

开始这种手工的模拟,我们发现写下虚拟地址的各个位,标识出我们会需要的各种字段,并确定它们的十六进制值,是非常有帮助的。当硬件解码地址时,它也执行相似的任务。

	TLBT								TLBI							
	0x03								0x03							
位位置	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
VA = 0x03d4	0	0	0	0	1	1	1	1	0	1	0	1	0	0		
	VPN								VPO							
	0x0f								0x14							

开始时,MMU从虚拟地址中抽取出VPN(0x0F),并且检查TLB,看它是否因为前面的某个内存引用缓存了PTE 0x0F的一个副本。TLB从VPN中抽取出TLB索引(0x03)和TLB标记(0x3),组0x3的第二个条目中有效匹配,所以命中,然后将缓存的PPN(0x0D)返回给MMU。


如果TLB不命中,那么MMU就需要从主存中取出相应的PTE。然而,在这种情况下,我们很幸运,TLB会命中。现在,MMU有了形成物理地址所需要的所有东西。它通过将来自PTE的PPN(0x0D)和来自虚拟地址的VPO(0x14)连接起来,这就形成了物理地址(0x354)。

接下来,MMU发送物理地址给缓存,缓存从物理地址中抽取出缓存偏移CO(0x0)、缓存组索引CI(0x5)以及缓存标记CT(0x0D)。

	CT								CI							
	0x0d								0x05							
位位置	11	10	9	8	7	6	5	4	3	2	1	0				
PA = 0x354	0	0	1	1	0	1	0	1	0	1	0	0				
	PPN								PPO							
	0x0d								0x14							

因为组0x5中的标记与CT相匹配,所以缓存检测到一个命中,读出在偏移量CO处的数据字节(0x36),并将它返回给MMU,随后MMU将它传递回CPU。

翻译过程的其他路径也是可能的。例如,如果TLB不命中,那么MMU必须从页表中的PTE中取出PPN。如果得到的PTE是无效的,那么就产生一个缺页,内核必须调入合适的页面,重新运行这条加载指令。另一种可能性是PTE是有效的,但是所需要的内存块在缓存中不命中。

 **练习题 9.4** 说明 9.6.4 节中的示例内存系统是如何将一个虚拟地址翻译成一个物理地址和访问缓存的。对于给定的虚拟地址,指明访问的TLB条目、物理地址和返回的缓存字节值。指出是否发生了TLB不命中,是否发生了缺页,以及是否发生了缓存不命中。如果是缓存不命中,在“返回的缓存字节”栏中输入“—”。如果有缺页,则在“PPN”一栏中输入“—”,并且将C部分和D部分空着。

**虚拟地址:** 0x03d7

A. 虚拟地址格式

13	12	11	10	9	8	7	6	5	4	3	2	1	0

B. 地址翻译