CS 382: Homework Assignment 4
Due: December 3, 11:55pm

Collaboration Policy. Homeworks will be done **individually**: each student must submit their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. If some code was shown in class, it can be used, but it must be obtained from Canvas, the instructor or the TA.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Deliverable. A single **pdf** file on Canvas. **HANDWRITTEN HOMEWORK IS PROHIBITED**.
Pay attention to the section of the textbook each problem is based on.

Problem 1 (20 points) In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where *elements within the same row of a matrix are stored contiguously.* You can assume that undefined symbols are primitive types, i.e. int, double etc. Answer the following questions based on **Section 5.1** of the textbook.

```
for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {
        Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
        err = max(err, fabs(Anew[j][i] - A[j][i]);
    }
}
```

(a) Which variables exhibit temporal locality?

(b) Which variables exhibit spatial locality? Locality is affected by both the reference order and data layout.

(c) Would the program be slower or faster if the outer loop iterated over i and the inner loop iterated over j? Why?

Problem 2 (30 points) Below is a list of 8-bit memory address references, given as **word** addresses. Answer the following questions based on **Section 5.3** of the textbook.

```
0x43, 0xc4, 0x2b, 0x42, 0xc5, 0x28,
0xbe, 0x05, 0x92, 0x2a, 0xba, 0xbd
```

(a) For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

(b) For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Problem 3 (10 points) Cache block size (B) can affect both miss rate and miss latency. Assuming a machine with a base CPI of 1, and an average of 1.35 references (both instruction and data) per instruction, find the block size that minimizes the total miss latency given the following miss rates for various block sizes.

Block Size	8	16	32	64	128
Miss Rate	5%	3%	2%	1.5%	1.1%

Answer the following questions based on **Section 5.3** of the textbook.

(a) What is the optimal block size for a miss penalty of 30 $\times$ B cycles?

(a) What is the optimal block size for a miss latency of 24 + B cycles?

(c) If miss latency was constant (independent of B), what is the optimal block size?

Problem 4 (20 points) Consider a **byte addressing** architecture with 64-bit memory addresses.

(a) Which bits of the address would be used in the tag, index and offset in a direct-mapped cache with 512 1-word blocks.

(b) Which bits of the address would be used in the tag, index and offset in a direct-mapped cache with 64 8-word blocks.

(c) What is the ratio of bits used for storing data to total bits stored in the cache in each of the above cases?

(d) Which bits of the address would be used in the tag, index and offset in a two-way set associative cache with 1-word blocks and a total capacity of 512 words.

Problem 5 (20 points) Given a byte-addressed memory with 12-bit addresses, and an attached four-way set associate cache with a total of 16 one-word blocks, make a table showing the final state of the cache after accessing the following memory addresses. The table should include the tags and data that will be stored in cache at the end of all insertions. Data should be shown using the corresponding addresses in main memory, as in **Section 5.4** of the textbook. Use the least-recently-used rule to evict from cache as needed.

The sequence of addresses accessed is:

```
0x143, 0xc4a, 0x22b, 0x42f, 0x492, 0x2a2, 0x3ba, 0xb2d
```