次数×100%

§5.3 页面置换算法

- 为实现请求分页虚拟内存，系统必须解决内存分配算法和页面置换算法两个主要问题。其中页面置换算法的选择是请求分页系统设计的关键。
- 选择置换算法的原则是保证系统的缺页次数最少。
- 为了评价一个算法的优劣，可将该算法应用到一个特定的内存访问序列（引用串），并计算缺页次数。可以人工生成内存引用串或者通过跟踪一个给定的系统并记录每个内存引用的地址。后一种方法会产生大量的数据，为减少数据量，可采用以下方式进行化简：（1）对给定页大小只需要考虑页码，而不需要完整地址。（2）如果有一个对页面*p*的引用，那么任何紧随其后的对页面*p*的引用不会产生缺页。

§5.2 请求分页技术

- 例如，如果跟踪一个特定的进程，记录下如下的地址顺序（用十进制表示）：

0100，0432，0101，0612，0102，0103，0104，0101，0611，0102，0103，0104，0101，0610，0102，0103，0104，0101，0609，0102，0105

如果页的大小为100 B，那么就得到如下引用串：

1，4，1，6，1，6，1，6，1，6，1

- 针对某一特定引用串和页面置换算法，为了确定缺页数量，还需要知道可用内存块的数量。显然，随着可用内存块数量的增加，缺页的次数会相应的减少。

§5.3 页面置换算法

1.最佳置换算法（OPT Optimal）

- 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。采用最佳置换算法，通常可保证获得最低的缺页率。
- 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

