“放”的总结

- 连续存放：
  - 单道连续划分；
  - 多道连续固定划分；
  - 多道连续可变划分。

- 不连续存放：
  - 页式存储；
  - 段式存储；
  - 段页式存储。

5.3 虚存

目的：提供用户进程一个巨大的虚拟存储空间。

手段：利用外存(磁盘)实现此虚空间。

5.3.1 虚存的基本思想

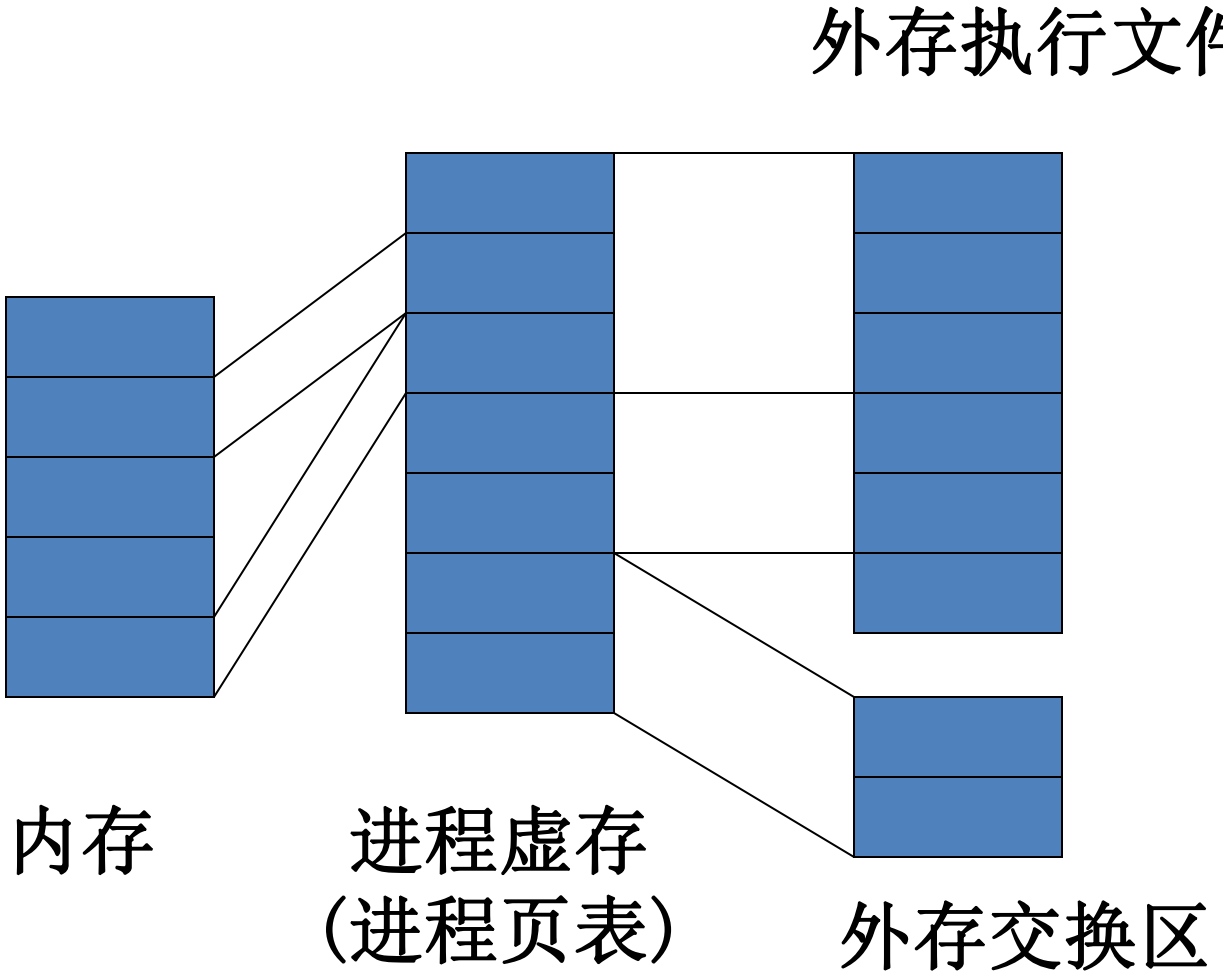
系统为进程提供一个比物理内存大得多的虚拟存储空间，逻辑空间大小不受物理内存大小的限制。

逻辑空间的容量由系统的有效地址长度决定。假设地址长度为32字节，按字节寻址，则逻辑空间（虚存空间）大小为2^{32}个字节

实现虚空间的基本方法

在页式（段式、段页式）管理的基础上，仅将进程的一部分页（段）放于主存。页（段）表项中注明该页或段是否在主存。程序执行时，如果访问的页（段）不存在主存，根据页（段）表项的指示，将其从外存调入主存，如果此时无可用的内存空间，则先淘汰若干页帧或段。

页式虚存示意



交换区（SWAP）：进程刚建立时，页面所在辅存即程序文件所在的辅存位置。但程序文件中一般包含有程序的二进制目标码及数据初始值和初值为0的工作区。后两者在回写时不能写入程序文件，因此引入了交换区。

5.3.2 页式虚存管理

一、页表项结构:

