Homework #2 – 28 pts
Submit your homework to TEACH by Wed. 10/17/2018, at 11:59pm

Your submission should be comprised of two files: a .pdf file containing answers to the questions and one .c file containing C code. Legible, handwritten solutions are acceptable for undergraduate students (472). The handwritten solutions must be electronically scanned and submitted as a PDF file. Graduate students (572) must compose their solutions in MS Word, LaTeX, or some other word processor and submit the results as a PDF file.

1. (6 pts) Suppose the existence of a direct-mapped cache design with a 32-bit address. Assume that each address maps to a single byte. Assume the cache scheme uses the following breakdown of the address bits:

Tag	Index	Offset
31-9	8-5	4-0

   a. What is the cache block size (in words)?
   b. How many entries does the cache have?
   c. Including space for the valid bits, tags, and actual block data, how many bits would be required to actually implement this hypothetical cache?

2. (10 pts) Using the cache design from problem 1, assume the following byte-addressed cache references are recorded. Assume that the cache is initially empty. Also assume that accesses occurred from left to right (e.g. address 0 was requested, then address 4, followed by address 16, etc).

Byte Address											
0	4	16	132	232	160	1024	30	140	3100	180	2180

   a. What was the number of cache hits?
   b. Create a table to show the final state of the cache including the value of each Index, Valid bit, and Tag.

3. (6 pts) Suppose the existence of a different direct-mapped cache design with a 32-bit address. Assume that each address maps to a single byte. Assume this cache scheme uses the following breakdown of the address bits:

Tag	Index	Offset
31-6	5-3	2-0

   a. What is the cache block size (in words)?
   b. How many entries does this cache have?
   c. Including space for the valid bits, tags, and actual block data, how many bits would be required to actually implement this hypothetical cache?

4. (6 pts) Again assuming an initially empty cache, repeat the steps described in problem 2.
    a. What was the number of cache hits?
    b. Create a table to show the final state of the cache including the value of each Index, Valid bit, and Tag.

5. This question is not graded, and does not need to be submitted. It is for your benefit to prepare yourself. As future work for this class it’s likely that you will need to implement a model of a cache using C or C++ (your choice).
    a. If you are provided with the number of tag bits, index bits, and offset bits, how could you write a program to automatically calculate the answers for questions 1a through 1c?
    b. Can you envision a strategy to implement a cache model in C using **structs**?
    c. Are you more comfortable implementing a cache model using the available containers in C++?
    d. Take some time to envision your design. If the user provides you with a list of memory accesses, how could your design maintain your model cache?
    e. The idea is that the hardware caches we discuss in class could be modeled by your code. Complete with read accesses and writes.