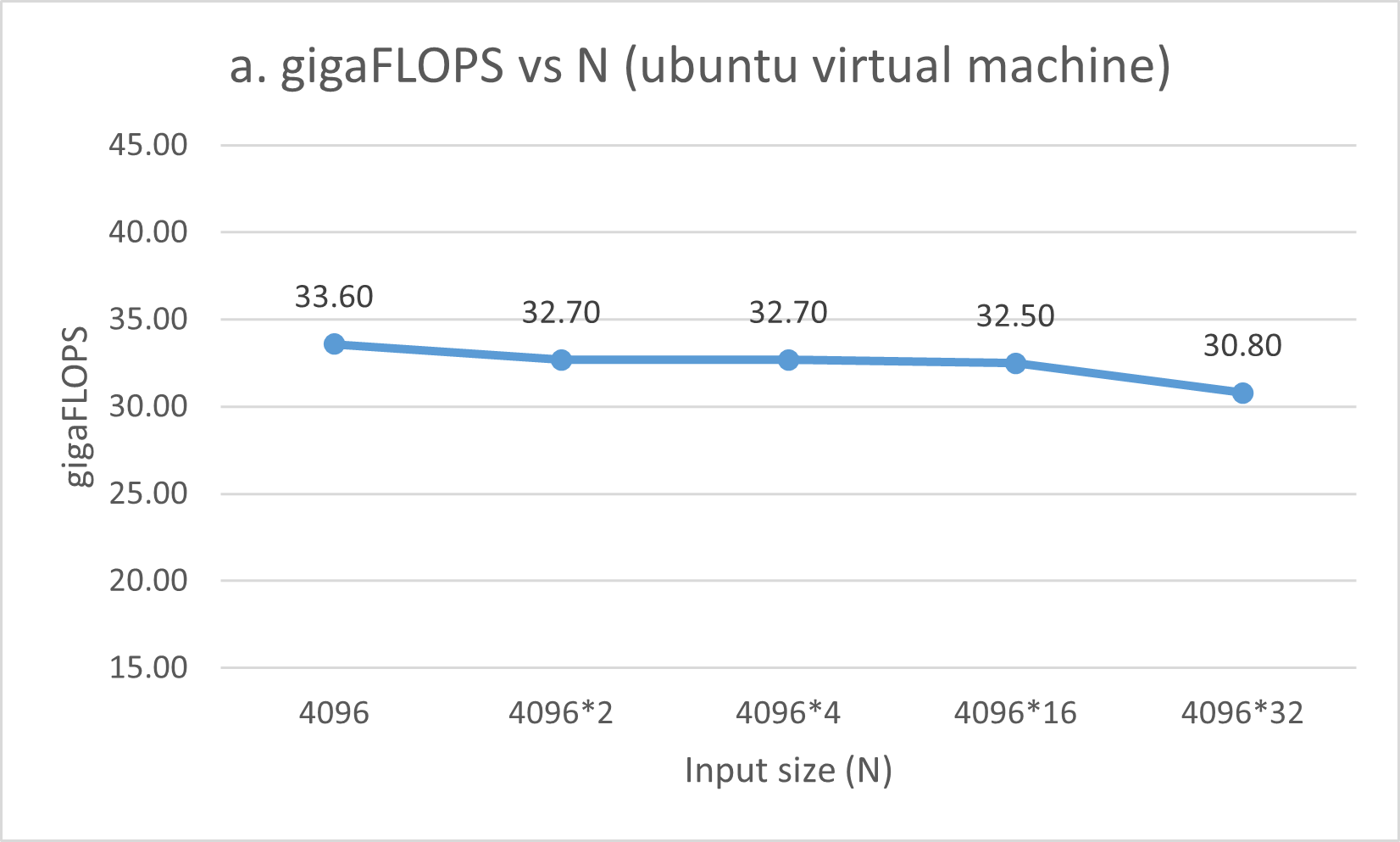
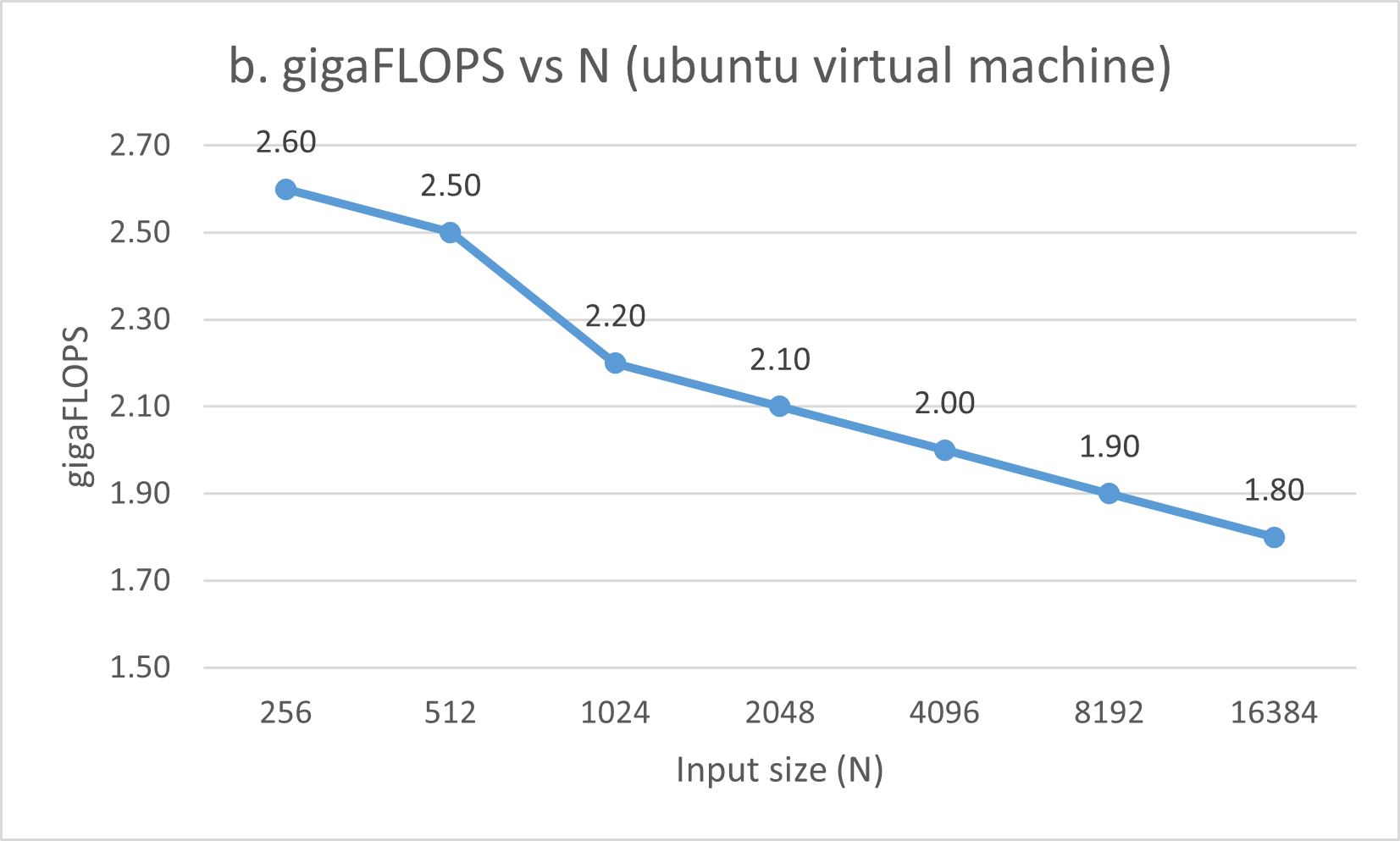
Question 1A.



