不被用户程序访问。XD(禁止执行)位是在 64 位系统中引入的，可以用来禁止从某些内存页取指令。这是一个重要的新特性，通过限制只能执行只读代码段，使得操作系统内核降低了缓冲区溢出攻击的风险。

当 MMU 翻译每一个虚拟地址时，它还会更新另外两个内核缺页处理程序会用到的位。每次访问一个页时，MMU 都会设置 A 位，称为*引用位*(reference bit)。内核可以用这个引用位来实现它的页替换算法。每次对一个页进行了写之后，MMU 都会设置 D 位，又称*修改位*或*脏位*(dirty bit)。修改位告诉内核在复制替换页之前是否必须写回牺牲页。内核可以通过调用一条特殊的内核模式指令来清除引用位或修改位。

图 9-25 给出了 Core i7 MMU 如何使用四级的页表来将虚拟地址翻译成物理地址。36 位 VPN 被划分成四个 9 位的片，每个片被用作到一个页表的偏移量。CR3 寄存器包含 L1 页表的物理地址。VPN 1 提供到一个 L1 PET 的偏移量，这个 PTE 包含 L2 页表的基地址。VPN 2 提供到一个 L2 PTE 的偏移量，以此类推。

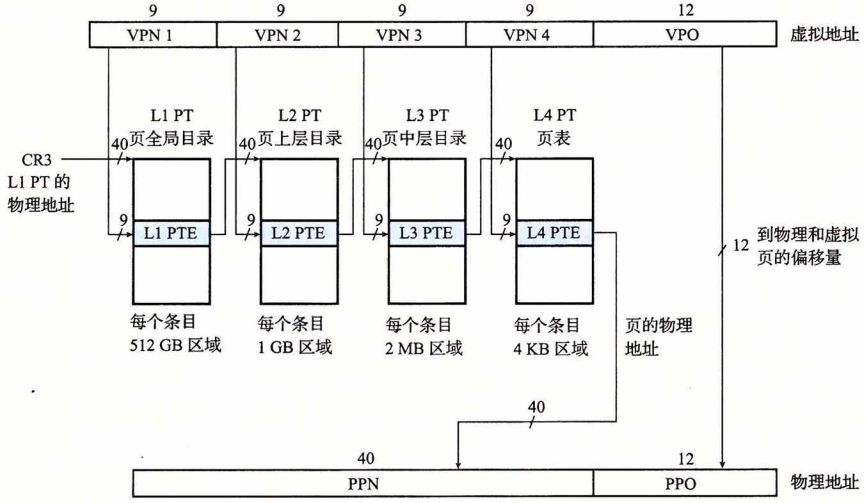

图 9-25　Core i7 页表翻译(PT：页表，PTE：页表条目，VPN：虚拟页号，VPO：虚拟页偏移，PPN：物理页号，PPO：物理页偏移量。图中还给出了这四级页表的 Linux 名字)

旁注　优化地址翻译

在对地址翻译的讨论中，我们描述了一个顺序的两个步骤的过程，1)MMU 将虚拟地址翻译成物理地址，2)将物理地址传送到 L1 高速缓存。然而，实际的硬件实现使用了一个灵活的技巧，允许这些步骤部分重叠，因此也就加速了对 L1 高速缓存的访问。例如，页面大小为 4KB 的 Core i7 系统上的一个虚拟地址有 12 位的 VPO，并且这些位和相应物理地址中的 PPO 的 12 位是相同的。因为八路组相联的、物理寻址的 L1 高速缓存有 64 个组和大小为 64 字节的缓存块，每个物理地址有 6 个($\log_2 64$)缓存偏移位和 6 个($\log_2 64$)索引位。这 12 位恰好符合虚拟地址的 VPO 部分，这绝不是偶然！当 CPU 需要翻译一个虚拟地址时，它就发送 VPN 到 MMU，发送 VPO 到高速 L1 缓存。当 MMU