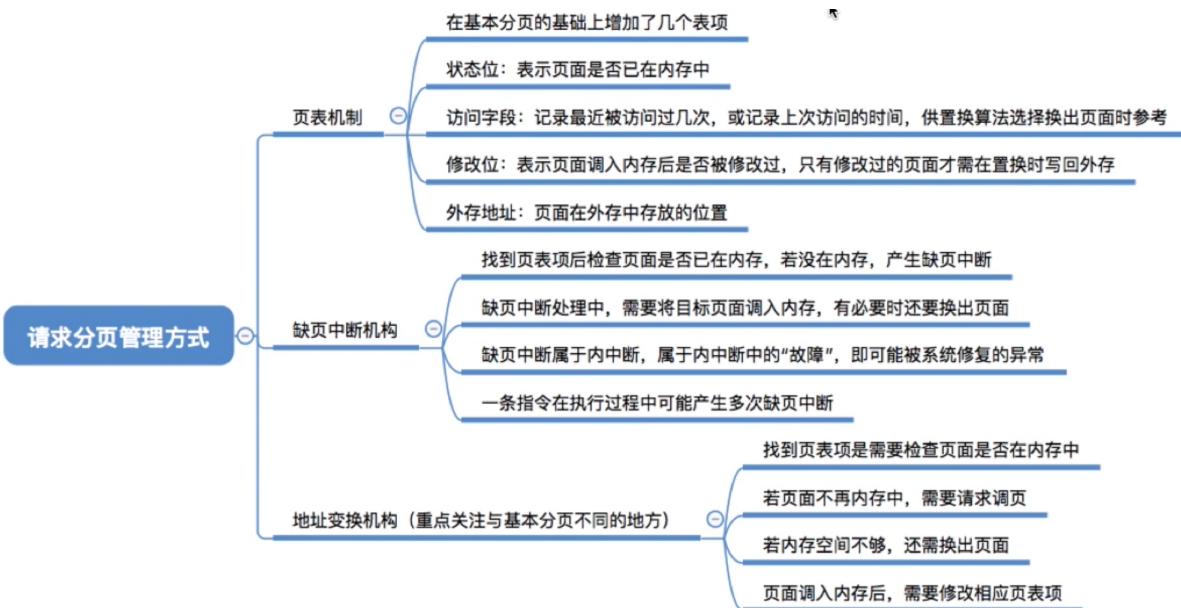


请求分页管理方式



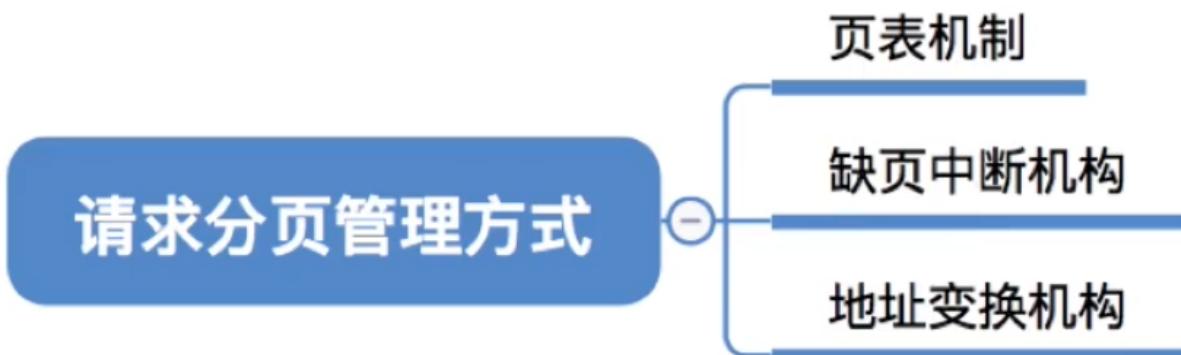
请求分页存储管理与基本分页存储管理的主要区别:

在程序执行过程中, 当所访问的信息不在内存时, 由操作系统负责将所需信息从外存调入内存, 然后继续执行程序。

若内存空间不够, 由操作系统负责将内存中暂时用不到的信息换出到外存。

操作系统要提供请求调页功能, 将缺失页面从外存调入内存

操作系统要提供页面置换的功能, 将暂时用不到的页面换出外存



注意与基本分页存储管理的页表机制、地址变换流程对比学习

页表机制

与基本分页管理相比, 请求分页管理中, 为了实现“请求调页”, 操作系统需要知道每个页面是否已经调入内存; 如果还没调入, 那么也需要知道该页面在外存中存放的位置。