合法位	修改位	页类型	保护码	内存块号	页号

合法位：置该位表示该页在内存。

修改位：置该位表示该页被修改过，在释放或淘汰时应写回外存。

页类型：零页时，表示该页在分配物理页帧时应清0页帧空间;回写swap区页时，表示回写swap区。

保护码：R,W,E保护说明。

内存块号：当在合法位置上时，该页所在内存的块号。

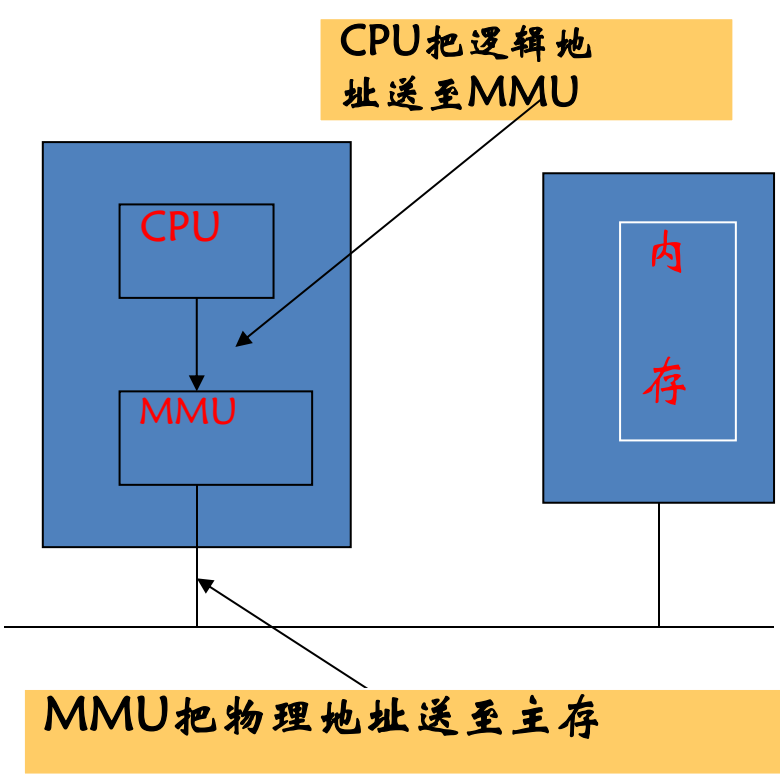
页号：逻辑页号。

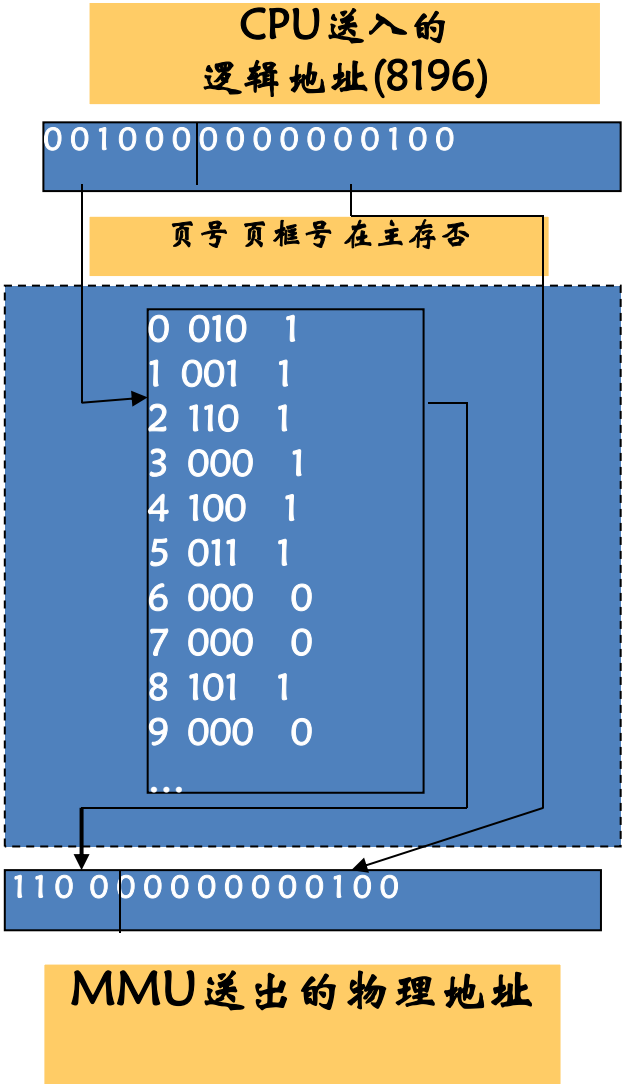
二、硬件动态地址转换

在执行虚存访问指令时,由硬件合成物理地址。若能在联想存储器中获得该虚页的物理页帧号，则访问之。若要查当前进程页表，须先检查该页页表项的合法位，该位若置上，则从页表项中获得页帧号，否则要发一个页故障(page fault)信息或叫做缺页中断，当缺页中断处理完后，重新执行访存指令.

联想存储器中的页表项都是合法页的页表项.

