5.2.3 页面调入策略

1. 何时调入页面

1) 预调页策略
2) 请求调页策略

2. 从何处调入页面

在请求分页系统中的外存分为两部分：**用于存放文件的文件区和用于存放对换页面的对换区**。通常，由于对换区是采用连续分配方式，而事件是采用离散分配方式，故对换区的磁盘I/O速度比文件区的高。这样，每当发生缺页请求时，系统应从何处将缺页调入内存，可分成如下三种情况

5.2.3 页面调入策略

2. 从何处调入页面

(1) 系统拥有足够的对换区空间，这时可以全部从对换区调入所需页面，以提高调页速度。为此，在进程运行前，便须将与该进程有关的文件，从文件区拷贝到对换区。

(2) 系统缺少足够的对换区空间，这时凡是不会被修改的文件，都直接从文件区调入；而当换出这些页面时，由于它们未被修改而不必再将它们换出，以后再调入时，仍从文件区直接调入。但对于那些可能被修改的部分，在将它们换出时，便须调到对换区，以后需要时，再从对换区调入。

(3) UNIX方式。由于与进程有关的文件都放在文件区，故凡是未运行过的页面，都应从文件区调入。而对于曾经运行过但又被换出的页面，由于是被放在对换区，因此在下次调入时，应从对换区调入。由于UNIX系统允许页面共享，因此，某进程所请求的页面有可能已被其它进程调入内存，此时也就无须再从对换区调入。

4. 缺页率

假设进程的逻辑空间为n页，系统为其分配的内存块为m(m<=n)，进程运行中访问页面成功次数为S，失败次数为F，则进程总的访页次数就是A=S+F，定义缺页率为f=F/A。