当内存空间不够时, 要实现“页面置换”, 操作系统需要通过某些指标来决定到底换出哪个页面; 有的页面没有被修改过, 就不用再浪费时间写回外存。有的页面修改过, 就需要将外存中的旧数据覆盖, 因此, 操作系统也需要记录各个页面是否被修改的信息。

The diagram illustrates the evolution of a page table. On the left, a 'basic segmented storage management page table' is shown with three entries (0, 1, 2) mapping to memory blocks (a, b, c). A green callout notes that 'request page table items added four fields'. On the right, a 'request demand-paged storage management page table' is shown with five columns: Page Number, Memory Block Number, Status Bit, Access Field, Modify Bit, and External Address. The first entry (Page 0) is marked as 'not present' (无). A brown callout notes that it can record the number of recent accesses or the last access time for replacement algorithm selection. Another brown callout notes that it records whether the page has been modified after being loaded into memory. A final brown callout notes the external address where the page is stored.

页号	内存块号
0	a
1	b
2	c

基本分页存储管理的页表

页号	内存块号	状态位	访问字段	修改位	外存地址
0	无	0	0	0	x
1	b	1	10	0	y
2	c	1	6	1	z

请求分页存储管理的页表

缺页中断机构

页号	内存块号	状态位	访问字段	修改位	外存地址
0	无	0	0	0	x
1	b	1	10	0	y
2	c	1	6	1	z

假设此时要访问逻辑地址=(页号, 页内偏移量)=(0, 1024)

在请求分页系统中，每当要访问的页面不在内存时，便产生一个缺页中断，然后由操作系统的缺页中断处理程序处理中断。

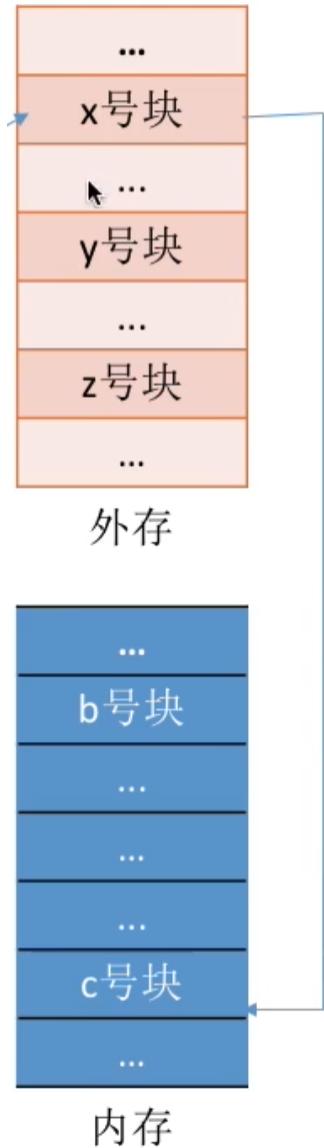
此时缺页的进程阻塞，放入阻塞队列，调页完成后再将其唤醒，放回就绪队列。

如果内存中有空闲块，则为进程分配一个空闲块，将所缺页面装入该块，并修改页表中相应的页表项。

页号	内存块号	状态位	访问字段	修改位	外存地址
0	a	1	0	0	x
1	b	1	10	0	y
2	c	1	6	1	z



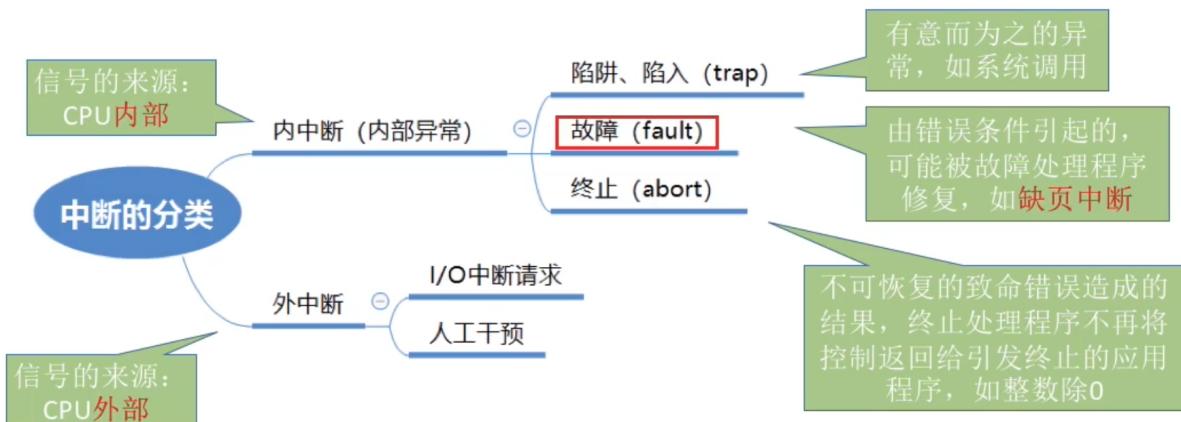
如果内存中没有空闲块，则由页面置换算法选择一个页面淘汰，若该页面在内存期间被修改过，则要将其写回外存。未修改过的页面不用写回外存。