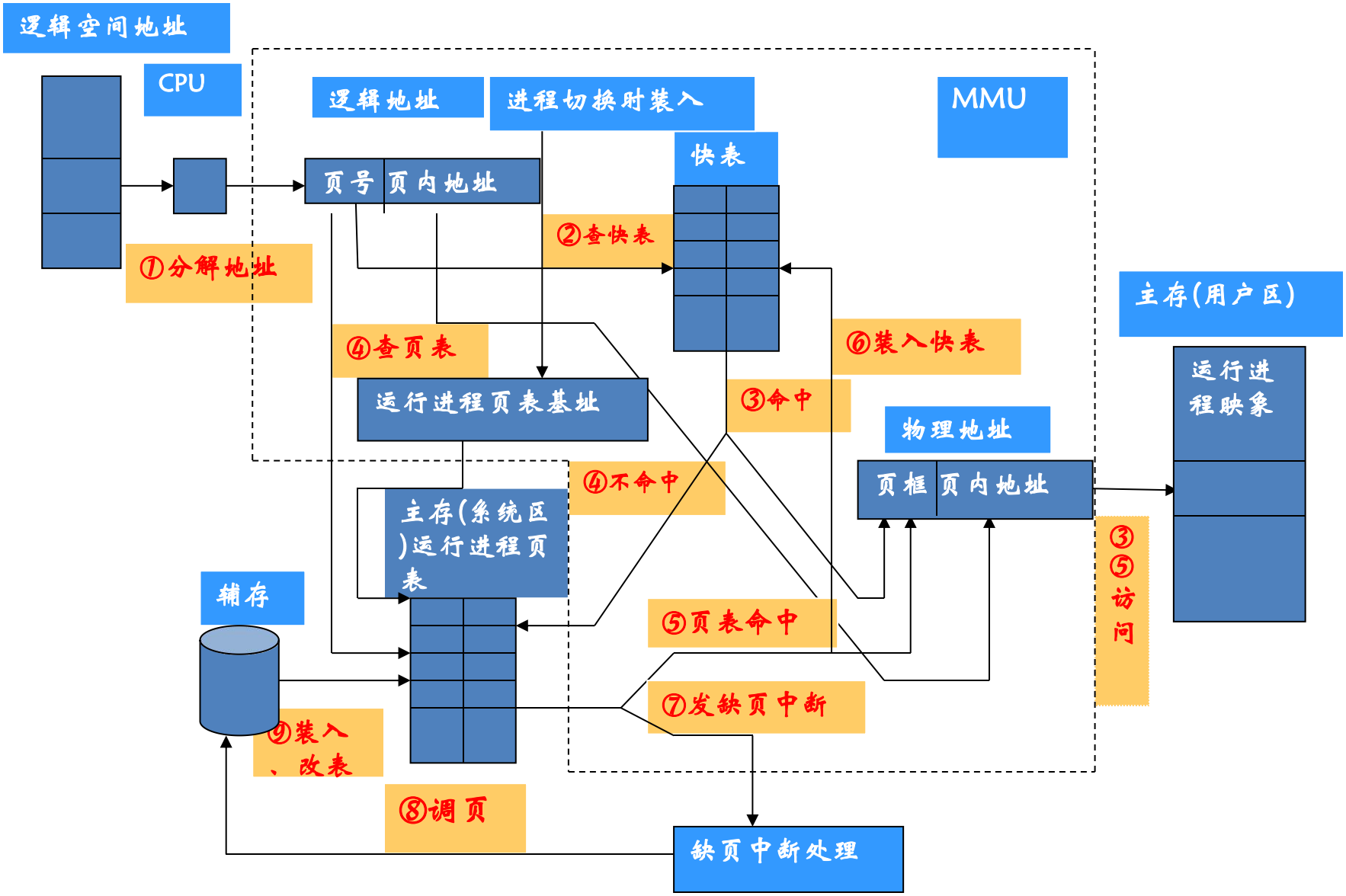
分页式虚拟存储系统的硬件支撑



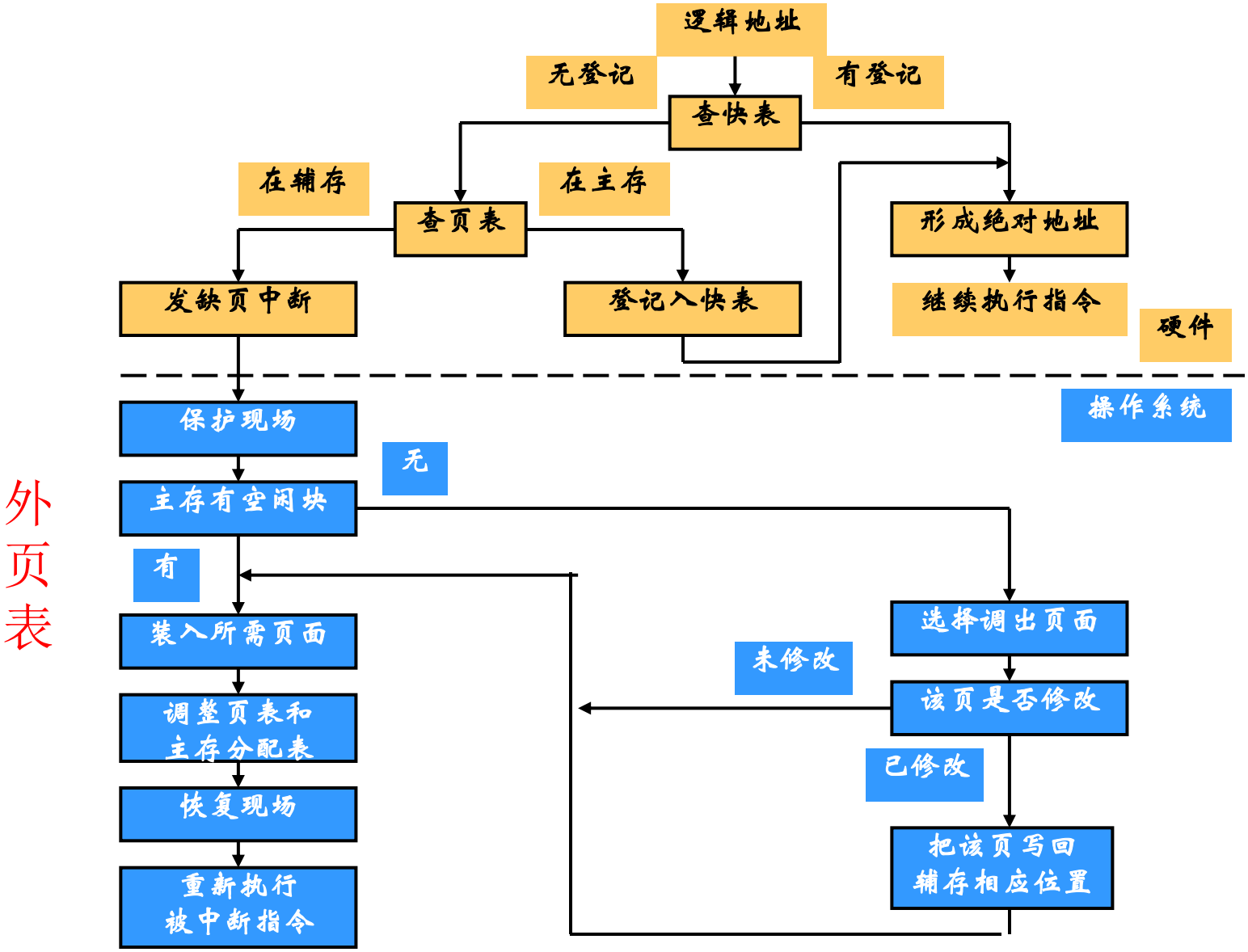


MMU的位置、功能和16个4KB页面情况下MMU的内部操作

请求分页虚存地址转换过程(1)



