CSCI 2500 — Computer Organization
Lab 07 (document version 1.0)
Ca$he Money

- This lab is due by the end of your lab session on Wednesday, December 1, 2021.
- This lab is to be completed **individually**. Do not share your code with anyone else.
- You **must** show your code and your solutions to a TA or mentor and answer their questions to receive credit for each checkpoint.
- Labs are available on Mondays before your lab session. Plan to start each lab early and ask questions during office hours, in the Discussion Forum on Submitty, and during your lab session.

This lab will be calculating a few different values relevant to direct-mapped caches. You'll be determining the number of cache blocks, the index address for a cache entry, as well as the size of a cache block given some parameters.

Fill in the provided template code given in `lab07.c`. This has empty functions for all of the checkpoints and instructions on what to implement. You can add helper functions as necessary (but it reaaallly shouldn't be necessary).

1. **Checkpoint 1:** Checkpoint 1: This lab focuses on material from Chapter 5 regarding cache memory. For the first checkpoint, download the `lab07.c` code. Walk through the code to understand what is expected overall.

   Figuring out how many blocks (also known as slots or lines) exist in your direct-mapped cache is rather straightforward. To do so, determine how many possible index bit combinations exist. For example, with two bits, we have binary values `00`, `01`, `10`, and `11`, which is simply 2^2.

   Implement this in the `block_count()` function. Test cases are given in `main()`.

2. **Checkpoint 2:** For the second checkpoint, you will determine the index of a would-be cache entry via the `get_index()` function. More specifically, given an address, a starting point for your index, and the ending point of your index, implement logic in the `get_index()` function to find the appropriate index in a direct-mapped cache.

3. **Checkpoint 3:** For the third checkpoint, you will determine the size of a cache block via the `get_cache_block_size()` function. More specifically, given the starting and ending point of your index as well as the block offset bits (BOBs), determine the total cache block size.