6.17 A. 解决这个问题的关键是想象出图 6-48 中的图像。注意，每个高速缓存行只包含数组的一个行，高速缓存正好只够保存一个数组，而且对于所有的 i，src 和 dst 的行 i 映射到同一个高速缓存行。因为高速缓存不够大，不足以容纳这两个数组，所以对一个数组的引用总是驱逐出另一个数组的有用的行。例如，对 dst[0][0]写会驱逐当我们读 src[0][0]时加载进来的那一行。所以，当我们接下来读 src[0][1]时，会有一个不命中。

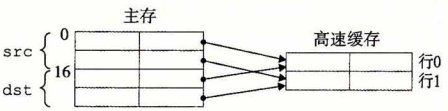

图 6-48　练习题 6.17 的图

B. 当高速缓存为 32 字节时，它足够大，能容纳这两个数组。因此，所有的不命中都是开始时的冷不命中。

dst数组

	列0	列1
行0	m	m
行1	m	m

src数组

	列0	列1
行0	m	m
行1	m	h

dst数组

	列0	列1
行0	m	h
行1	m	h

src数组

	列0	列1
行0	m	h
行1	m	h

6.18 每个 16 字节的高速缓存行包含着两个连续的 algae_position 结构。每个循环按照内存顺序访问这些结构，每次读一个整数元素。所以，每个循环的模式就是不命中、命中、不命中、命中，依此类推。注意，对于这个问题，我们不必实际列举出读和不命中的总数，就能预测出不命中率。

A. 读总数是多少？512 个读。

B. 缓存不命中的读总数是多少？256 个不命中。

C. 不命中率是多少？256/512＝50%。

6.19 对这个问题的关键是注意到这个高速缓存只能保存数组的 1/2。所以，按照列顺序来扫描数组的第二部分会驱逐扫描第一部分时加载进来的那些行。例如，读 grid[8][0]的第一个元素会驱逐当我们读 grid[0][0]的元素时加载进来的那一行。这一行也包含 grid[0][1]。所以，当我们开始扫描下一列时，对 grid[0][1]第一个元素的引用会不命中。

A. 读总数是多少？512 个读。

B. 缓存不命中的读总数是多少？256 个不命中。

C. 不命中率是多少？256/512＝50%。

D. 如果高速缓存有两倍大，那么不命中率会是多少呢？如果高速缓存有现在的两倍大，那么它能够保存整个 grid 数组。所有的不命中都会是开始时的冷不命中，而不命中率会是 1/4＝25%。

6.20 这个循环有很好的步长为 1 的引用模式，因此所有的不命中都是最开始时的冷不命中。

A. 读总数是多少？512 个读。

B. 缓存不命中的读总数是多少？128 个不命中。

C. 不命中率是多少？128/512＝25%。

D. 如果高速缓存有两倍大，那么不命中率会是多少呢？无论高速缓存的大小增加多少，都不会改变不命中率，因为冷不命中是不可避免的。

6.21 从 L1 的吞吐量峰值是大约 12 000MB/s，时钟频率是 2100MHz，而每次读访问都是以 8 字节 long 类型为单位的。所以，从这张图中我们可以估计出在这台机器上从 L1 访问一个字需要大约 2100/12 000×8＝1.4≈1.5 周期，比正常访问 L1 的延迟 4 周期快大约 2.5 倍。这是由于 4×4 的循环展开得到的并行允许同时进行多个加载操作。