请求分页虚存地址转换过程(2)



三、缺页处理

缺页中断处理程序：

1. 根据发生页故障的虚地址得到页表项。

2. 申请一个可用的页帧(根据所采用的替换策略可能需要引起淘汰某一页)；

3. 检查页类型，若为零页，则将页帧清0，将页帧号填入页表项的页帧号一栏，置合法位为1。若非零页，则调用I/O子系统将外存块号所指的数据读到可用页帧，将页帧号填入页表项中，合法位置1，结束。

四、页淘汰

页淘汰可以发生在申请页帧时，而现代OS一般都定时进行页淘汰。由页面替换策略决定，若已决定淘汰页P，则淘汰一页的主要工作有：

1. 查P页表项的修改位，若未修改，则合法位清0，将页帧送回空闲页帧队列。
2. 若已修改，则检查类型栏。
3. 若是零页或回写swap区页，则申请一块swap区空间，将P的外存块号置上。
4. 调用I/O子系统，将页帧上的数据写到外存块号所指的外存空间。合法位清0，将页帧送回空闲页帧队列。

五、页面替换策略

虚存的作用：

- 解决主存空间不足；
- 让更多的进程并发运行，提高系统的吞吐率。

页故障引发：

- Page Out / Page In；
- 访问辅存。

必须防止系统发生抖动（控制页故障）。

页面分配策略：固定分配

- 进程保持页框数固定不变，称**固定分配**；
- 进程创建时，根据进程类型和程序员的要求决定页框数，只要有一个缺页中断产生，进程就会有一页被替换。

页面分配策略：可变分配

- 进程分得的页框数可变，称**可变分配**；
- 进程执行的某阶段**缺页率**较高，说明目前**局部性**较差，系统可多分些页框以降低缺页率，反之说明进程目前的局部性较好，可减少分给进程的页框数

页面替换策略：局部替换和全局替换

- 如果页面替换算法的作用范围是整个系统，称全局页面替换算法，它可以在运行进程间动态地分配页框。
- 如果页面替换算法的作用范围局限于本进程，称为局部页面替换算法，它实际上需要为每个进程分配固定的页框。

固定分配和局部替换策略配合使用(1)

- 进程分得的页框数不变，发生缺页中断，只能从进程的页面中选页替换，保证进程的页框总数不变。
- 策略难点：应给每个进程分配多少页框?给少了,缺页中断率高;给多了,使主存中能同时执行的进程数减少，进而造成处理器和其它设备空闲。

固定分配和局部替换策略配合使用(2)

采用固定分配算法，系统把页框分配给进程，采用：

①平均分配，

②按比例分配，

③优先权分配。

可变分配和全局替换策略配合使用

- 先每个进程分配一定数目页框，OS保留若干空闲页框，进程发生缺页中断时，从系统空闲页框中选一个给进程，这样产生缺页中断进程的主存空间会逐渐增大，有助于减少系统的缺页中断次数。
- 系统拥有的空闲页框耗尽时，会从主存中选择一页淘汰，该页可以是主存中任一进程的页面，这样又会使那个进程的页框数减少，缺页中断率上升。
- SVR4

可变分配和局部替换配合使用

其实现要点如下:

(1)新进程装入主存时，根据应用类型、程序要求，分配给一定数目页框,可用请页式或预调式完成这个分配。

(2)产生缺页中断时,从该进程驻留集中选一个页面替换。

(3)不时重新评价进程的分配，增加或减少分配给进程的页框以改善系统性能。

(4)Windows NT

影响缺页中断率的因素(1)

- 假定作业p共计n页，系统分配给它的主存块只有m块（1≤m≤n）。如果作业p在运行中成功的访问次数为S，不成功的访问次数为F，则总的访问次数A为：

$$A = S + F$$

称f为缺页中断率，定义：

$$f = F / A$$

影响缺页中断率的因素(2)

影响缺页中断率f的因素有：

(1)主存页框数。

(2)页面大小。

(3)页面替换算法。

(4)程序特性。

程序局部性例子

- 程序将数组置为“0”，假定仅分得一个主存块，页面尺寸为128个字，数组元素按行存放，开始时第一页在主存。
- int A[128][128];　　int A[128][128];
- for(int j=0;j<128;j++)　　for(int i=0;i<128;i++)
  for(int i=0;i<128;i++)　　for(int j=0;j<128;j++)
- A[i][j]=0;　　A[i][j]=0;
- 缺页中断次数：
- 128×128-1　　128-1

