Jane Lockshin
CSCI 564: Advanced Computer Architecture
Homework 2 (02/22/2018)

1. **Cache performance:**
    a. **Cache size:**
        i. Too small: Higher miss rates from capacity misses.
        ii. Too large: Increased time to search addresses results in increased hit times.
    b. **Line size:**
        i. Too small: Higher miss rates from compulsory misses.
        ii. Too large: Higher miss rates from conflict misses, and higher miss penalties due to increased time in fetching missed blocks from lower level.
    c. **Associativity:**
        i. Direct-mapped: Higher miss rates due to conflict misses.
        ii. Fully associative: Higher hit times due to increased tag comparisons.

2. **In matrix_add function:**
    a. **L1 D-cache miss rate:**
        i. Total number of sets = cache size/(block size * associativity) = 64 KB/(64 bytes * 2) = 512 entries (Index field = 9 bits, offset = 6 bits).
        ii. For every 16 iterations, there will be 48/48 misses (100% miss rate). The first 3 will be compulsory misses (3/48 or 6.35%), and the remaining 45 are conflict misses (45/48 or 93.75%).
    b. **Average memory access time:**
        i. Hit time + miss rate x miss penalty = 1 cycle + 100% x 20 cycles = 21 cycles.

3. If you double the size of a 16KB cache, you can make an array that needs 32KB in order to have a 100% hit rate. However, if this code ran in only 16KB, the first two iterations will be compulsory misses, and the rest will be capacity misses due to the smaller cache size.
   C code:
```
int i = 0, int j = 0;
int arr[8192];
while (i < 99999) {
        j = 0;
        while (j < 8192) {
                arr[j]++;
                j = j + 4;
        }
}
```