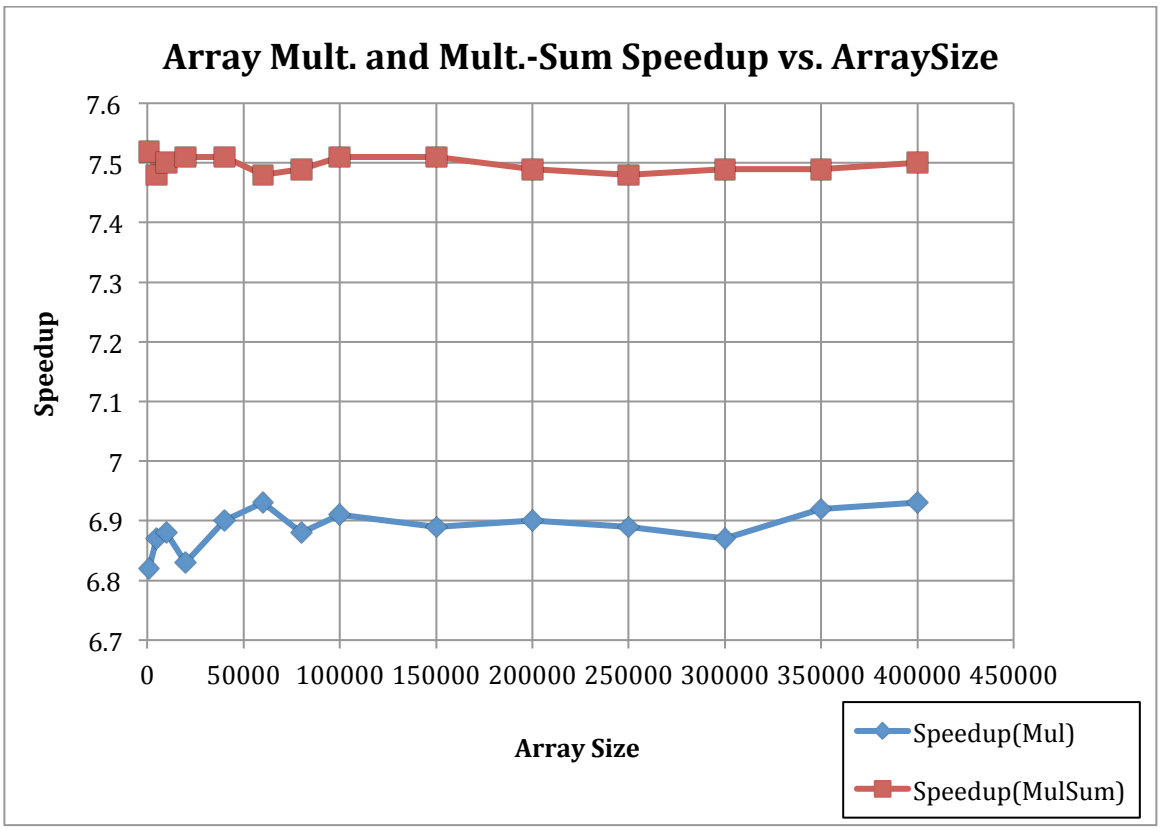
Project 4: Vectorized Array Multiplication and Reduction using SSE

1. **What machine you ran this on.**

   - Flip machine, although the uptime was really high and so I'm really not sure about the recorded performances. But it was unusually high for the whole day.

2. **Show the table and graph.**

ArraySize	Speedup(Mul)	Speedup(MulSum)
1000	6.82	7.52
5000	6.87	7.48
10,000	6.88	7.5
20,000	6.83	7.51
40000	6.9	7.51
60000	6.93	7.48
80000	6.88	7.49
100000	6.91	7.51
150000	6.89	7.51
200000	6.9	7.49
250000	6.89	7.48
300000	6.87	7.49
350000	6.92	7.49
400000	6.93	7.5



3. **What patterns are you seeing in the speedups?**

   Apart from the slight dips and rises through different array sizes, the speedups for array multiplication and array multiplication sum remains almost consistent throughout the array sizes. However, there is a good difference between the speedups for array multiplication and array multiplication sum. The speedup for Array Multiplication Sum is consistently higher than the speedup of Array Multiplication.

4. **Are they consistent across a variety of array sizes?**

   The patterns are almost consistent across a variety of array sizes. There are slight dips and rises in the graph for certain array sizes but overall they seem to vary within a small range and hence the graph looks almost consistent for varying array sizes.

5. **Why or why not, do you think?**

   The speedup of array multiplication sum seems higher than the array multiplication speedup, which may be happening due to reduction. As we have

seen earlier in the class that reduction helps improve performance a lot. Therefore, the same can be observed with the speedups here as the speedups are calculated as S = Psse/Pnon-sse = Tnon-sse/Tsse (P = Performance, T = Elapsed Time) for Array Mult and Array MultSum. The occasional dips in speedups can majorly be due to the machine it's been run on.

6. **Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-mutiplication?**
   As SSE SIMD is 4-floats-at-a-time, the processor fetches 4 float values from the cache at once. Since the values are in cache, when we get a cache hit some amount of clock cycles are incurred to access that data. Whereas, in the non-SIMD multiplication, the processor fetches single float at a time as per the requirement and the cache is accessed again and again. Hence, since there is more number of cache hits, the performance basically takes the hit and eventually goes down in non-SIMD array-multiplication.

7. **Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-mutiplication-reduction?**

   Again, as SSE SIMD is 4-floats-at-a-time, the processor fetches 4 float values from the cache at once. Since the values are in cache, when we get a cache hit some amount of clock cycles are incurred to access that data. Whereas, in the non-SIMD multiplication-reduction, the processor fetches single float at a time as per the requirement and the cache is accessed again and again. Hence, since there is more number of cache hits, the performance basically takes the hit and eventually goes down in non-SIMD array- multiplication-reduction.