六、 页面替换算法

1) 先进先出页面替换算法FIFO

2) 最佳页面替换算法OPT

3) 最近最少用页面替换算法LRU

4) 第二次机会页面替换算法SCR

5) 时钟页面替换算法Clock

页面替换策略中的基本概念

驻留集(工作集)：进程的合法页集合

访问串：进程访问虚空间的地址踪迹

举例：某进程依次访问如下地址，0100，0432，0101，0612，0102，0103，……

页式虚存管理以页为基本单位，只需页号即可。设页面大小为100，上述访问串可简化为1，4，1，6，1，1，……

页面替换策略分类：

- 驻留集大小固定的替换策略
- 驻留集大小可变的替换策略

设驻留集大小为m，$s(t)$为t 时刻的驻留集，$r(t)$为t 时刻访问的页号。t取$0, 1, \ldots, t$，指访存指令执行时刻。

一、驻留集大小固定的替换策略

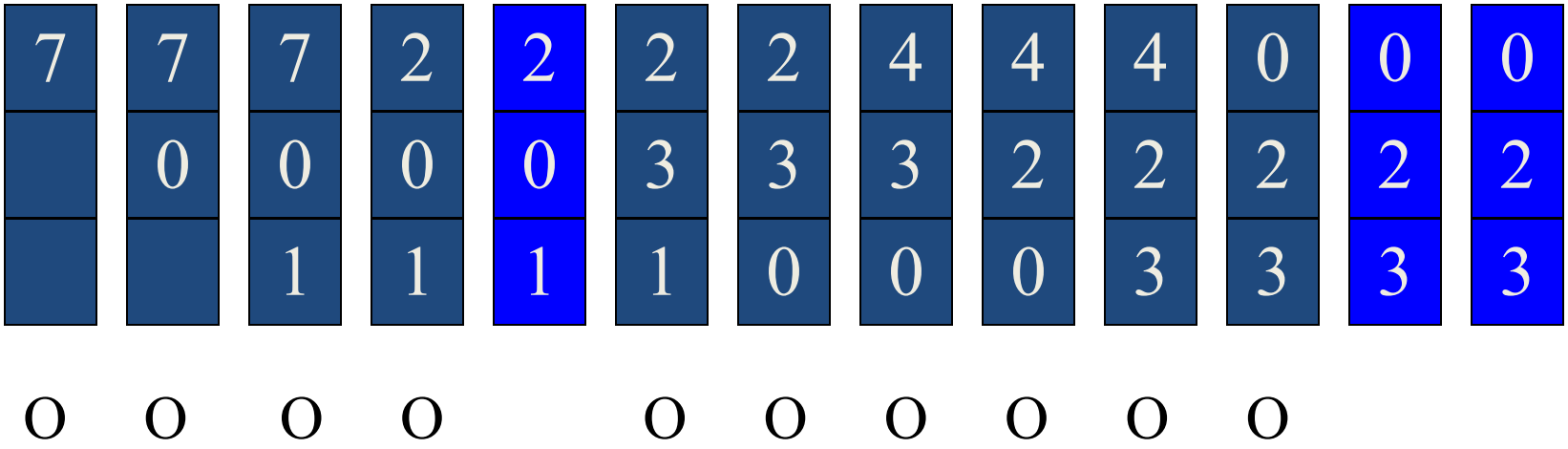
驻留集与paging in/out的关系：

- 进程刚创建时，驻留集为空。即$s(t)$=空。
- 若t+1时刻访问的页在$s(t)$中时，则访问之。即若$r(t+1) \in s(t)$，则$s(t+1)= s(t)$。
- 若t+1时刻访问的页不在$s(t)$中时，且驻留集大小小于m，则paging in。即若$r(t+1)! \in s(t)$，且$|s(t)|<m$，则$s(t+1)=s(t)+\{r(t+1)\}$。
- 若t+1时刻访问的页不在$s(t)$中时，且驻留集大小等于m，则先paging out y页，再paging in $r(t+1)$页。即$s(t+1)=s(t)-\{y\}+\{r(t+1)\}$。

(1) FIFO替换算法(替换最早进入的页)

举例：驻留集大小为3，访问串为：

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



FIFO方法的特点：

- 实现方便。不需要额外硬件。
- 效果不好，有Belady奇异。

Belady奇异：指替换策略不满足随着驻留集的增大，页故障数一定减少的规律。

Belady异常现象(一)

- 产生：一般是内存中的物理页面数越多，产生缺页次数越少。
- 例：一个程序使用五个虚页面，0~4，访问次序：
  0 1 2 3 0 1 4 0 1 2 3 4

  用三个实页：

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

  页1： 0 1 2 3 0 1 4 4 4 2 3 3

  页2： 0 1 2 3 0 1 1 1 4 2 2

  页3： 0 1 2 3 0 0 0 1 4 4

缺页中断* * * * * * * * 共9次

Belady异常现象(二)

用4个实页时：

	0	1	2	3	0	1	4	0	1	2	3	4
页1：	0	1	2	3	3	3	4	0	1	2	3	4
页2：		0	1	2	2	2	3	4	0	1	2	3
页3：			0	1	1	1	2	3	4	0	1	2
页4：				0	0	0	1	2	3	4	0	1
中断：	*	*	*	*			*	*	*	*	*	*

