```
    char b;
    char a;
};

struct pixel buffer[480][640];
int i, j;
char *cptr;
int *iptr;
```

有如下假设：

- sizeof(char)==1 和 sizecf(int)==4。
- buffer 起始于内存地址 0。
- 高速缓存初始为空。
- 唯一的内存访问是对于 buffer 数组中元素的访问。变量 i、j、cptr 和 iptr 存放在寄存器中。

下面代码中百分之多少的写会在高速缓存中不命中？

```
for (j = 0; j < 640; j++) {
    for (i = 0; i < 480; i++){
        buffer[i][j].r = 0;
        buffer[i][j].g = 0;
        buffer[i][j].b = 0;
        buffer[i][j].a = 0;
    }
}
```

** 6.42　给定作业 6.41 中的假设，下面代码中百分之多少的写会在高速缓存中不命中？

```
char *cptr = (char *) buffer;
for (; cptr < (((char *) buffer) + 640 * 480 * 4); cptr++)
    *cptr = 0;
```

** 6.43　给定作业 6.41 中的假设，下面代码中百分之多少的写会在高速缓存中不命中？

```
int *iptr = (int *)buffer;
for (; iptr < ((int *)buffer + 640*480); iptr++)
    *iptr = 0;
```

** 6.44　从 CS:APP 的网站上下载 mountain 程序，在你最喜欢的 PC/Linux 系统上运行它。根据结果估计你系统上的高速缓存的大小。

** 6.45　在这项任务中，你会把在第 5 章和第 6 章中学习到的概念应用到一个内存使用频繁的代码的优化问题上。考虑一个复制并转置一个类型为 int 的 $N \times N$ 矩阵的过程。也就是，对于源矩阵 S 和目的矩阵 D，我们要将每个元素 $s_{i,j}$ 复制到 $d_{j,i}$。只用一个简单的循环就能实现这段代码：

```
void transpose(int *dst, int *src, int dim)
{
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++)
            dst[j*dim + i] = src[i*dim + j];
}
```

这里，过程的参数是指向目的矩阵(dst)和源矩阵(src)的指针，以及矩阵的大小 N(dim)。你的工作是设计一个运行得尽可能快的转置函数。

** 6.46　这是练习题 6.45 的一个有趣的变体。考虑将一个有向图 g 转换成它对应的无向图 g'。图 g' 有一条从顶点 u 到顶点 v 的边，当且仅当原图 g 中有一条 u 到 v 或者 v 到 u 的边。图 g 是由如下的它的*邻接矩阵*(adjacency matrix)G 表示的。如果 N 是 g 中顶点的数量，那么 G 是一个 $N \times N$ 的矩阵，它的元素是全 0 或者全 1。假设 g 的顶点是这样命名的：v_0，v_1，…，v_{N-1}。那么如果有一条从 v_i 到 v_j 的边，那么 G[i][j]为 1，否则为 0。注意，邻接矩阵对角线上的元素总是 1，而无向图的邻