所示。为了和只有24位基址的286兼容，基址被分为3片分布在描述符的各个位置。实际上，基址允许每个段的起始地址位于32位线性地址空间内的任何位置。

如果禁止分页（通过全局控制寄存器中的一位），线性地址就被解释为物理地址并被送往存储器用于读写操作。因此在禁止分页时，我们就得到了一个纯的分段方案。各个段的基址在它的描述符中。另外，段之间允许互相覆盖，这可能是因为验证所有的段都互不重叠太麻烦太费时间的缘故。

另一方面，如果允许分页，线性地址将通过页表映射到物理地址，很像我们前面讲过的例子。这里惟一真正复杂的是在32位虚拟地址和4KB页的情况下，一个段可能包含多达100万个页面，因此使用了一种两级映射，以便在段较小时减小页表大小。

每个运行程序都有一个由1024个32位表项组成的页目录（page directory）。它通过一个全局寄存器来定位。这个目录中的每个目录项都指向一个也包含1024个32位表项的页表，页表项指向页框，这个方案如图3-42所示。

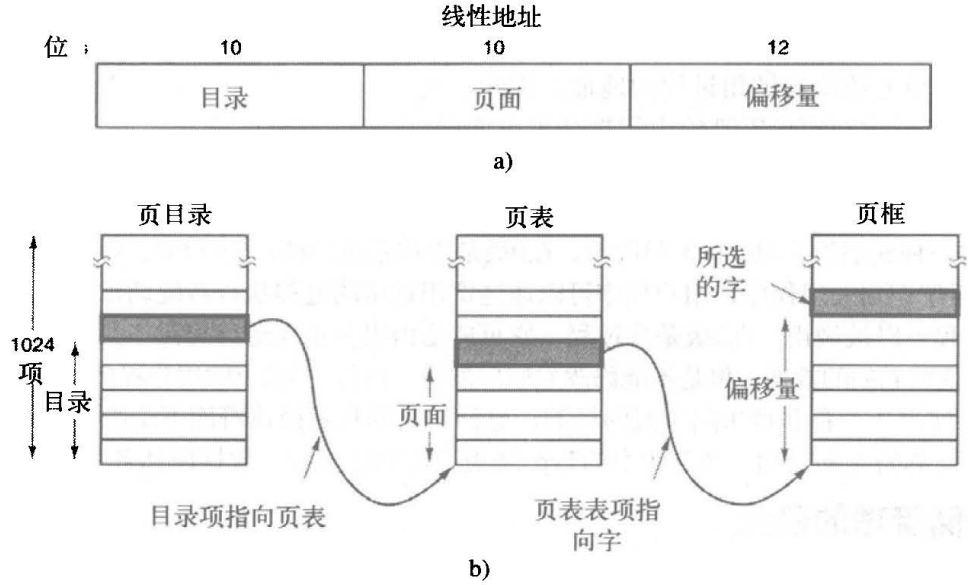

图3-42 线性地址到物理地址的映射

在图3-42a中我们看到线性地址被分为三个域：目录、页面和偏移量。目录域被作为索引在页目录中找到指向正确的页表的指针，随后页面域被用作索引在页表中找到页框的物理地址，最后，偏移量被加到页框的地址上得到需要的字节或字的物理地址。

每个页表项是32位，其中20位是页框号。其余的位包含了由硬件设置供操作系统使用的访问位和“脏”位、保护位和一些其他有用的位。

每个页表有描述1024个4KB页框的表项，因此一个页表可以处理4MB的内存。一个小于4MB的段的页目录中将只有一个表项，这个表项指向一个惟一的页表。通过这种方法，长度短的段的开销只是两个页面，而不是一级页表时的100万个页面。

为了避免重复的内存访问，Pentium处理器和MULTICS一样，也有一个小的TLB把最近使用过的“目录-页面”二元组映射为页框的物理地址。只有在当前组合不在TLB中时，图3-42所示的机制才被真正执行并更新TLB。只要TLB的缺失率很低，则性能就不错。

还有一点值得注意，如果某些应用程序不需要分段，而是需要一个单独的、分页的32位地址空间，这样的模式是可以做到的。这时，所有的段寄存器可以用同一个选择子设置，其描述符中基址设为0，段长度被设置为最大。指令偏移量会是线性地址，只使用了一个地址空间——效果上就是正常的分页。事实上，所有当前的Pentium操作系统都是这样工作的。OS/2是惟一一个使用Intel MMU体系结构所有功能的操作系统。

不管怎么说，我们不得不称赞Pentium处理器的设计者，因为他们面对的是互相冲突的目标，实现纯的分页、纯的分段和段页式管理，同时还要与286兼容，而他们高效地实现了所有的目标，最终的设