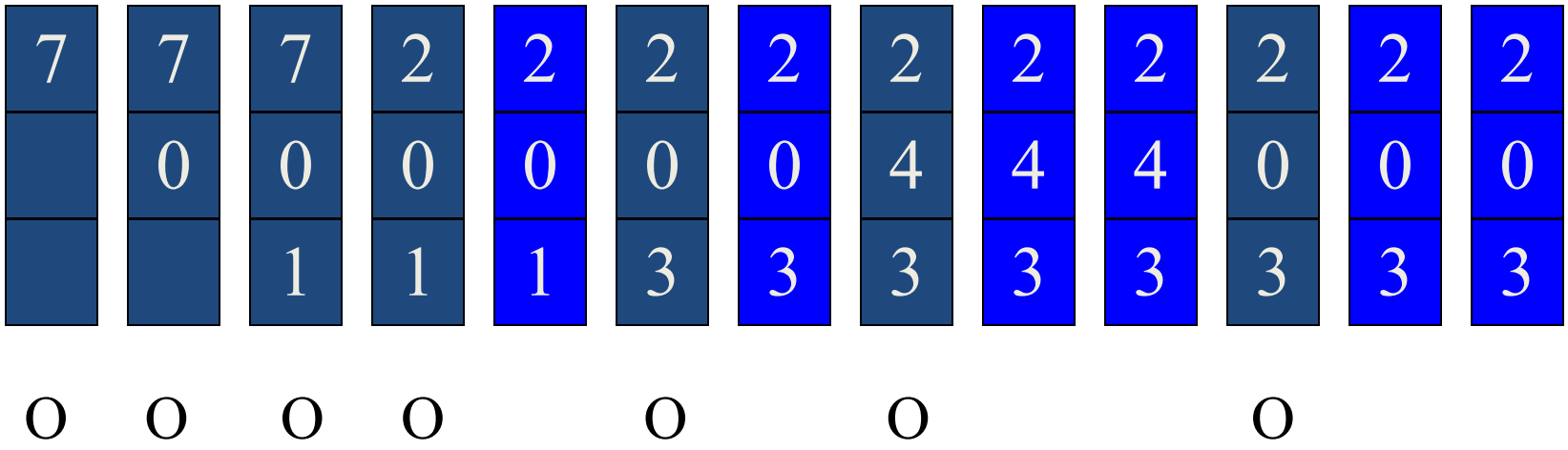
中断次数：10次

(2) OPT(Optimal replacement)

淘汰下次访问距当前最远的那些页中序号最小的页。

举例：驻留集大小为3，访问串为

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



OPT方法特点：

- 最优的固定驻留集大小替换策略。
- 不可实现。

OPT策略对任意一个访问串的控制均有最小的时空积（进程所占空间与时间的乘积）。

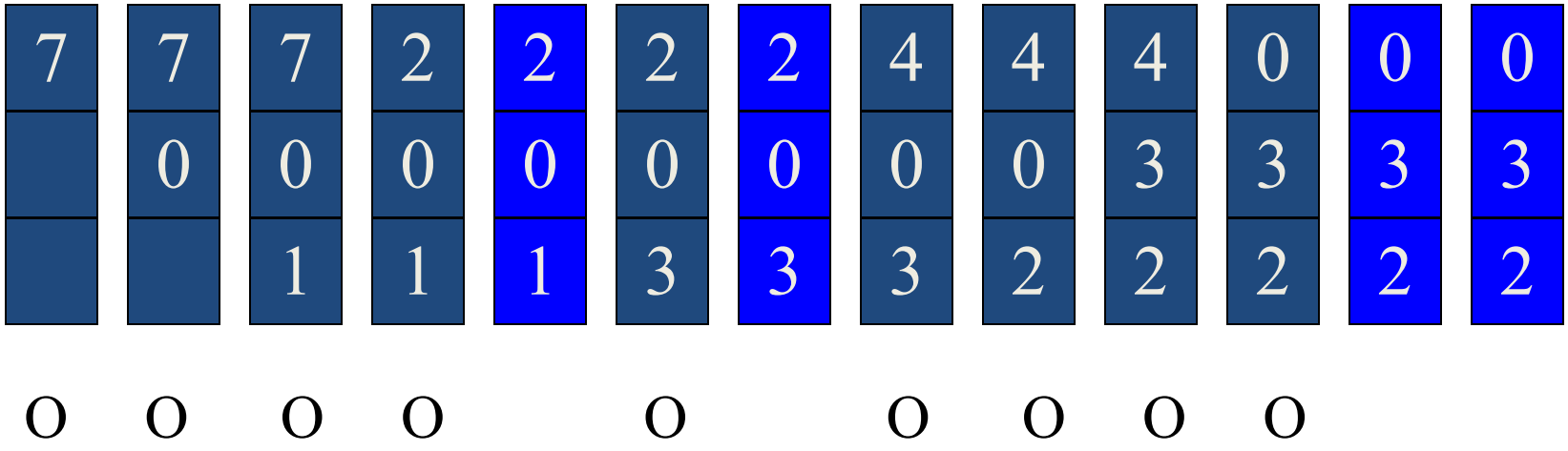
由于需要预先得知整个访问串的序，故不能用于实践,仅作为一种标准，用以测量其他可行策略的性能。

(3) LRU(Least Recently Used)

淘汰上次使用距当前最远的页。

举例：驻留集大小为3，访问串为

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



LRU策略是一种栈算法。

满足：$S(m, t)$ 属于 $S(m+1, t)$ 的替换算法被称为栈算法。

LRU策略中，当驻留集大小为m时，$S(m, t)$ 中保持着最近使用过的m个页帧；当驻留集大小为$m+1$时，$S(m+1, t)$ 中保持着最近使用过的$m+1$个页帧。故$S(m, t)$ 属于 $S(m+1, t)$ 的LRU策略是栈算法。

栈算法没有Belady奇异。

设$n>m$，对于栈算法有$S(m, t)$属于$S(n, t)$，任取$r(t)$，若$r(t)!\in S(n, t)$，则$r(t)!\in S(m, t)$。因此，驻留集为n 时出现的页故障一定会出现在驻留集为m 时。

LRU没有Belady奇异。

LRU策略的特点：要硬件配合，实现费用高，但效果适中。

实现方法1：给每个页帧设一个计数器，每访问一页，对应页帧计数清0，其余页帧计数加1，淘汰计数最大的页帧。

实现方法2：用类似栈的结构来管理和实现LRU。

4)第二次机会页面替换算法

改进FIFO算法,把FIFO与页表中的”引用位”结合起来使用:

- 检查FIFO中的队首页面(最早进入主存的页面),如果它的”引用位”是0,这个页面既老又没有用，选择该页面淘汰;
- 如果”引用位”是1,说明它进入主存较早，但最近仍在使用。把它的”引用位”清0,并把这个页面移到队尾，把它看作是一个新调入的页。
- 算法含义：最先进入主存的页面，如果最近还在被使用的话,仍然有机会作为像一个新调入页面一样留在主存中。

