个示例，我们假设 CPU 读 1 字节的字。虽然这种手工的模拟很乏味，你可能想要跳过它，但是根据我们的经验，在学生们做过几个这样的练习之前，他们是不能真正理解高速缓存是如何工作的。

初始时，高速缓存是空的(即每个有效位都是 0)：

组	有效位	标记位	块[0]	块[1]
0	0			
1	0			
2	0			
3	0			

表中的每一行都代表一个高速缓存行。第一列表明该行所属的组，但是请记住提供这个位只是为了方便，实际上它并不真是高速缓存的一部分。后面四列代表每个高速缓存行的实际的位。现在，让我们来看看当 CPU 执行一系列读时，都发生了什么：

1) **读地址 0 的字。**因为组 0 的有效位是 0，是缓存不命中。高速缓存从内存(或低一层的高速缓存)取出块 0，并把这个块存储在组 0 中。然后，高速缓存返回新取出的高速缓存行的块[0]的 m[0](内存位置 0 的内容)。

组	有效位	标记位	块[0]	块[1]
0	1	0	m[0]	m[1]
1	0			
2	0			
3	0			

2) **读地址 1 的字。**这次会是高速缓存命中。高速缓存立即从高速缓存行的块[1]中返回 m[1]。高速缓存的状态没有变化。

3) **读地址 13 的字。**由于组 2 中的高速缓存行不是有效的，所以有缓存不命中。高速缓存把块 6 加载到组 2 中，然后从新的高速缓存行的块[1]中返回 m[13]。

组	有效位	标记位	块[0]	块[1]
0	1	0	m[0]	m[1]
1	0			
2	1	1	m[12]	m[13]
3	0			

4)**读地址 8 的字。**这会发生缓存不命中。组 0 中的高速缓存行确实是有效的，但是标记不匹配。高速缓存将块 4 加载到组 0 中(替换读地址 0 时读入的那一行)，然后从新的高速缓存行的块[0]中返回 m[8]。

组	有效位	标记位	块[0]	块[1]
0	1	1	m[8]	m[9]
1	0			
2	1	1	m[12]	m[13]
3	0			

5)**读地址 0 的字。**又会发生缓存不命中，因为在前面引用地址 8 时，我们刚好替换了块 0。这就是冲突不命中的一个例子，也就是我们有足够的高速缓存空间，但是却交替地引用映射到同一个组的块。