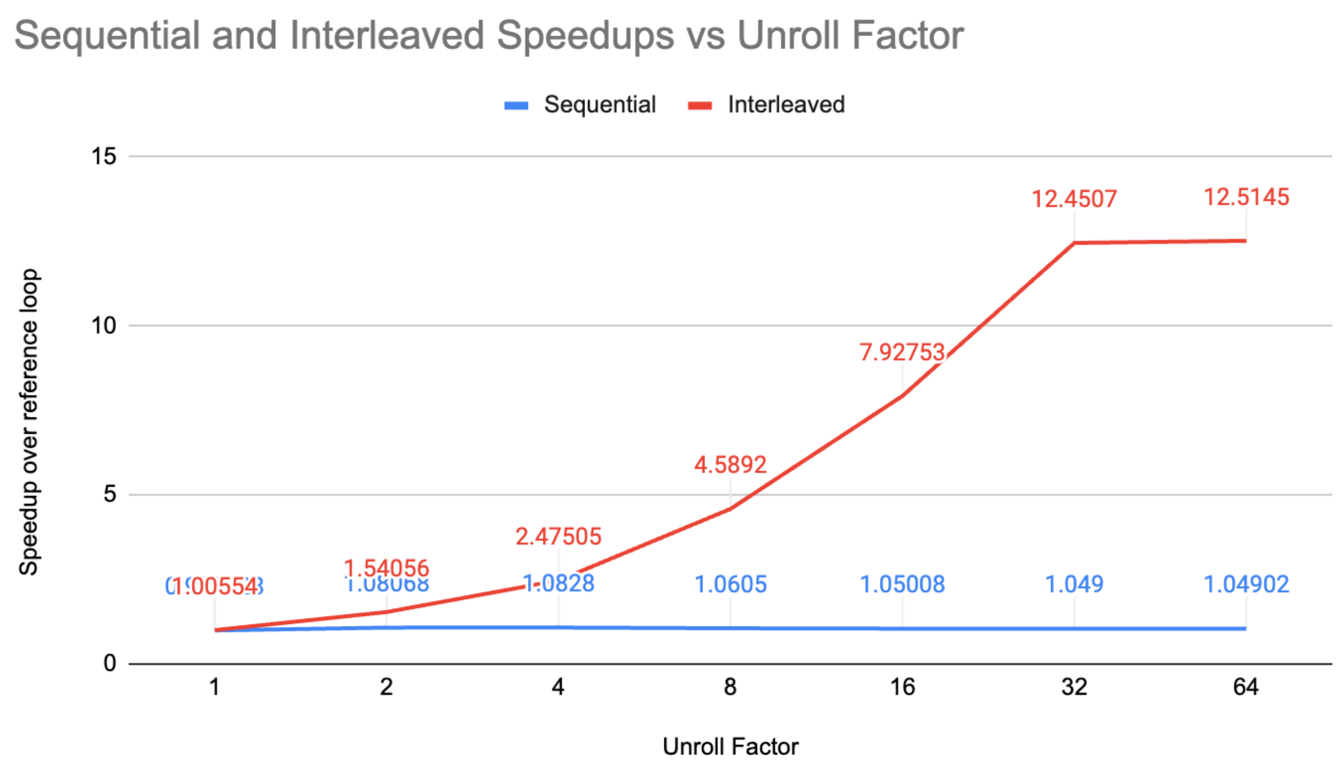
Pranav Vempati
Thomas Banghart
CSE 211: Compiler Design
HW 3