页号	内存块号	状态位	访问字段	修改位	外存地址
0	c	1	0	0	x
1	b	1	10	0	y
2	无	0	0	0	z

缺页中断是因为当前执行的指令想要访问的目标页面未调入内存而产生的，因此属于内中断

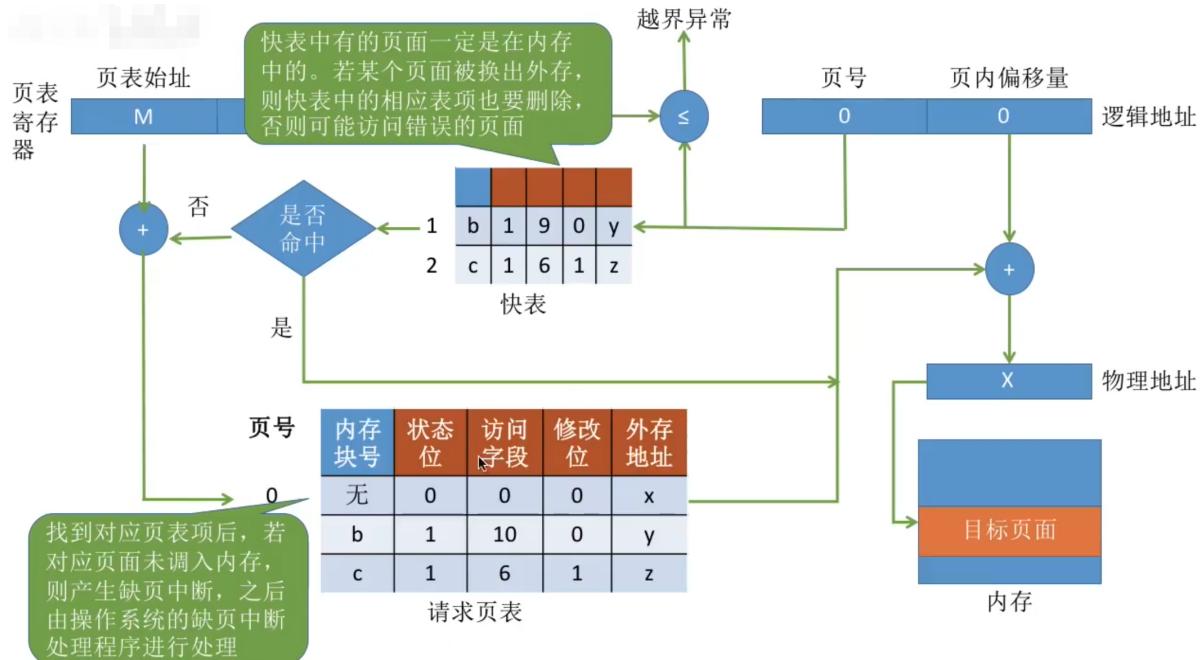
一条指令在执行期间，可能产生多次缺页中断。（如：copy A to B，即将逻辑地址A中的数据复制到逻辑地址B，而A、B属于不同的页面，则有可能产生两次中断）



地址变换机构

新增步骤

1. 请求调页（查找页表项时进行判断）
2. 页面置换（需要调入页面，但没有空闲内存块时进行）
3. 需要修改请求页表中新增的表项



补充细节：

1. 只有写指令才需要修改“修改位”。并且，一般来说只需要修改快表中的数据，只有要将快表项删除时才需要写回内存中的慢表。这样可以减少访存次数。
2. 和普通的中断处理一样，缺页中断处理依然需要保留CPU现场。
3. 需要用某种页面置换算法来决定一个换出页面
4. 换入/换出页面都需要启动慢速的I/O操作，可见，如果换入/换出太频繁，会有很大的开销。
5. 页面调入内存后，需要修改慢表，同时也需要将表项复制到快表中。

在具有快表机构的请求分页系统中，访问一个逻辑地址时，若发生缺页，则地址变换步骤是：

查快表（未命中）——查慢表（发现未调入内存）——调页（调入的页面对应的表项会直接加入快表）——查快表（命中）——访问目标内存单元