内存块用完，选择哪个淘汰出去呢？

哈哈！你最远有机会被访问，那就先把你淘汰出去吧！

页7被换出，装入页2

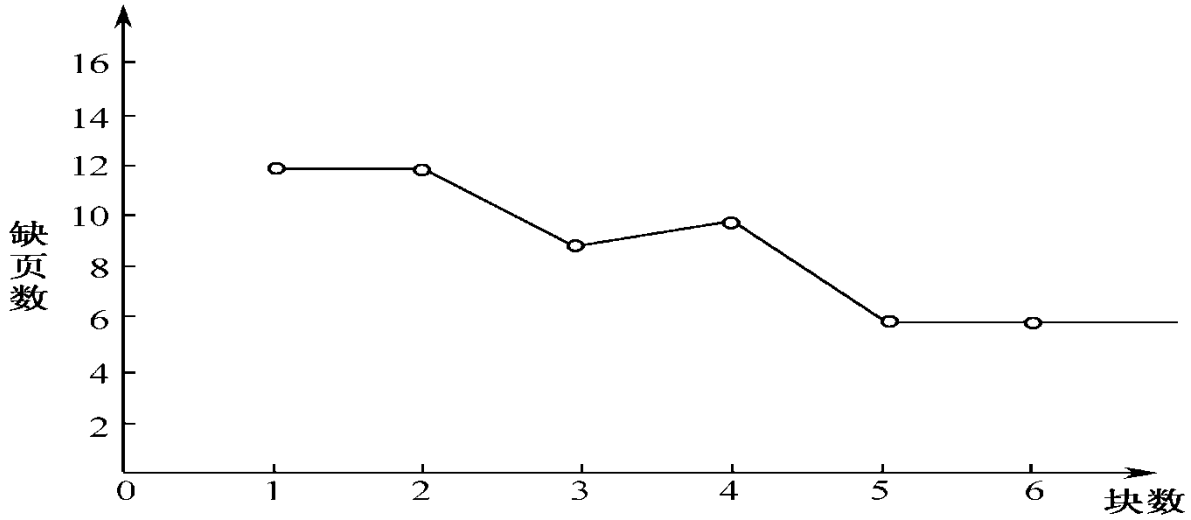
§ 5.3 页面置换算法

2. 先进先出(FIFO)页面置换算法

- **FIFO算法是最早出现的页面置换算法。该算法总是淘汰最先进入内存的页面，即选择在内存中停留时间最长（年龄最老）的一页予以淘汰 。**

§ 5.3 页面置换算法

- 为了说明FIFO页面置换算法相关的可能问题，考虑一下引用串：1，2，3，4，1，2，5，1，2，3，4，5。

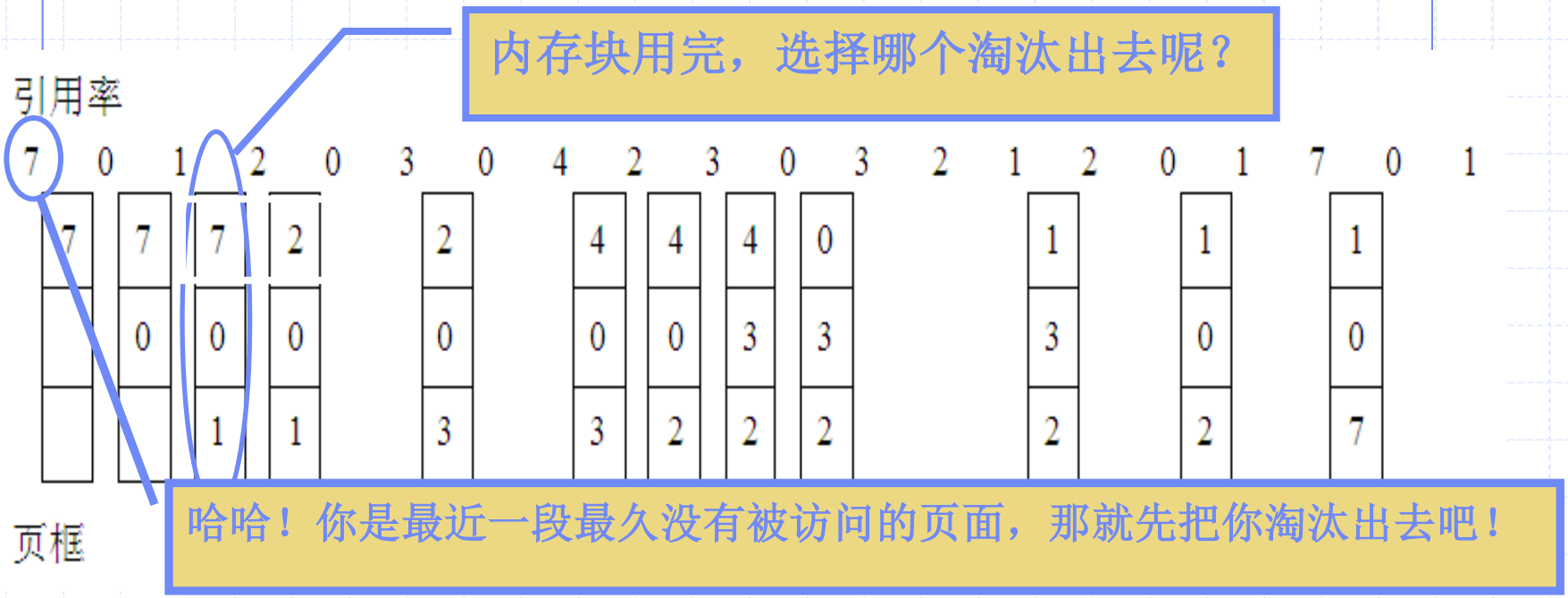


- 注意到对4个可用内存块的缺页次数（10）比3个内存块的缺页次数（9）还要大。这种令人难以相信的结果称为Belady异常现象，即缺页次数随内存块增加而增加。（原因：FIFO算法的置换特征与进程访问内存的动态特征是矛盾的，即被置换的页面并不是进程不会访问的。）

§5.3 页面置换算法

3. 最近最久未使用(LRU)置换算法

最近最久未使用置换算以“最近的过去”作为“不久将来”的近似，选择最近一段时间内最久没有使用的页面淘汰掉。它的实质是：当需要置换一页时，选择在最近一段时间里最久没有使用过的页面予以淘汰 。