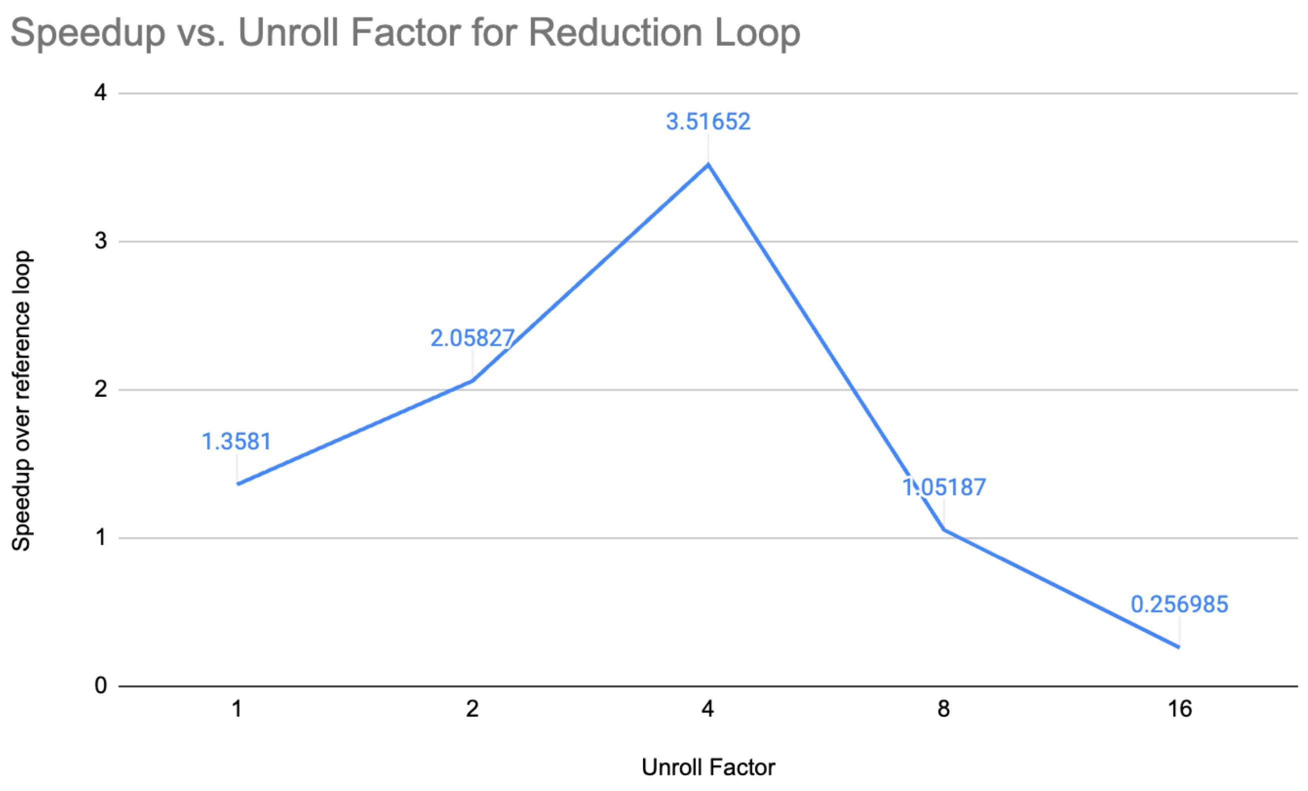
Part 1.1: Loop Unrolling for ILP



As the unroll factor increases, the interleaved method exhibits a more pronounced speedup compared to the sequential method. This is indicative of the interleaved method's ability to better leverage the hardware capabilities, as it circumvents data dependency bottlenecks associated with the updated floating point value. The slight speedup in the sequential unrolling is most likely due to the decreased number of branch instructions but due to the data dependencies between the floating point addition, unrolling has a limited impact on the overall runtime of the program.

The critical point here is that interleaving doesn't alter the logical flow of computations; it simply rearranges the order of execution to maximize parallelism where possible. Since the floating point value of `tmp` needs to be evaluated before the next operation, reordering instructions is not possible. However, interleaving ensures that while one instance of the computation is evaluating the current value of tmp, another instance can simultaneously start processing the next iteration, leading to a more efficient utilization of available resources via instruction level parallelism.

Part 1.2: Loop Unrolling for ILP in Reduction Loops



As the unroll factor increases, the speedup initially rises, indicating improved performance. However, at an unroll factor of 8, the speedup starts to diminish, and by 16 it incurs slowdown. The diminishing returns observed in the speedup are likely attributed to the overhead introduced by the loop unrolling process.

In our specific implementation, the calculation of read and write indices—expressed as `a[(size/partitions) * j] = a[(size/partitions) * j + i]`—involves multiplication and addition operations to determine the memory addresses for each partition. As the number of partitions grows with larger unroll factors, these addressing calculations become more frequent and complex, contributing to the overhead. Moreover, the management of these indices and the corresponding partial sums requires additional registers or memory accesses, which can lead to register pressure and increased cache misses. These factors can negate the benefits of unrolling, as evidenced by the performance drop at higher unroll factors.