影响缺页率的因素有：

(1) 页面大小

(2) 进程所分配物理块的数目

(3) 页面置换算法

(4) 程序固有特性（局部化程度）

缺页中断处理时间：$t = \beta \times t_a + (1-\beta) \times t_b$

其中：置换页面被修改过的概率是β，缺页处理时间是t_a；置换页面未被修改过的概率是1-β，缺页处理时间t_b。

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。采用最佳置换算法，通常可保证获得最低的缺页率。

OPT置换算法示例：

假定系统为某个进程分配了三个物理块，进程的访问顺序为7，0，1，2，0，3，0，4，2，3，0，3，2，1，2，0，1，7，0，1

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

0，1，7，0，1				
2	2	7	7	7
0	0	0	0	0
1	1	1	1	1
		*		

缺页次数9，成功访问次数11次，

缺页率f=9/20=45%

5.3.1 最佳置换算法和先进先出置换算法

2.FIFO淘汰算法示例：

7，0，1，2，0，3，0，4，2，3，0，3，2，1，2

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

0，1，7，0，1

0	0	7	7	7
1	1	1	0	0
2	2	2	2	1
		*	*	*

3块内存时，缺页率f=15/20=75%；

5.3.2 最近最久未使用和最少使用置换算法

1. LRU(Least Recently Used)置换算法的描述

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

0,	1,	7,	0,	1
1	1	1	1	1
0	0	0	0	0
2	2	7	7	7
*		*		

12次，缺页率

f=12/20=60%

5.3.2 最近最久未使用和最少使用置换算法

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

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

5.3.2 最近最久未使用和最少使用置换算法

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

5.3.2 最近最久未使用和最少使用置换算法

2) 栈

4，7，0，7，1，0，1，2，1，2，6

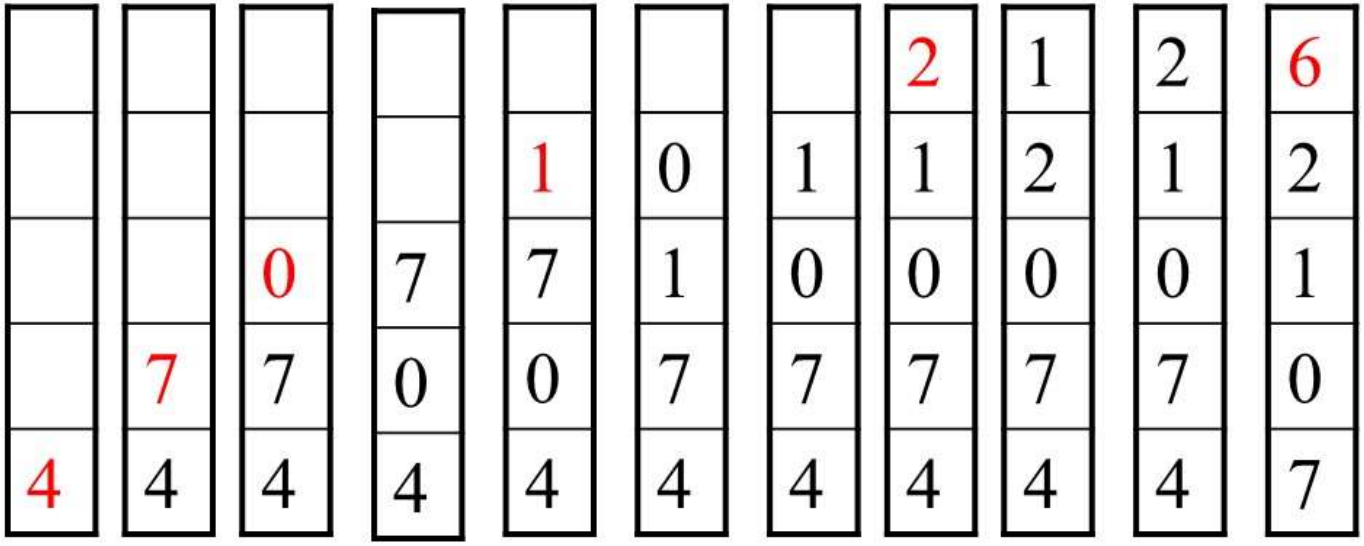


图 5-7 用栈保存当前使用页面时栈的变化情况