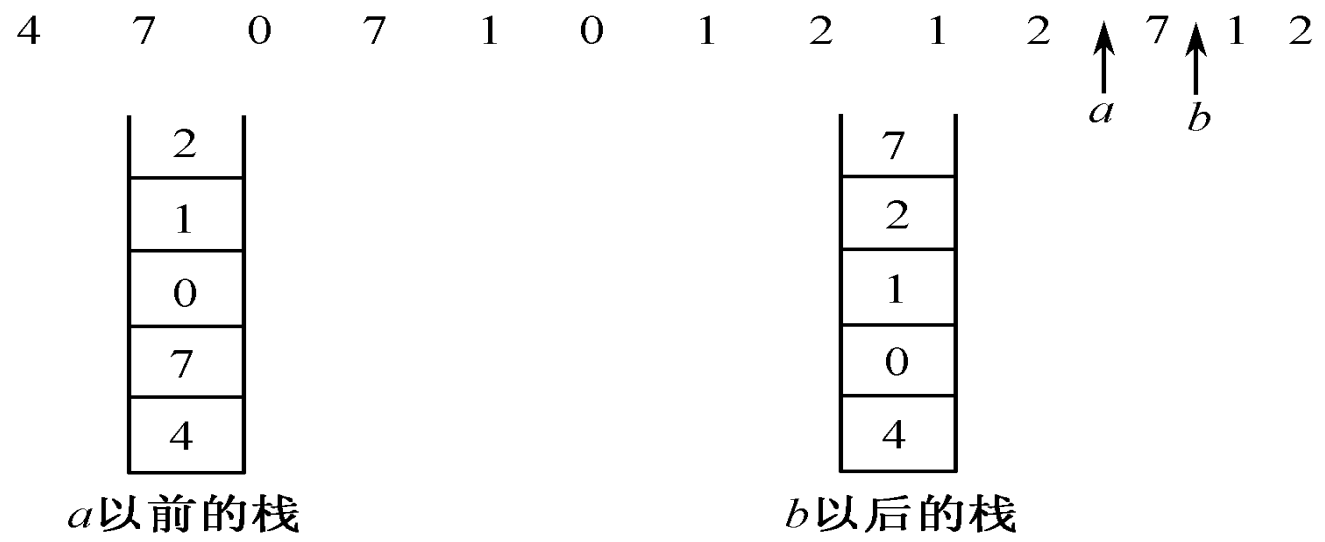


§5.3 页面置换算法

LRU算法需要实际硬件的支持。实现时的问题是：怎样确定最后访问以来所经历时间的顺序。有以下两种可行的办法：

（1）计数器。

（2） 栈。



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 5-6　某进程具有 8 个页面时的 LRU 访问情况

§5.3 页面置换算法

例题分析：

例1: 某页式虚拟存储管理系统的物理空间为3K，页面大小为1K，一进程按下列地址顺序引用内存单元：3635、3632、1140、3584、2892、3640、40、2148、1700、2145、3209、0、1102、1100。如果上述数字均为十进制，而内存尚未装入任何页，试计算采用OPT、LRU和FIFO页面置换算法的缺页次数。

