CS 203 (2019 Fall) Assignment #1

Student ID #:

Name:

1. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's Law depends on the fraction of the original, unenhanced execution time that could make use of enhanced mode. Thus we cannot directly use this 50% measurement to compute speedup with Amdahl's Law.
   (1) What is the speedup we have obtained from fast mode?

   (2) What percentage of the original execution time has been converted to fast mode?

2. When making changes to optimize part of a processor, it is often the case that speeding up one type of instruction comes at the cost of slowing down something else. For example, if we put in a complicated fast floating-point unit, that takes space, and something might have to be moved farther away from the middle to accommodate it, adding an extra cycle in delay to reach that unit. The basic Amdahl's Law equation does not take into account this trade-off.
   (1) If the new fast floating-point unit speeds up floating-point operations by, on average, 2x, and floating-point operations take 20% of the original program's execution time, what is the overall speedup (ignoring the penalty to any other instructions)?

   (2) Now assume that speeding up the floating-point unit slowed down data cache accesses, resulting in a 1.5x slowdown (or 2/3 speedup). Data cache accesses consume 10% of the execution time. What is the overall speedup now?

(3) After implementing the new floating-point operations, what percentage of execution time is spent on floating-point operations? What percentage is spent on data cache accesses?

3. You are trying to appreciate how important the principle of locality is in justifying the use of a cache memory, so you experiment with a computer having an L1 data cache and a main memory (you exclusively focus on data accesses). The latencies (in CPU cycles) of the different kinds of accesses are as follows: cache hit, 1 cycle; cache miss, 110 cycles; main memory access with cache disabled, 105 cycles.
   (1) When you run a program with an overall miss rate of 3%, what will the average memory access time (in CPU cycles) be?

   (2) Next, you run a program specifically designed to produce completely random data addresses with no locality. Toward that end, you use an array of size 1 GB (all of which fits in the main memory). Accesses to random elements of this array are continuously made (using a uniform random number generator to generate the elements indices). If your data cache size is 64 KB, what will the average memory access time be?

(3) If you compare the result obtained in part (2) with the main memory access time when the cache is disabled, what can you conclude about the role of the principle of locality in justifying the use of cache memory?

(4) You observed that a cache hit produces a gain of 104 cycles (1 cycle vs. 105), but it produces a loss of 5 cycles in the case of a miss (110 cycles vs. 105). In the general case, we can express these two quantities as G (gain) and L (loss). Using these two quantities (G and L), identify the highest miss rate after which the cache use would be disadvantageous.