CSCI 2021 Machine Architecture and Organization, Fall 2018, Written Assignment #4

Instructions:

- This assignment must be done individually.
- Posted Monday November 26 and due on Friday December 7
- This assignment must be submitted as a PDF to Canvas by 11:55PM on the due date, there is not a late option.
- You may type your assignment or you may hand write your assignment and submit a scanned copy to Canvas. If you do not have access to a scanner, use an app such as CamScanner on your phone.
- Your assignment must be legible. If you turn in an assignment that we cannot clearly read, we are not obligated to grade it and can give it a 0. If you are concerned about the legibility of your handwriting, please type your assignment.
- Along with your name, include your student ID number, x500 (internet ID), and discussion section at the top of your assignment.
- There are **four** problems; we will go over them in lab after the due date.
- The textbook in this context is: R. Bryant, D. O'Hallaron, Computer Systems: A Programmer's Perspective (3rd Edition)

Problem 1 (35 points)

Consider the following matrix transpose routine:

```
typedef int array[4][4];

void transpose (array dst, array src) {
     int i, j;
     for(i = 0; i < 4; i++) {
          for(j = 0; j < 4; j++) {
               dst[i][j] = src[j][i];
          }
     }
}
```

Assume that this code runs on a machine with the following properties:

- The `int` type is 4 bytes long
- The `src` array starts at address 0x00 and the `dst` array starts at 0x50
- There is a single L1 data cache that is direct-mapped, write-through, write-allocate, with a block size of 8 bytes.
- The cache has 8 data lines, for a total of 64 data bytes, and the cache is initially empty

- Accesses to the `src` and `dst` arrays are the only sources of read and write misses

For each row and column, indicate whether the 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.

dst	Col 0	Col 1	Col 2	Col 3
Row 0	m			
Row 1				
Row 2				
Row 3				

src	Col 0	Col 1	Col 2	Col 3
Row 0	m			
Row 1				
Row 2				
Row 3				

What is the cache miss rate for this function?

Problem 2 (25 points)

For this question, we consider the memory system of a small embedded processor. The size of the physical address space is 4K bytes, and the memory is byte-addressable. The single-level cache is 3-way associative, with a 2-byte block size and 24 total lines.

In the following tables, all numbers are given in hexadecimal. The content of the cache is as follows (V = Valid, B0 = Byte 0, B1 = Byte 1):

3-way set associative cache

Index	Tag	V	B0	B1	Tag	V	B0	B1	Tag	V	B0	B1
0	27	0	C7	B1	C8	1	86	DE	21	0	E3	E1
1	A6	1	FA	DD	53	0	AD	0F	7B	1	A1	47
2	FC	1	D9	8D	0B	0	B2	39	FD	1	6A	AC
3	·1E	0	1E	62	36	1	8F	B5	1A	1	D7	92
4	F1	1	BE	BF	CE	0	D5	21	A9	1	19	95
5	49	0	55	11	7A	1	C4	16	66	0	18	2F
6	8B	1	F2	BD	1C	0	14	1D	FD	1	01	97
7	D1	1	05	39	8F	1	BA	E7	38	1	D1	B8

A. Please indicate (by labeling the following diagram) the bits in the physical memory address that would be used to determine the following (ignore extra unused fields): CO the cache offset, CI the cache set, CT the cache tag.

B. For physical address **0xA64**, indicate the cache entry accessed and the cache byte value returned in hexadecimal. Indicate whether a cache miss occurs. If there is a cache miss, enter "unknown" for "Cache Byte returned." First, write the physical address in the same format as above, putting one bit per box and ignoring unused boxes.

Then, compute the following parameters of the cache access:

Cache Offset (CO)	
Cache Index (CI)	
Cache Tag (CT)	
Cache Hit? (Y/N)	
Cache Byte Returned	

C. Repeat part B, with the physical address **0x367**.

Cache Offset (CO)	
Cache Index (CI)	
Cache Tag (CT)	
Cache Hit? (Y/N)	
Cache Byte Returned	

D. What is the total size of this cache in bits, including the data bits, and the space used to store tags and valid bits?

E. If we can improve the cache hit rate of a program from 95% to 97% on a cache with a hit time of 15 cycles and a miss penalty of 300 cycles, what is the percentage of improvement in its average memory access time?

Problem 3 (20 points)

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 that the cache can store), *B* is the block size in bytes, *E* is the number of lines per set, *S* is the number of cache sets, *t* is the number of tag bits, *s* is the number of set index bits, and *b* is the number of block offset bits. For each cache, use what is given to fill out the rest of the row.

Cache	*m*	*C*	*B*	*E*	*S*	*t*	*s*	*b*
1	16	512	4	8				
2	16	512	2				6	
3	32	1024	16	4				
4	32	1024				26		4
5	32	2048			16			4

Problem 4 (20 points)

Given the code below, describe two optimizations that could be used to improve the performance of the function `func`. These optimizations may relate to cache performance discussed in Chapter 6 or any of the optimizations discussed in Chapter 5 of the textbook. For each optimization, either provide an example of the code that would implement the optimization, or describe how to

do it in sufficient detail that it could be easily implemented. In addition, briefly explain why the change improves the program's performance.

```
int f(int x, int y) {
   return x * x + y * y;
}

void func(int mtx[N][N], int* res) {
   *res = 0;
   for(int i = 0; i < N; i++) {
      for(int j = 0; j < N; j++) {
         for(int k = 0; k < f(i, j); k++) {
            *res = *res + f(mtx[j][i], i + j) + f(mtx[j][i], k);
         }
      }
   }
}
```