6.27

对地址的分析如下（CT为标记，CI为组索引，CO为块偏移）：

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

CT CT CT CT CT CT CT CT CI CI CI CO CO

A.

组1中有效的数据：标记为45和38（0100 0101和0011 1000）

命中的地址范围为01000101001xx和00111000001xx

即0100010100100，0100010100101，0100010100110，0100010100111，0011100000100，0011100000101，0011100000110，0011100000111

转化成十六进制为0x08A4 0x08A5 0x08A6 0x08A7 0x0704 0x0705 0x0706 0x0707

B.

组6中有效的数据：标记只有91（1001 0001）

命中的地址范围为10010001110xx

即1001000111000，1001000111001，1001000111010，1001000111011

转化成十六进制为0x1238 0x1239 0x123A 0x123B

6.31

对地址的分析如下（CT为标记，CI为组索引，CO为块偏移）：

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

CT CT CT CT CT CT CT CT CI CI CI CO CO

A.地址格式：

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

0 0 1 1 1 0 0 0 1 1 0 1 0‬

B.内存引用：

参数 值

高速缓存块偏移（CO） 0x2

高速缓存组索引（CI） 0x6

高速缓存标记（CT） 0x38

高速缓存命中（是/否） 是

返回的高速缓存字节 0xEB

6.35

两个数组都能装入cache，且不会发生冲突

dst数组

|  |  |  |  |
| --- | --- | --- | --- |
| m | h | h | h |
| m | h | h | h |
| m | h | h | h |
| m | h | h | h |

src数组

|  |  |  |  |
| --- | --- | --- | --- |
| m | h | h | h |
| m | h | h | h |
| m | h | h | h |
| m | h | h | h |

6.39

2048字节，块大小为32字节，直接映射，则推出组数为64个

int是4个字节，然后point\_color结构体里面有4个int，那么一个结构体就是16字节，即一块能够装载2个结构体

16\*16的结构体数组，8行就能占满整个cache，题目中按列遍历:

A.写总数为16×16×4=1024次

B.循环体的每个第一次调用都会不命中，将数据装入cache后，后续的三次都会命中，然后继续循环，调用同一列下一行时，第一次一定会不命中，然后再接着三次命中，不命中的写总数为1/4×1024=256次

C.不命中率25％

6.43

题目中的代码是指针按照一个int的大小（即4字节）的跨度遍历，从而将整个数组覆盖范围赋值为0。而由于cache一行的大小正好是4字节，所以每一次都需要重新装入一行，即cache每一次都无法命中。故写不命中率是100％。