制到内存中的页表项，而除了访问位，其他的值不变。当页表项中从页表装入到TLB中时，所有的值都来自内存。

2. 软件TLB管理

到目前为止，我们已经假设每一台具有虚拟内存的机器都具有由硬件识别的页表，以及一个TLB。在这种设计中，对TLB的管理和TLB的失效处理都完全由MMU硬件来实现。只有在内存中没有找到某个页面时，才会陷入到操作系统中。

在过去，这样的假设是正确的。但是，许多现代的RISC机器，包括SPARC、MIPS以及HP PA，几乎所有的页面管理都是在软件中实现的。在这些机器上，TLB表项被操作系统显式地装载。当发生TLB访问失效，不再是由MMU到页表中查找并取出需要的页表项，而是生成一个TLB失效并将问题交给操作系统解决。系统必须先找到该页面，然后从TLB中删除一个项，接着装载一个新的项，最后再执行先前出错的指令。当然，所有这一切都必须在有限的几条指令中完成，因为TLB失效比缺页中断发生的更加频繁。

让人感到惊奇的是，如果TLB大（如64个表项）到可以减少失效率时，TLB的软件管理就会变得足够有效。这种方法的最主要的好处是获得了一个非常简单的MMU，这就在CPU芯片上为高速缓存以及其他改善性能的设计腾出了相当大的空间。Uhlig等人在论文（Uhlig，1994）中讨论过软件TLB管理。

到目前为止，已经开发了多种不同的策略来改善使用软件TLB管理的机器的性能。其中一种策略是在减少TLB失效的同时，又要在发生TLB失效时减少处理开销（Bala 等人，1994）。为了减少TLB失效，有时候操作系统能用“直觉”指出哪些页面下一步可能会被用到并预先为它们在TLB中装载表项。例如，当一个客户进程发送一条消息给同一台机器上的服务器进程，很可能服务器将不得不立即运行。了解了这一点，当执行处理send的陷阱时，系统也可以找到服务器的代码页、数据页以及堆栈页，并在有可能导致TLB失效前把它们装载到TLB中。

无论是用硬件还是用软件来处理TLB失效，常见方法都是找到页表并执行索引操作以定位将要访问的页面。用软件做这样的搜索的问题是，页表可能不在TLB中，这就会导致处理过程中的额外的TLB失效。可以通过在内存中的固定位置维护一个大的（如4KB）TLB表项的软件高速缓存（该高速缓存的页面总是被保存在TLB中）来减少TLB失效。通过首先检查软件高速缓存，操作系统能够实质性地减少TLB失效。

当使用软件TLB管理时，一个基本要求是要理解两种不同的TLB失效的区别在哪里。当一个页面访问在内存中而不在TLB中时，将产生软失效（soft miss）。那么此时所要做的就是更新一下TLB，不需要产生磁盘I/O。典型的处理需要10~20个机器指令并花费几个纳秒完成操作。相反，当页面本身不在内存中（当然也不在TLB中）时，将产生硬失效。此刻需要一次磁盘存取以装入该页面，这个过程大概需要几毫秒。硬失效的处理时间往往是软失效的百万倍。

3.3.4 针对大内存的页表

在原有的内存页表的方案之上，引入快表（TLB）可以用来加快虚拟地址到物理地址的转换。不过这不是惟一需要解决的问题，另一个问题是怎样处理巨大的虚拟地址空间。下面将讨论两种解决方法。

1. 多级页表

第一种方法是采用多级页表。一个简单的例子如图3-13所示。在图3-13a中，32位的虚拟地址被划分为10位的PT1域、10位的PT2域和12位的Offset（偏移量）域。因为偏移量是12位，所以页面长度是4KB，共有2^{20}个页面。

引入多级页表的原因是避免把全部页表一直保存在内存中。特别是那些从不需要的页表就不应该保留。比如一个需要12MB内存的进程，其最底端是4MB的程序正文段，后面是4MB的数据段，顶端是4MB的堆栈段，在数据段上方和堆栈段下方之间是大量根本没有使用的空闲区。

考察图3-13b例子中的二级页表是如何工作的。在左边是顶级页表，它具有1024个表项，对应于10位的PT1域。当一个虚拟地址被送到MMU时，MMU首先提取PT1域并把该值作为访问顶级页表的索引。因为整个4GB（32位）虚拟地址空间已经被分成1024个4MB的块，所以这1024个表项中的每一个都表示4MB的虚拟地址空间。