§5.3 页面置换算法

例2:内存分配一页，初始时矩阵数据均不在内存；页面大小为128个整数；矩阵$A_{128X128}$按行存放。这两个程序执行时分别会产生多少次缺页中断？

程序编制方法1：

```
for j:=1 to 128
    for i:=1 to 128
        A[i,j]:=0;
```

程序编制方法2：

```
for i:=1 to 128
    for j:=1 to 128
        A[i,j]:=0;
```

§5.3 页面置换算法

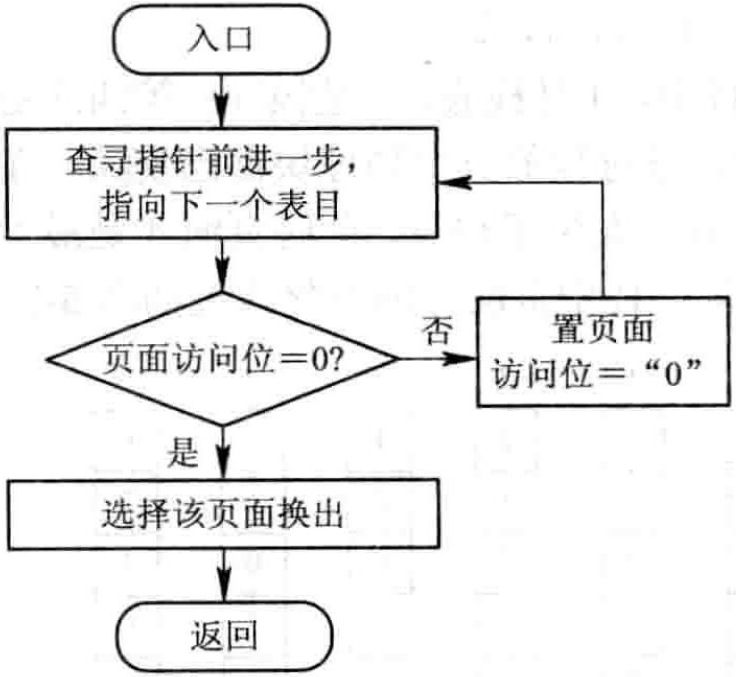
4. LRU的近似算法

（1）附加引用位算法

- 通过在规定时间间隔里记录引用位，能获得额外顺序信息。可以为位于内存中的每个页表中的每一页保留一个8 bit的字节。在规定的时间间隔（如每100ms）内，时钟定时器产生中断并将控制权交给操作系统。操作系统把每个页的引用位转移到其8 bit字节的高位，而将其他位右移，并抛弃最低位。这些8 bit移位寄存器包含着该页在最近8个时间周期内的使用情况。如果移位寄存器含有00000000，那么该页在8个时间周期内没有使用；如果移位寄存器的值为11111111，那么该页在过去每个周期内都至少使用过一次。
- 具有值为11000100的移位寄存器的也要比值为01110111的页使用更为频繁。如果将这8 bit字节作为无符号整数，那么具有最小值的页为LRU页，可以被置换出去。如果这个数字不惟一，可以置换所有具有最小值的页或在这些页之间采用FIFO来选择替换。

