Example 1: Pixel Scan

A bitmap image is composed of pixels. Each pixel in the image is represented as four values: three for the primary colors(red, green and blue - RGB) and one for the transparency information defined as an alpha channel. In this problem, you will compare the performance of direct mapped and 4-way associative caches for a square bitmap image initialization. Both caches have a size of 128 bytes. The direct-mapped cache has 8-byte blocks while the 4-way associative cache has 4-byte blocks. You are given the definitions:

```
typedef struct {
   unsigned char r;
   unsigned char g;
   unsigned char b;
   unsigned char a;
} pixel_t;

pixel_t pixel[16][16];
register int i, j;
```

Also assume that

- `sizeof(unsigned char) == 1`
- `pixel` begins at memory address 0
- Both caches are initially empty
- The array is stored in row-major order
- Variables `i,j` are stored in registers and any access to these variables does not cause a cache miss

(a) What fraction of the writes in the following code will result in a miss in the direct mapped cache?

```
for (i = 0; i < 16; i ++){
   for (j = 0; j < 16; j ++){
      pixel[i][j].r = 0;
      pixel[i][j].g = 0;
      pixel[i][j].b = 0;
      pixel[i][j].a = 0;
   }
}
```

Miss rate for writes to pixel: 12.5%

(b) Using code in part A, what fraction of the writes will result in a miss in the 4-way associative cache?

Miss rate for writes to pixel: 25%

(c) What fraction of the writes in the following code will result in a miss in the direct mapped cache?

```
for (i = 0; i < 16; i ++){
   for (j = 0; j < 16; j ++){
      pixel[j][i].r = 0;
      pixel[j][i].g = 0;
      pixel[j][i].b = 0;
      pixel[j][i].a = 0;
   }
}
```

Miss rate for writes to pixel: 25%

(d) Using code in part C, what fraction of the writes will result in a miss in the 4-way associative cache?

Miss rate for writes to pixel: 25%

Example 2: Cache Hits/Misses

This problem tests your understanding of conflict misses. Consider the following transpose routine:

```
typedef int array[2][2];

void transpose(array dst, array src)
{
   int i, j;
   for (i = 0; i < 2; i++)
   {
      for (j = 0; j < 2; j++)
      {
         dst[i][j] = src[j][i];
      }
   }
}
```

running on a hypothetical machine with the following properties:

- `sizeof(int) == 4`
- The `src` array starts at address `0` and the `dst` array starts at address `16` (decimal).
- There is a single L1 cache that is direct mapped and write-allocate, with a block size of 8 bytes.
- Accesses to the `src` and `dst` arrays are the only sources of read and write misses, respectively.

(a) Suppose the cache has a total size of 16 data bytes (i.e., the block size times the number of sets is 16 bytes) and that the cache is initially empty. Then for each `row` and `col`, indicate whether each access to `src[row][col]` and `dst[row][col]` is a hit (h) or a miss (m). For example, reading `src[0][0]` is a miss and writing `dst[0][0]` is also a miss, as shown below.

dst		
	col 0	**col 1**
row 0	m	h
row 1	m	m

src		
	col 0	**col 1**
row 0	m	m
row 1	m	m

(b) Repeat part A for a cache with a total size of 32 data bytes.

dst		
	col 0	**col 1**
row 0	m	h
row 1	m	h

src		
	col 0	**col 1**
row 0	m	h
row 1	m	h