5）时钟页面替换算法(1)

算法实现要点(1)：

把SCR算法中的FIFO队列改为循环页面队列，减少入队出队操作。

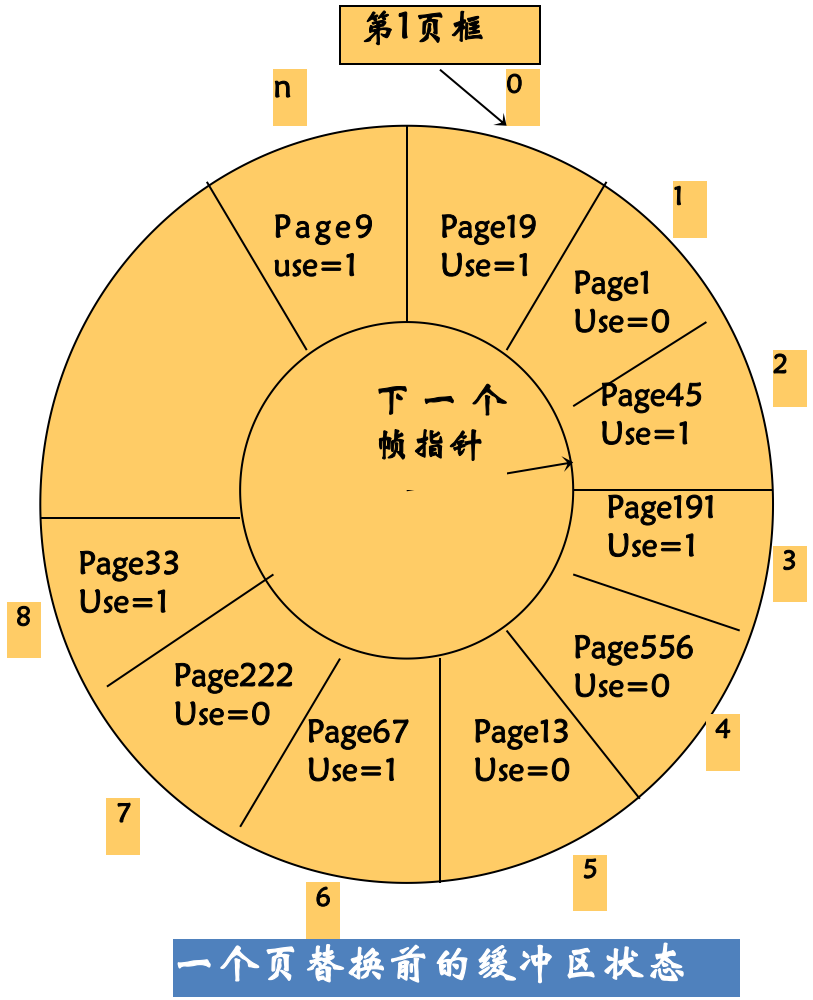
- 一个页面首次装入主存，其“引用位”置0。
- 主存中的任何页面被访问时,”引用位”置1。
- 淘汰页面时,从指针当前指向的页面开始扫描循环队列，把迁到的”引用位”是1的页面的”引用位”清0,跳过这个页面;把所迁到的”引用位”是0的页面淘汰掉,指针推进一步。

时钟页面替换算法(2)

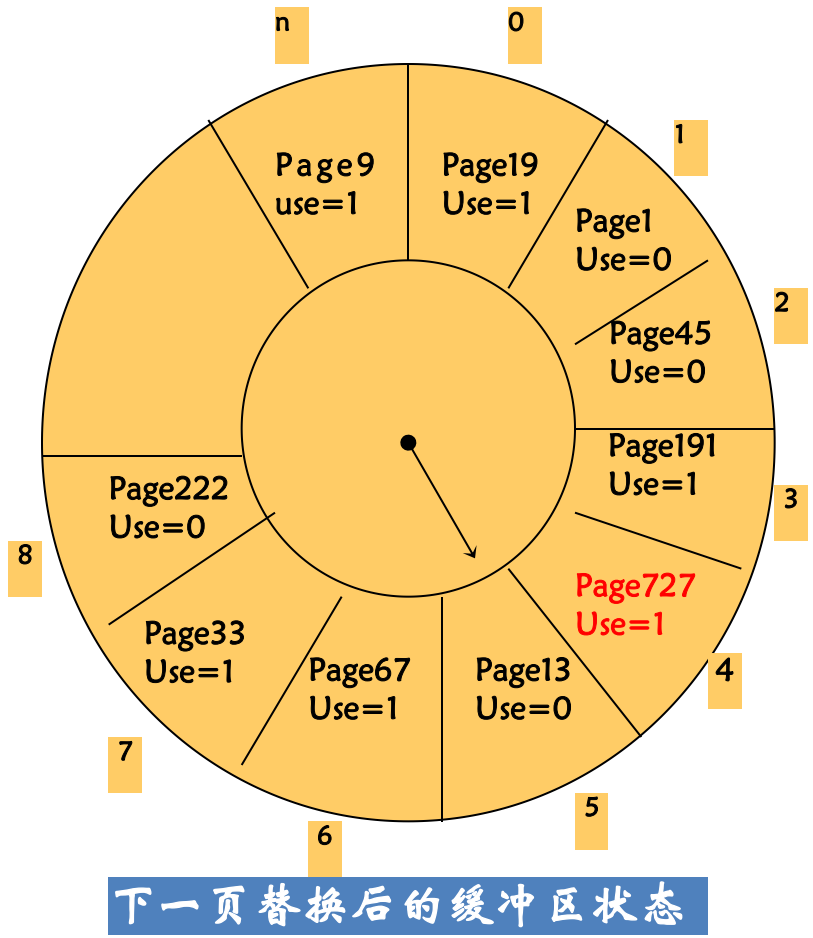
算法实现要点(2)：

- 扫描循环队列时，如果迁到的所有页面的”引用位”为1，指针就会绕整个循环队列一圈，把碰到的所有页面的”引用位”清0;指针停在起始位置，并淘汰掉这一页,然后，指针推进一步。

