CSCI 2021, Spring 2015, Homework Assignment IV

Problem 0:
Clearly label your assignment with the time of your recitation section (8:00, 9:05, 10:10, 11:15, 12:20, 1:25, 2:30). This will help us turn back your graded assignments more efficiently.

Problem 1:
Textbook Problem **5.15** on page 549.

Problem 2:
PART A
Write a C procedure that implements the inner product described in textbook problem 5.15 with the following optimizations: four-way loop unrolling, four parallel accumulators and re-association. The resulting code should be able to achieve a CPE of 2.00 for both data types on a x86-64 processor.

PART B
For each data type, list the instructions on the critical path with and without re-association.

Fetch	Interger	Floating Point
Without Re-association		
With Re-association		

Problem 3:

The following table gives the parameters for a number of different caches, where m is the number of physical address bits, C is the cache size (number of data bytes), B is the block size in bytes, and E is the number of lines per set. For each cache, determine the number of cache sets (S), tag bits (t), set index bits (s), and block offset bits (b).

Cache	m	C	B	E	S	t	s	b
1.	32	1024	4	4				
2.	32	1024	4	256				
3.	32	1024	8	1				
4.	32	1024	8	128				
5.	32	1024	32	1				
6.	32	1024	32	4				

Problem 4:

You are writing a new 3D game that you hope will earn you fame and fortune. You are currently working on a function to blank the screen buffer before drawing the next frame. The screen you are working with is a 640x480 array of pixels. The machine you are working on has a 64 KB direct mapped cache with 4 byte lines. The C structures you are using are:

```
struct pixel {
    char r;
    char g;
    char b;
    char a;
};

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

Assume:

- `sizeof(char) = 1`
- `sizeof(int) = 4`
- `buffer` begins at memory address 0
- The cache is initially empty.
- The only memory accesses are to the entries of the array `buffer`. Variables `i`, `j`, `cptr`, and `iptr` are stored in registers.

A. What percentage of the writes in the following code will miss in the cache?

```
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;
    }
}
```

Miss rate for writes to `buffer`: _____ %

B. What percentage of the writes in the following code will miss in the cache?

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

Miss rate for writes to `buffer`: _____ %

C. What percentage of the writes in the following code will miss in the cache?

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

Miss rate for writes to `buffer`: _____ %

D. Which code (A, B, or C) should be the fastest? _____

Problem 5:

Consider a direct mapped cache of size 64K with block size of 16 bytes. Furthermore, the cache is write-back and write-allocate. You will calculate the miss rate for the following code using this cache. Remember that `sizeof(int) == 4`. Assume that the cache starts empty and that local variables and computations take place completely within the registers and do not spill onto the stack.

A. Now consider the following code to copy one matrix to another. Assume that the `src` matrix starts at address 0 and that the `dest` matrix follows immediately follows it.

```
void copy_matrix(int dest[ROWS][COLS], int src[ROWS][COLS])
{
    int i, j;

    for (i=0; i<ROWS; i++) {
        for (j=0; j<COLS; j++) {
            dest[i][j] = src[i][j];
        }
    }
}
```

1. What is the cache miss rate if `ROWS = 128` and `COLS = 128`?
   Miss rate = _____%
2. What is the cache miss rate if `ROWS = 128` and `COLS = 192`?
   Miss rate = _____%
3. What is the cache miss rate if `ROWS = 128` and `COLS = 256`?
   Miss rate = _____%

B. Now consider the following two implementations of a horizontal flip and copy of the matrix. Again assume that the `src` matrix starts at address 0 and that the `dest` matrix immediately follows it.

```
void copy_n_flip_matrix1(int dest[ROWS][COLS], int src[ROWS][COLS])
{
    int i, j;

    for (i=0; i<ROWS; i++) {
        for (j=0; j<COLS; j++) {
            dest[i][COLS - 1 - j] = src[i][j];
        }
    }
}
```

1. What is the cache miss rate if `ROWS = 128` and `COLS = 128`?
   Miss rate = _____%
2. What is the cache miss rate if `ROWS = 128` and `COLS = 192`?
   Miss rate = _____%

```
void copy_n_flip_matrix2(int dest[ROWS][COLS], int src[ROWS][COLS])
{
    int i, j;

    for (j=0; j<COLS; j++) {
        for (i=0; i<ROWS; i++) {
            dest[i][COLS - 1 - j] = src[i][j];
        }
    }
}
```

1. What is the cache miss rate if `ROWS = 128` and `COLS = 128`?
   Miss rate = _____%
2. What is the cache miss rate if `ROWS = 192` and `COLS = 128`?
   Miss rate = _____%

Problem 6:

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 following 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:_____%

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: _____ %

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:_____%

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:_____%

Problem 7:

For a 2-way set associative cache, with 32-byte cache blocks, study the cache miss rate for the following loops, assuming:

1. The cache initially is empty;
2. Integers are 4 bytes long and two-dimensional arrays are stored in row-major order.

```
int array[1024][1024];
for (int i = 1; i < 1024; i++)  {
       for (int j = i; j < 1024; j++)  {
         array[i][j]  = array[i-1][j-1] + array[i-1][j];
       }
   }
```

1. What is the cache miss rate if the cache is 64Kbyte in size. Miss rate = _____
2. What is the cache miss rate if the cache is 3-Kbyte in size. Miss rate = _____
3. What loop transformations that we have studied in class can reduce cache misses when the cache is 3KBytes in size. Loop transformation is _____

Problems 0, 2, and 5 should be submitted for grading.