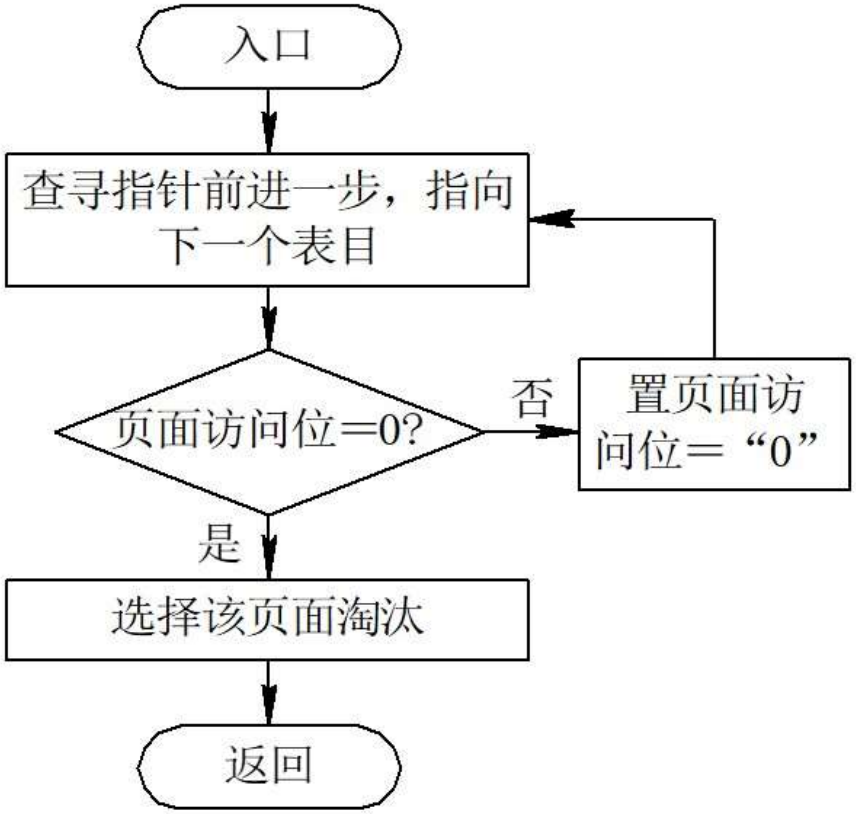
6次，缺页率f=6/11=54.54%

3.最少使用(LFU： Least Frequently Used)置换算法

最少使用置换算法是为每个页面设置一个移位寄存器，用来记录该页面被访问的频率，该页被访问则高位置1，每隔一定时间（如100ms）右移一位，在一段时间内总是淘汰最少使用的页面。

5.3.3 Clock置换算法

1. 简单的Clock置换算法（最近未用先淘汰）



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

5.3.3 Clock置换算法

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。

5.3.3 Clock置换算法

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 页面缓冲算法(PBA: Page Buffering Algorithm)

1. 影响页面换进换出效率的若干因素

(1) 页面置换算法 —— 好的置换算法会具有较低的缺页率。

(2) 写回磁盘的频率 —— 设置已修改链表，减少启动磁盘操作次数。

(3) 读入内存的频率 —— 换入的页若属于已修改链表中的，就可以减少换入开销。

2. 页面缓冲算法(PBA: Page Buffering Algorithm)

(1) 空闲页面链表：被淘汰的未修改的页放置表尾。

(2) 修改页面链表

该算法的特点：

(1) 降低页面换入、换出频率。

(2) 可采用简单置换算法（如FIFO），实现容易。

5.3.5 访问内存的有效时间

请求分页管理中，内存有效访问时间(EAT)由三部分构成：访问页表、访问实际物理地址数据的时间(t)以及缺页中断处理时间。

据此内存访问操作有三种方式：

1. 被访问页在内存，且其对应的页表项在快表中。

此时无缺页，EAT= λ +t

λ 为访问快表时间。

2. 被访问页在内存，且其对应的页表项不在快表中。

EAT= λ +t+ λ +t=2*(λ +t)

查找快表、页表、修改快表及访问物理单元时间。

3. 被访问页不在内存。

EAT= λ +t+ ε + λ +t　　这里 ε 为缺页中断处理时间；

考虑到快表命中率(a)和缺页率(f)，则：

EAT=a*(λ +t)+(1−a)*[f*(ε +2(λ +t))+(1−f)*2(λ +t)]

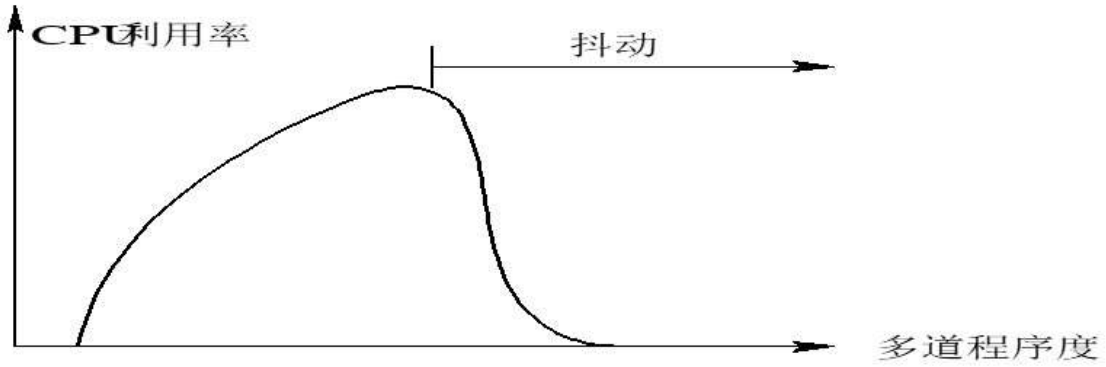
=(1−a)*f* ε +(2−a)(λ +t)　　余略。

5.4.1 多道程序度与 “抖动”