时钟页面替换算法的一个例子



一个页替换前的缓冲区状态



下一页替换后的缓冲区状态

时钟页面替换改进算法(1)

- 把”引用位”和”修改位”结合起来使用，共组合成四种情况：

(1)最近没有被引用，没有被修改(r=0,m=0)

(2)最近被引用,没有被修改(r=1,m=0)

(3)最近没有被引用,但被修改(r=0,m=1)

(4)最近被引用过,也被修改过(r=1,m=1)

肘钟页面替换改进算法(2)

- 步1：选择最佳淘汰页面，从指针当前位置开始,扫描循环队列。扫描过程中不改变”引用位”，把迁到的第一个r=0,m=0的页面作为淘汰页面。
- 步2：如果步1失败，再次从原位置开始，查找r=0且m=1的页面，把把迁到的第一个这样的页面作为淘汰页面，而在扫描过程中把指针所扫过的页面的”引用位”r置0。

时钟页面替换改进算法(3)

- 步3：如果步2失败，指针再次回到了起始位置，由于此时所有页面的”引用位”r均已为0，再转向步1操作，必要时再做步2操作，这次一定可以挑出一个可淘汰的页面。

例子--计算缺页中断次数和被淘汰页面(1)

- 假设采用固定分配策略，进程分得三个页框,执行中按下列次序引用5个独立的页面: 2 3 2 1 5 2 4 5 3 2 5 2。

例子--计算缺页中断次数和被淘汰页面(2)

- 2 3 2 1 5 2 4 5 3 2 5 2

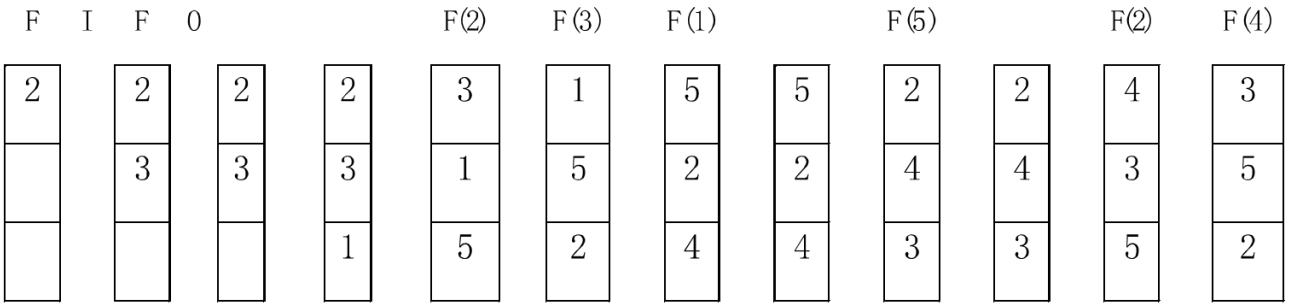
- OPT F(1) F(2) F(4)

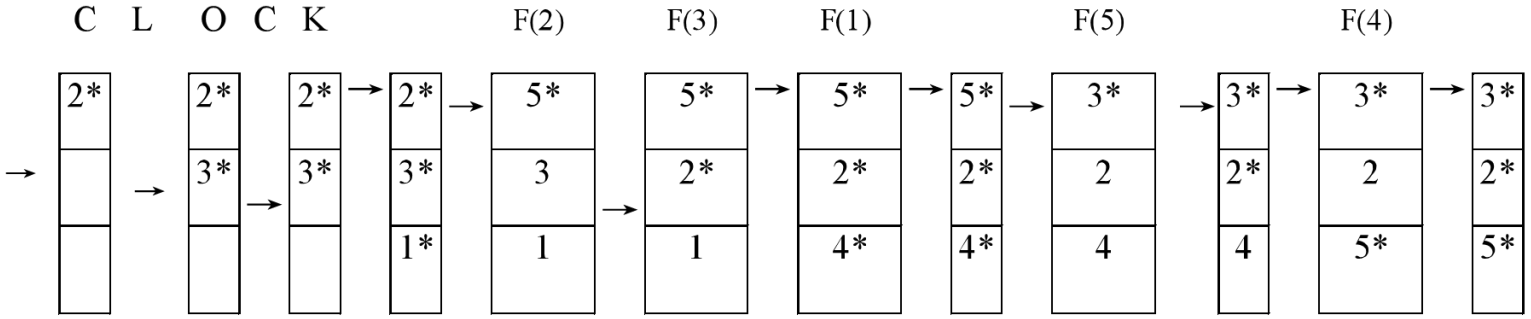
2	2	2	2	2	2	4	4	4	2	2	2	
	3	3	3	3	3	3	3	3	3	3	3	
			1	5	5	5	5	5	5	5	5	

- LRU F(3) F(1) F(2) F(4)

2	3	2	1	5	2	4	5	3	2	5	2	
	2	3	2	1	5	2	4	5	3	2	5	
			3	2	1	5	2	4	5	3	3	

例子--计算缺页中断次数和被淘汰页面(3)





例子--计算缺页中断次数和被淘汰页面(4)

性能比较

- OPT　F(1)　F(2)　F(4)　　3次
- LRU　F(3) F(1) F(2) F(4)　　3次
- CLOCK　F(2) F(3) F(1) F(5) F(4)　　3次
- FIFO　F(1) F(3) F(1) F(5) F(2) F(4)　　3次