

图 6-33 组相联高速缓存中的组选择

图 6-34 展示了相联高速缓存中行匹配的基本思想。这里的一个重要思想就是组中的任何一行都可以包含任何映射到这个组的内存块。所以高速缓存必须搜索组中的每一行，寻找一个有效的行，其标记与地址中的标记相匹配。如果高速缓存找到了这样一行，那么我们就命中，块偏移从这个块中选择一个字，和前面一样。

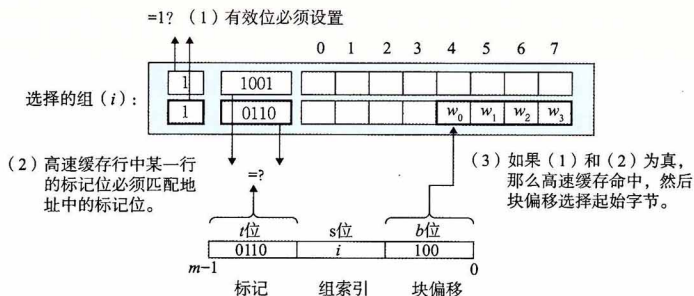


图 6-34 组相联高速缓存中的行匹配和字选择

### 3. 组相联高速缓存中不命中时的行替换

如果 CPU 请求的字不在组的任何一行中，那么就是缓存不命中，高速缓存必须从内存中取出包含这个字的块。不过，一旦高速缓存取出了这个块，该替换哪个行呢？当然，如果有一个空行，那它就是个很好的候选。但是如果该组中没有空行，那么我们必须从中选择一个非空的行，希望 CPU 不会很快引用这个被替换的行。

程序员很难在代码中利用高速缓存替换策略，所以在此我们不会过多地讲述其细节。最简单的替换策略是随机选择要替换的行。其他更复杂的策略利用了局部性原理，以使在比较近的将来引用被替换的行的概率最小。例如，最不常使用 (Least-Frequently-Used, LFU) 策略会替换在过去某个时间窗口内引用次数最少的那一行。最近最少使用 (Least-Recently-Used, LRU) 策略会替换最后一次访问时间最久远的那一行。所有这些策略都需要额外的时间和硬件。但是，越往存储器层次结构下面走，远离 CPU，一次不命中的开销就会更加昂贵，用更好的替换策略使得不命中最少也变得更加值得了。

## 6.4.4 全相联高速缓存

全相联高速缓存 (fully associative cache) 是由一个包含所有高速缓存行的组 (即  $E=C/$