位部分）和偏移量（低位部分）两部分。例如，对于16位地址和4KB 的页面大小，高4位可以指定16个虚拟页面中的一页，而低12位接着确定了所选页面中的字节偏移量（0~4095）。但是使用3或者5或者其他位数拆分虚拟地址也是可行的。不同的划分对应不同的页面大小。

虚拟页号可用做页表的索引，以找到该虚拟页面对应的页表项。由页表项可以找到页框号（如果有的话）。然后把页框号拼接到偏移量的高位端，以替换掉虚拟页号，形成送往内存的物理地址。

页表的目的是把虚拟页面映射为页框。从数学角度说，页表是一个函数，它的参数是虚拟页号，结果是物理页框号。通过这个函数可以把虚拟地址中的虚拟页面域替换成页框域，从而形成物理地址。

页表项的结构

下面将讨论单个页表项的细节。页表项的结构是与机器密切相关的，但不同机器的页表项存储的信息都大致相同。图3-11中给出了页表项的一个例子。不同计算机的页表项大小可能不一样，但32位是一个常用的大小。最重要的域是页框号。毕竟页映射的目的是找到这个值，其次是“在/不在”位，这一位是1时表示该表项是有效的，可以使用；如果是0，则表示该表项对应的虚拟页面现在不在内存中，访问该页面会引起一个缺页中断。

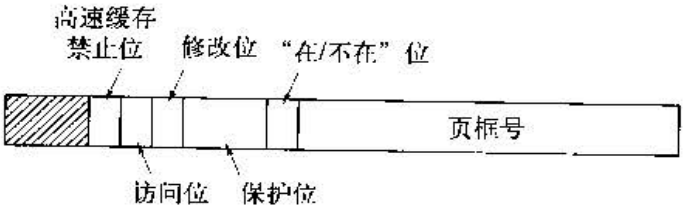

图3-11 一个典型的页表项

“保护”（protection）位指出一个页允许什么类型的访问。最简单的形式是这个域只有一位，0表示读/写，1表示只读。一个更先进的方法是使用三位，各位分别对应是否启用读、写、执行该页面。

为了记录页面的使用状况，引入了“修改”（modified）位和“访问”（referenced）位。在写入一页时由硬件自动设置修改位。该位在操作系统重新分配页框时是非常有用的。如果一个页面已经被修改过（即它是“脏”的），则必须把它写回磁盘。如果一个页面没有被修改过（即它是“干净”的），则只简单地把它丢弃就可以了，因为它在磁盘上的副本仍然是有效的。这一位有时也被称为脏位（dirty bit），因为它反映了该页面的状态。

不论是读还是写，系统都会在该页面被访问时设置访问位。它的值被用来帮助操作系统在发生缺页中断时选择要被淘汰的页面。不再使用的页面要比正在使用的页面更适合淘汰。这一位在即将讨论的很多页面置换算法中都会起到重要的作用。

最后一位用于禁止该页面被高速缓存。对那些映射到设备寄存器而不是常规内存的页面而言，这个特性是非常重要的。假如操作系统正在紧张地循环等待某个I/O设备对它刚发出的命令作出响应，保证硬件是不断地从设备中读取数据而不是访问一个旧的被高速缓存的副本是非常重要的。通过这一位可以禁止高速缓存。具有独立的I/O空间而不使用内存映射I/O的机器不需要这一位。

应该注意的是，若某个页面不在内存时，用于保存该页面的磁盘地址不是页表的一部分。原因很简单，页表只保存把虚拟地址转换为物理地址时硬件所需要的信息。操作系统在处理缺页中断时需要把该页面的磁盘地址等信息保存在操作系统内部的软件表格中。硬件不需要它。

在深入到更多应用实现问题之前，值得再次强调的是：虚拟内存本质上是用来创造一个新的抽象概念——地址空间，这个概念是对物理内存的抽象，类似于进程是对物理机器（CPU）的抽象。虚拟内存的实现，是将虚拟地址空间分解成页，并将每一页映射到物理内存的某个页框或者（暂时）解除映射。因此，本章的基本内容即关于操作系统创建的抽象，以及如何管理这个抽象。

3.3.3 加速分页过程

我们已经了解了虚拟内存和分页的基础。现在是时候深入到更多关于可能的实现的细节中去了。在任何分页式系统中，都需要考虑两个主要问题：

1) 虚拟地址到物理地址的映射必须非常快。

2) 如果虚拟地址空间很大，页表也会很大。

第一个问题是由于每次访问内存，都需要进行虚拟地址到物理地址的映射。所有的指令最终都必须来自内存，并且很多指令也会访问内存中的操作数。因此，每条指令进行一两次或更多页表访问是必要