3.4.7 用软件模拟LRU

前面两种LRU算法虽然在理论上都是可以实现的，但只有非常少的计算机拥有这种硬件。因此，需要一个能用软件实现的解决方案。一种可能的方案称为NFU（Not Frequently Used，最不常用）算法。该算法将每个页面与一个软件计数器相关联，计数器的初值为0。每次时钟中断时，由操作系统扫描内存中所有的页面，将每个页面的R位（它的值是0或1）加到它的计数器上。这个计数器大体上跟踪了各个页面被访问的频繁程度。发生缺页中断时，则置换计数器值最小的页面。

NFU的主要问题是它从来不忘记任何事情。比如，在一个多次（扫描）编译器中，在第一次扫描中被频繁使用的页面在程序进入第二次扫描时，其计数器的值可能仍然很高。实际上，如果第一次扫描的执行时间恰好是各次扫描中最长的，含有以后各次扫描代码的页面的计数器可能总是比含有第一次扫描代码的页面小，结果是操作系统将置换有用的页面而不是不再使用的页面。

幸运的是只需对NFU做一个小小的修改就能使它很好地模拟LRU。其修改分为两部分：首先，在R位被加进之前先将计数器右移一位；其次，将R位加到计数器最左端的位而不是最右端的位。

修改以后的算法称为老化（aging）算法，图3-18解释了它是如何工作的。假设在第一个时钟滴答后，页面0到页面5的R位值分别是1、0、1、0、1、1（页面0为1，页面1为0，页面2为1，以此类推）。换句话说，在时钟滴答0到时钟滴答1期间，访问了页0、2、4、5，它们的R位设置为1，而其他页面的R位仍然是0。对应的6个计数器在经过移位并把R位插入其左端后的值如图3-18a所示。图中后面的4列是在下4个时钟滴答后的6个计数器的值。

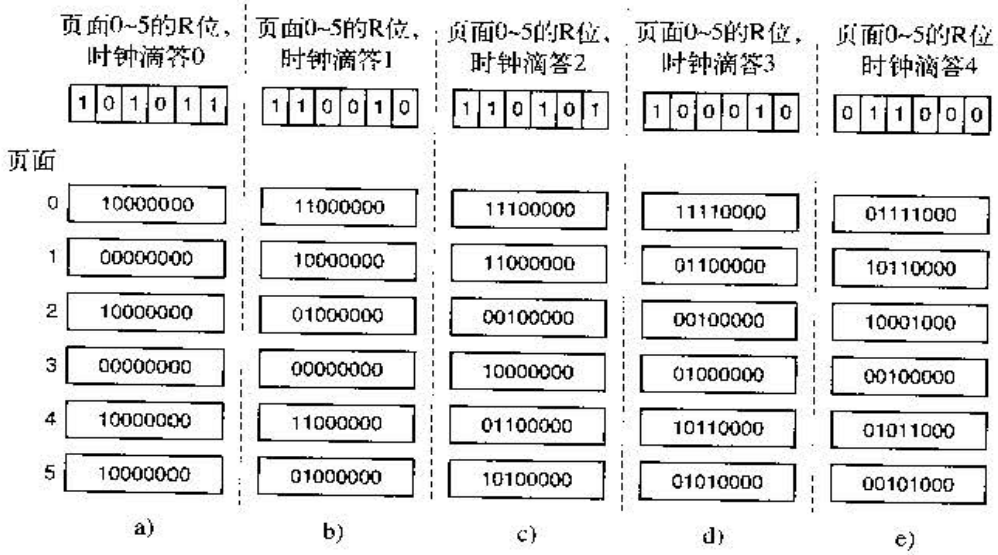

图3-18 用软件模拟LRU的老化算法。图中所示是6个页面在5个时钟滴答的情况，5个时钟滴答分别由a~e表示

发生缺页中断时，将置换计数器值最小的页面。如果一个页面在前面4个时钟滴答中都没有访问过，那么它的计数器最前面应该有4个连续的0，因此它的值肯定要比在前面三个时钟滴答中都没有被访问过的页面的计数器值小。

该算法与LRU有两个区别。如图3-18e中的页面3和页面5，它们都连续两个时钟滴答没有被访问过了，而在两个时钟滴答之前的时钟滴答中它们都被访问过。根据LRU，如果必须置换一个页面，则应该在这两个页面中选择一个。然而现在的问题是，我们不知道在时钟滴答1到时钟滴答2期间它们中的哪一个页面是后被访问到的。因为在每个时钟滴答中只记录了一位，所以无法区分在一个时钟滴答中哪个页面在较早的时间被访问以及哪个页面在较晚的时间被访问，因此，我们所能做的就是置换页面3，原因是页面5在更往前的两个时钟滴答中也被访问过而页面3没有。

LRU和老化算法的第二个区别是老化算法的计数器只有有限位数（本例中是8位），这就限制了其对以往页面的记录。如果两个页面的计数器都是0，我们只能在两个页面中随机选一个进行置换。实际上，有可能其中一个页面上次被访问是在9个时钟滴答以前，另一个页面是在1000个时钟滴答以前，而我们