Question 1B.



Question 2.

Graph A shows that when the routine in q1a.c is run with increasingly higher input sizes, the calculated GFLOPS value roughly stays the same. This indicates that the program is compute-bound, meaning that the program has a high arithmetical intensity. The GFLOPS value does not increase because the program is running at peak FLOPS, due to the balanced fmadd intrinsic instructions and minimal loads/stores.

Graph B’s GFLOPS value decreases slightly with higher input sizes; q1b.c is memory bound and has a low arithmetical intensity. Increasing the input size in this program increases the amount of cache misses, resulting in the CPU spending more time in slower memory trying to find and load the data.

Assuming that variables are being stored as 4 bytes each the arithmetical intensity of each routine is as follows:

* Q1a AI: 16/(2\*4) = **2.0**
* Q1b AI: 3/(5\*4) = **0.15**

Using the following equation, the peak FLOPS of the computer running the code has been calculated:

*Peak.FLOPS = num.CPU.cores x CPU.freq. x simd.length.in.bits / 32 x num.FMA.units x 2*

**345.6 FLOPS** = 4 cores x 2.7 GHz \* 256 bits / 32 \* 2 FMA units \* 2

Why are the measured values for q1a.c and q1b.c lower than the theoretical peak FLOPS of the computer?

q1a.c is a compute-bound program and has a high AI but the recorded GFLOPS values are still lower than the theoretical peak GFLOPS value because the program is being run on a single core, just like q1b.c. Even when it’s considered that the peak performance of one core is **86.4** GFLOPS(345.6/4), the code runs at much less than this, which is likely because of Operating System interrupts and also because the program is being run within a virtual machine that uses less system resources than what is available. Furthermore, this suggests that the code could potentially be further optimized.

q1b.c is a memory-bound program and has a low AI and therefore is bottlenecked by memory. The routine within q1b.c that is being measured has 4 loads and 1 store and is therefore repeatedly accessing memory. Because of Memory Hierarchy, the CPU will spend a large amount of time trying to find the data within L1, L2 and L3 cache and then eventually RAM. As the CPU goes through the hierarchy the number of cycles taken to fetch data increases, therefore the program takes longer to execute and the recorded GFLOPS value is significantly lower. This is all because the arrays become too big to fit within cache.