John Mikko Velasquez
Coding Project #2
CSC 656
10/19/25

What types of operations are more expensive and why, and which of the codes is performing a larger number of more expensive operations?
Memory access operations are more expensive because they require fetching data from memory which could result in cache misses, resulting in more operations compared to just arithmetic operations.
From largest to lowest number of expensive operations: sum_indirect, sum_vector, and sum_direct.

Computational rate. Which of the 3 methods has the best computational rate (MFLOP/s)? Why?
Direct sum has the best computational rate because it has no memory access operations. So it's purely just doing arithmetic operations (which is faster than memory access operations) allowing it to do more per second.

Memory bandwidth usage. Of the 2 methods vector sum and indirect sum, which has higher levels of memory bandwidth utilization? Why?
Vector sum has higher levels of memory bandwidth utilization because it performs sequential memory access whereas indirect sum performs non-sequential memory access. Meaning that vector sum is able to have higher rates of cache hits because the data is next to each other (spatial locality), whereas indirect sum will have lower cache hit rates because the data is all over the memory. As a result, data is transferred from memory faster and more effectively in vector sum.

Memory latency. Of the 2 methods vector sum and indirect sum, which shows lower levels of memory latency? Why?
Vector sum shows lower levels of memory latency because it performs sequential memory access leading to higher cache hit rates compared to indirect sum. For example, a block in the cache might contain the next 4 integers in the array resulting in faster access, but in indirect sum, data is all over the place. As a result, it's more efficient and faster in getting the data from memory.