- 数组 src 从地址 0 开始，而数组 dst 从地址 64 开始(十进制)。
- 只有一个 L1 数据高速缓存，它是直接映射、直写、写分配的，块大小为 16 字节。
- 这个高速缓存总共有 32 个数据字节，初始为空。
- 对 src 和 dst 数组的访问分别是读和写不命中中的唯一来源。

对于每个 row 和 col，指明对 src[row][col]和 dst[row][col]的访问是命中(h)还是不命中(m)。例如，读 src[0][0]会不命中，而写 dst[0][0]也会不命中。

dst数组

	列0	列1	列2	列3
行0	m			
行1				
行2				
行3				

src数组

	列0	列1	列2	列3
行0	m			
行1				
行2				
行3				

••6.35　对于一个总大小为 128 数据字节的高速缓存，重复练习题 6.34。

dst数组

	列0	列1	列2	列3
行0	m			
行1				
行2				
行3				

src数组

	列0	列1	列2	列3
行0	m			
行1				
行2				
行3				

••6.36　这道题测试你预测 C 语言代码的高速缓存行为的能力。对下面这段代码进行分析：

```
1       int x[2][128];
2       int i;
3       int sum = 0;
4
5       for (i = 0; i < 128; i++) {
6           sum += x[0][i] * x[1][i];
7       }
```

假设我们在下列条件下执行这段代码：

- sizeof(int)==4。
- 数组 x 从内存地址 0x0 开始，按照行优先顺序存储。
- 在下面每种情况中，高速缓存最开始时都是空的。
- 唯一的内存访问是对数组 x 的条目进行访问。其他所有的变量都存储在寄存器中。

给定这些假设，估计下列情况中的不命中率：

A. 情况 1：假设高速缓存是 512 字节，直接映射，高速缓存块大小为 16 字节。不命中率是多少？

B. 情况 2：如果我们把高速缓存的大小翻倍到 1024 字节，不命中率是多少？

C. 情况 3：现在假设高速缓存是 512 字节，两路组相联，使用 LRU 替换策略，高速缓存块大小为 16 字节。不命中率是多少？

D. 对于情况 3，更大的高速缓存大小会帮助降低不命中率吗？为什么能或者为什么不能？

E. 对于情况 3，更大的块大小会帮助降低不命中率吗？为什么能或者为什么不能？

••6.37　这道题也是测试你分析 C 语言代码的高速缓存行为的能力。假设我们在下列条件下执行图 6-47 中的 3 个求和函数：

- sizeof(int)==4。
- 机器有 4KB 直接映射的高速缓存，块大小为 16 字节。
- 在两个循环中，代码只对数组数据进行内存访问。循环索引和值 sum 都存放在寄存器中。
- 数组 a 从内存地址 0x08000000 处开始存储。

对于 $N=64$ 和 $N=60$ 两种情况，在表中填写它们大概的高速缓存不命中率。