§5.3 页面置换算法

（2）简单的Clock算法

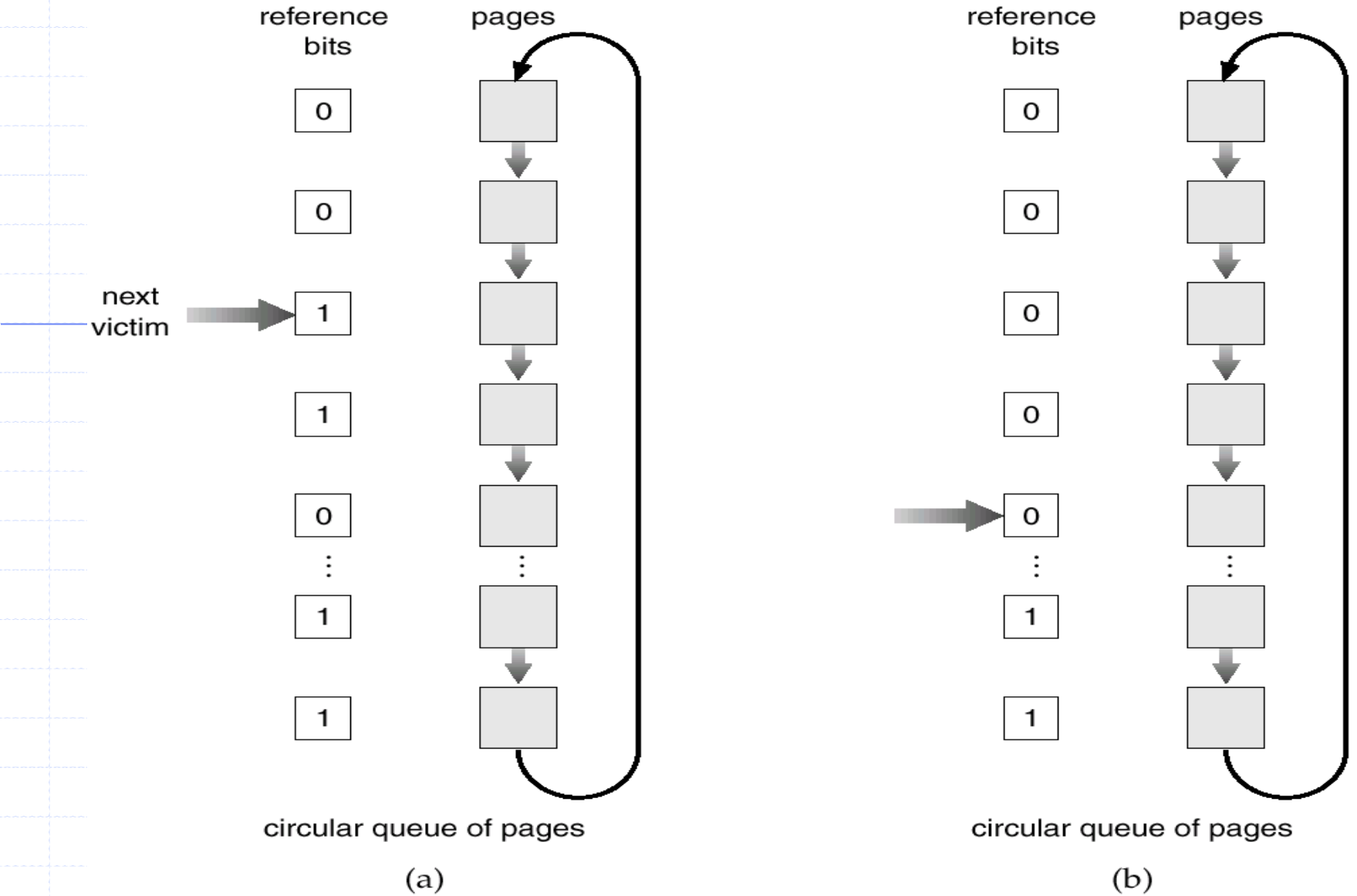


块号	页号	访问位	指针
0			
1			
2	4	0	
3			
4	2	1	
5			
6	5	0	
7	1	1	

替换
指针

图 5-8　简单 Clock 置换算法的流程和示例

§5.3 页面置换算法


reference bits
pages
0
0
next victim
1
1
0
1
1
circular queue of pages
(a)
reference bits
pages
0
0
0
0
0
1
1
circular queue of pages
(b)

§5.3 页面置换算法

（3）改进的Clock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。

§5.3 页面置换算法

在进行页面置换时，需要同时检查访问位和修改位，以确定该页是四类页面中的那一种。其执行过程可分成以下三步：

- 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。
- 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。
- 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。

§5.4 “抖动”与工作集

抖动问题

- 在虚存中，页面在内存与外存之间频繁调度，以至于调度页面所需时间比进程实际运行的时间还多，此时系统效率急剧下降，甚至导致系统崩溃。这种现象称为颠簸或抖动。
- 原因：
  - 页面淘汰算法不合理
  - 分配给进程的物理页面数太少

所谓工作集，是指在某段时间间隔 Δ 里，进程实际所要访问页面的集合。

引用页序列	窗口大小 3	4	5
24	24	24	24
15	15 24	15 24	15 24
18	18 15 24	18 15 24	18 15 24
23	23 18 15	23 18 15 24	23 18 15 24
24	24 23 18	—	—
17	17 24 23	17 24 23 18	17 24 23 18 15
18	18 17 24	—	—
24	—	—	—
18	—	—	—
17	—	—	—
17	—	—	—
15	15 17 18	15 17 18 24	—
24	24 15 17	—	—
17	—	—	—
24	—	—	—
18	18 24 17	—	—

工作集

重点概念和内容提示

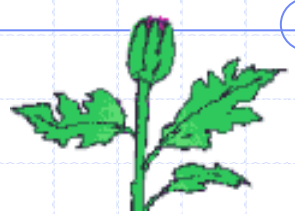
- 地址映射、虚拟地址空间、绝对地址空间、静态重定位和动态重定位、覆盖、交换、虚拟存储器、缺页中断、页表和段表
- 可变分区的实现思想、内存分配算法和内存回收
- 分页存储管理的原理和地址变换
- 分段存储管理的原理和地址映射
- 请求分页的原理和页面置换算法

§5.5 请求分段存储管理方式

1. 请求分段中的硬件支持
   - 请求段表机制
   - 缺段中断机构
   - 地址变换机构
2. 分段的共享和保护
   - 共享段表
   - 共享段的分配和回收
3. 分段保护

作业

1. 某虚拟存储器的用户空间共有32个页面，每页1KB，主存16KB。假定某时刻系统为用户的第0、1、2、3页分别分配的物理块号为5、10、4、7，试将虚拟地址1289、0A5CH和293CH、833CH变换为物理地址。



2. 假定系统为某进程分配了3个物理块，进程运行时的页面走向为 7, 1, 2, 1, 0, 4, 0, 3, 2, 4, 0, 3, 2, 1, 2, 1, 2, 7, 0, 1, 开始时3个物理块均为空，给出采用最佳置换算法时页面置换情况，并计算出该算法的缺页率？

（1）最佳置换淘汰算法

（2）先进先出淘汰算法

（3）最近最久未使用淘汰算法

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