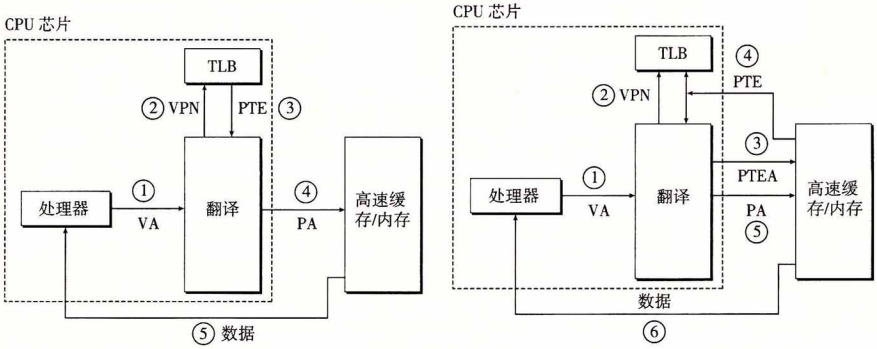
虚拟地址中的虚拟页号中提取出来的。如果 TLB 有 $T=2^t$ 个组，那么 TLB 索引(TLBI)是由 VPN 的 t 个最低位组成的，而 TLB 标记(TLBT)是由 VPN 中剩余的位组成的。

图 9-16a 展示了当 TLB 命中时(通常情况)所包括的步骤。这里的关键点是，所有的地址翻译步骤都是在芯片上的 MMU 中执行的，因此非常快。

- 第 1 步：CPU 产生一个虚拟地址。
- 第 2 步和第 3 步：MMU 从 TLB 中取出相应的 PTE。
- 第 4 步：MMU 将这个虚拟地址翻译成一个物理地址，并且将它发送到高速缓存/主存。
- 第 5 步：高速缓存/主存将所请求的数据字返回给 CPU。

当 TLB 不命中时，MMU 必须从 L1 缓存中取出相应的 PTE，如图 9-16b 所示。新取出的 PTE 存放在 TLB 中，可能会覆盖一个已经存在的条目。



a) TLB 命中　　b) TLB 不命中

图 9-16　TLB 命中和不命中的操作图

9.6.3　多级页表

到目前为止，我们一直假设系统只用一个单独的页表来进行地址翻译。但是如果我们有一个 32 位的地址空间、4KB 的页面和一个 4 字节的 PTE，那么即使应用所引用的只是虚拟地址空间中很小的一部分，也总是需要一个 4MB 的页表驻留在内存中。对于地址空间为 64 位的系统来说，问题将变得更复杂。

用来压缩页表的常用方法是使用层次结构的页表。用一个具体的示例是最容易理解这个思想的。假设 32 位虚拟地址空间被分为 4KB 的页，而每个页表条目都是 4 字节。还假设在这一时刻，虚拟地址空间有如下形式：内存的前 2K 个页面分配给了代码和数据，接下来的 6K 个页面还未分配，再接下来的 1023 个页面也未分配，接下来的 1 个页面分配给了用户栈。图 9-17 展示了我们如何为这个虚拟地址空间构造一个两级的页表层次结构。

一级页表中的每个 PTE 负责映射虚拟地址空间中一个 4MB 的片(chunk)，这里每一片都是由 1024 个连续的页面组成的。比如，PTE 0 映射第一片，PTE 1 映射接下来的一片，以此类推。假设地址空间是 4GB，1024 个 PTE 已经足够覆盖整个空间了。

如果片 i 中的每个页面都未被分配，那么一级 PTE i 就为空。例如，图 9-17 中，片 2～7 是未被分配的。然而，如果在片 i 中至少有一个页是分配了的，那么一级 PTE i 就指向一个二级页表的基址。例如，在图 9-17 中，片 0、1 和 8 的所有或者部分已被分配，所以它们的一级 PTE 就指向二级页表。