3.4.5 时钟页面置换算法

尽管第二次机会算法是一个比较合理的算法，但它经常要在链表中移动页面，既降低了效率又不是很有必要。一个更好的办法是把所有的页面都保存在一个类似钟面的环形链表中，一个表针指向最老的页面，如图3-16所示。

当发生缺页中断时，算法首先检查表针指向的页面，如果它的R位是0就淘汰该页面，并把新的页面插入这个位置，然后把表针前移一个位置；如果R位是1就清除R位并把表针前移一个位置，重复这个过程直到找到了一个R位为0的页面为止。了解了这个算法的工作方式，就明白为什么它被称为时钟（clock）算法了。

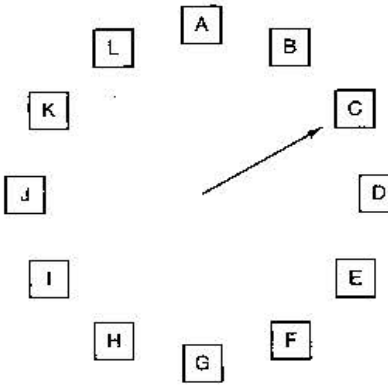

图3-16 时钟页面置换算法

3.4.6 最近最少使用页面置换算法

对最优算法的一个很好的近似是基于这样的观察：在前面几条指令中频繁使用的页面很可能在后面的几条指令中被使用。反过来说，已经很久没有使用的页面很有可能在未来较长的一段时间内仍然不会被使用。这个思想提示了一个可实现的算法：在缺页中断发生时，置换未使用时间最长的页面。这个策略称为LRU（Least Recently Used，最近最少使用）页面置换算法。

虽然LRU在理论上是可以实现的，但代价很高。为了完全实现LRU，需要在内存中维护一个所有页面的链表，最近最多使用的页面在表头，最近最少使用的页面在表尾。困难的是在每次访问内存时都必须要更新整个链表。在链表中找到一个页面，删除它，然后把它移动到表头是一个非常费时的操作，即使使用硬件实现也一样费时（假设有这样的硬件）。

然而，还是有一些使用特殊硬件实现LRU的方法。我们先考虑一个最简单的方法。这个方法要求硬件有一个64位计数器C，它在每条指令执行完后自动加1，每个页表项必须有一个足够容纳这个计数器值的域。在每次访问内存后，将当前的C值保存到被访问页面的页表项中。一旦发生缺页中断，操作系统就检查所有页表项中计数器的值，找到值最小的一个页面，这个页面就是最近最少使用的页面。

现在让我们看一看第二个硬件实现的LRU算法。在一个有n个页框的机器中，LRU硬件可以维持一个初值为0的$n \times n$位的矩阵。当访问到页框k时，硬件首先把k行的位都设置成1，再把k列的位都设置成0。在任何时刻，二进制数值最小的行就是最近最少使用的，第二小的行是下一个最近最少使用的，以此类推。这个算法的工作过程可以用图3-17所示的实例说明，该实例中有4个页框，页面访问次序为：

0 1 2 3 2 1 0 3 2 3

访问页面0后的状态如图3-17a所示，访问页1后的状态如图3-17b所示，以此类推。

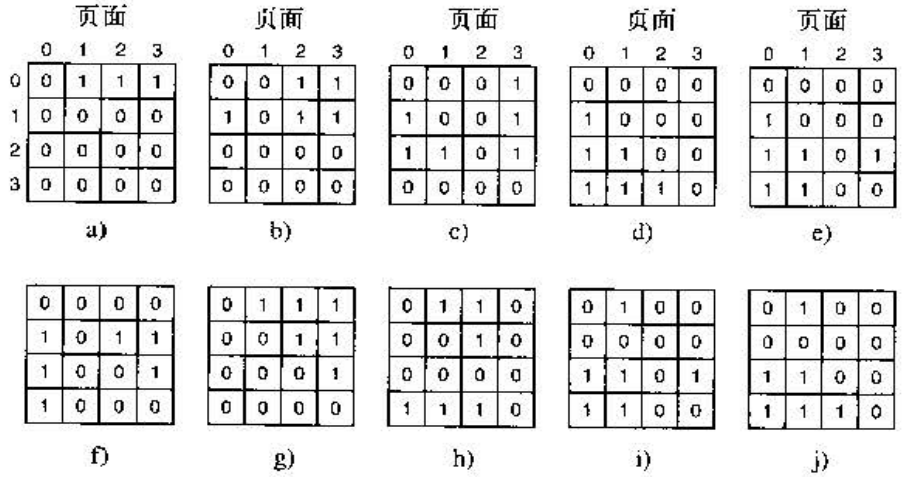

图3-17 使用矩阵的LRU，页面以0、1、2、3、2、1、0、3、2、3次序访问