1. 多道程序度与处理机的利用率

操作系统会对CPU的工作情况进行监督，如果发现CPU出现空闲，它会调入新的进程来增加多道程序度（内存中并发执行的进程数目），保持CPU的高利用率。

但实验表明当多道程序数量增加到一定数量，CPU的利用率会攀升到一个高点，随后会呈现下降趋势见下图。



2. 产生”抖动”的原因

在采用全局置换的方式下，它会导致其它进程的某些页被置换出内存，而该进程执行时会因此产生缺页，所以它又会置换另外进程的页，由此造成连锁反应，使得整个系统中页面置换频繁发生。

在每次置换过程中，都需要启动磁盘I/O，这种低速的延迟操作会造成CPU等待，操作系统发现CPU空闲后，又开始增加多道程序度…… 于是整个系统就在进行频繁的页面置换，这种状况就称为“抖动”，它会严重地影响到系统的性能。

5.5.1 请求分段中的硬件支持

1. 请求段表机制

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。

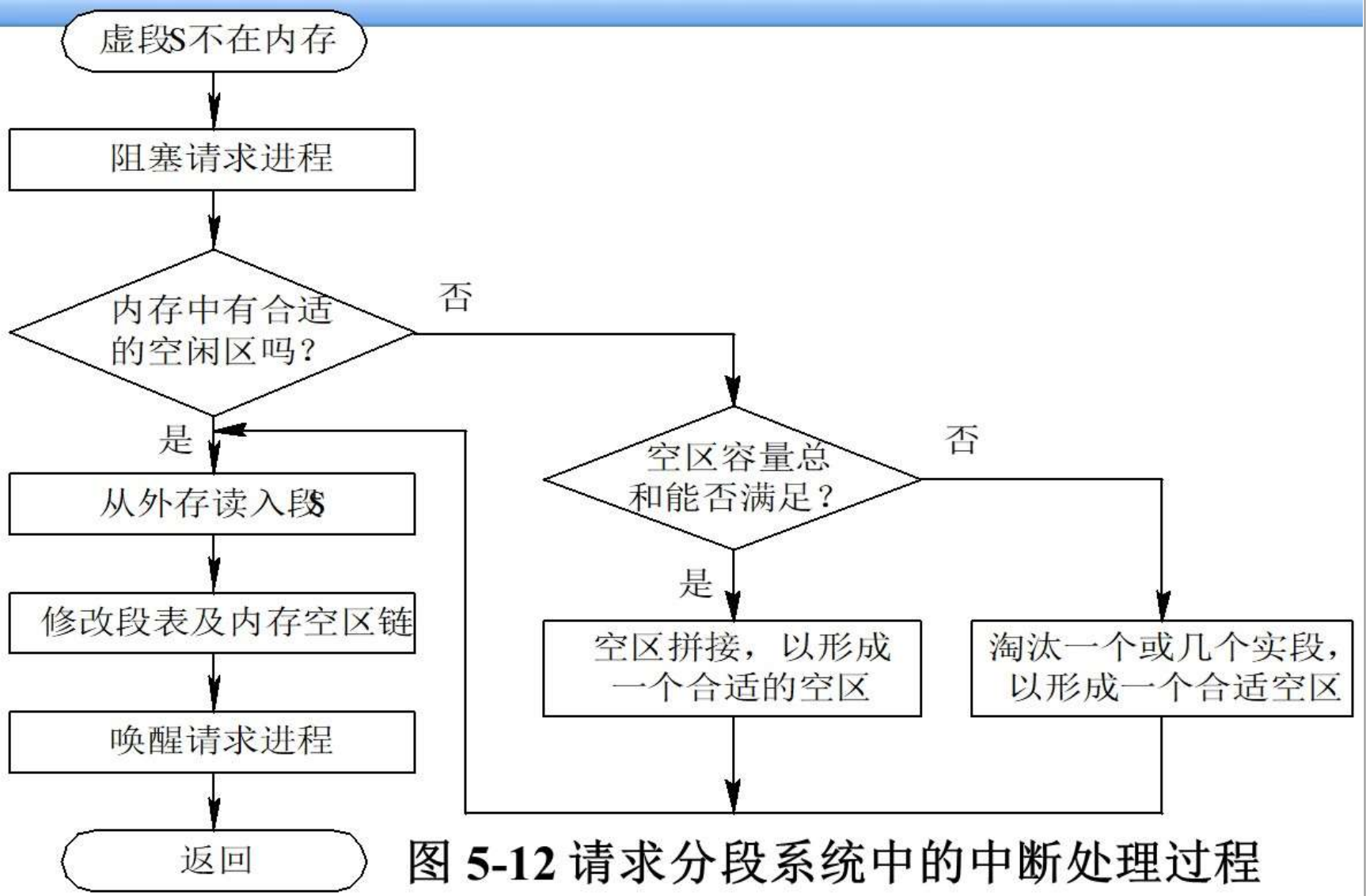


图 5-12 请求分段系统中的中断处理过程

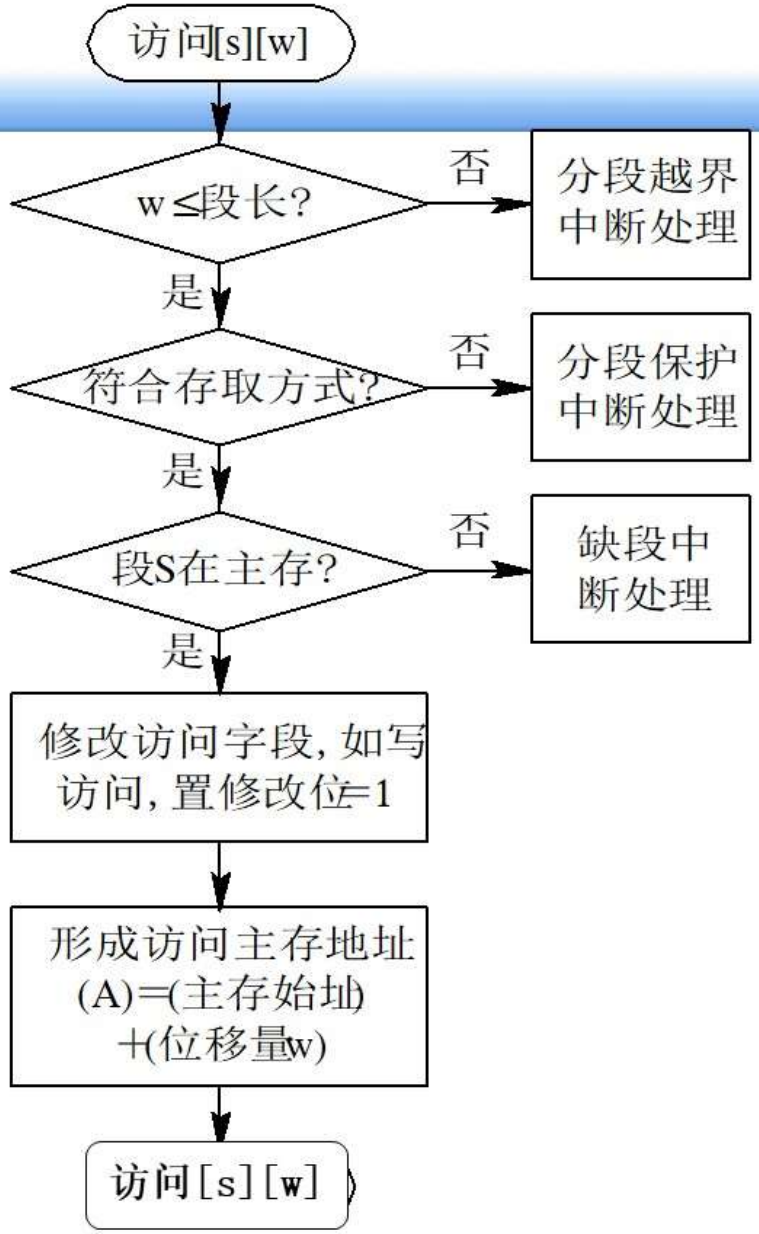
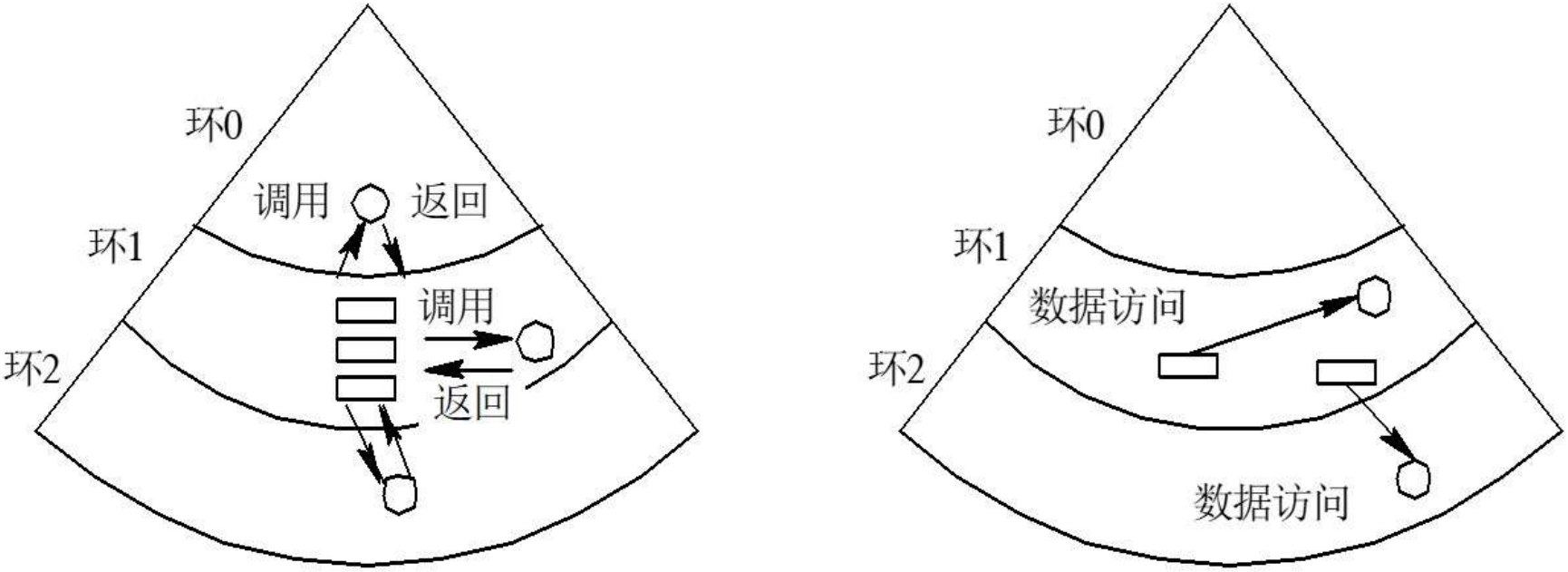


图 5-13 请求分段系统的地址变换过程

5.5.2 分段的共享与保护



(a) 程序间的控制传输　　(b) 数据访问

图 5-15 环保护机构