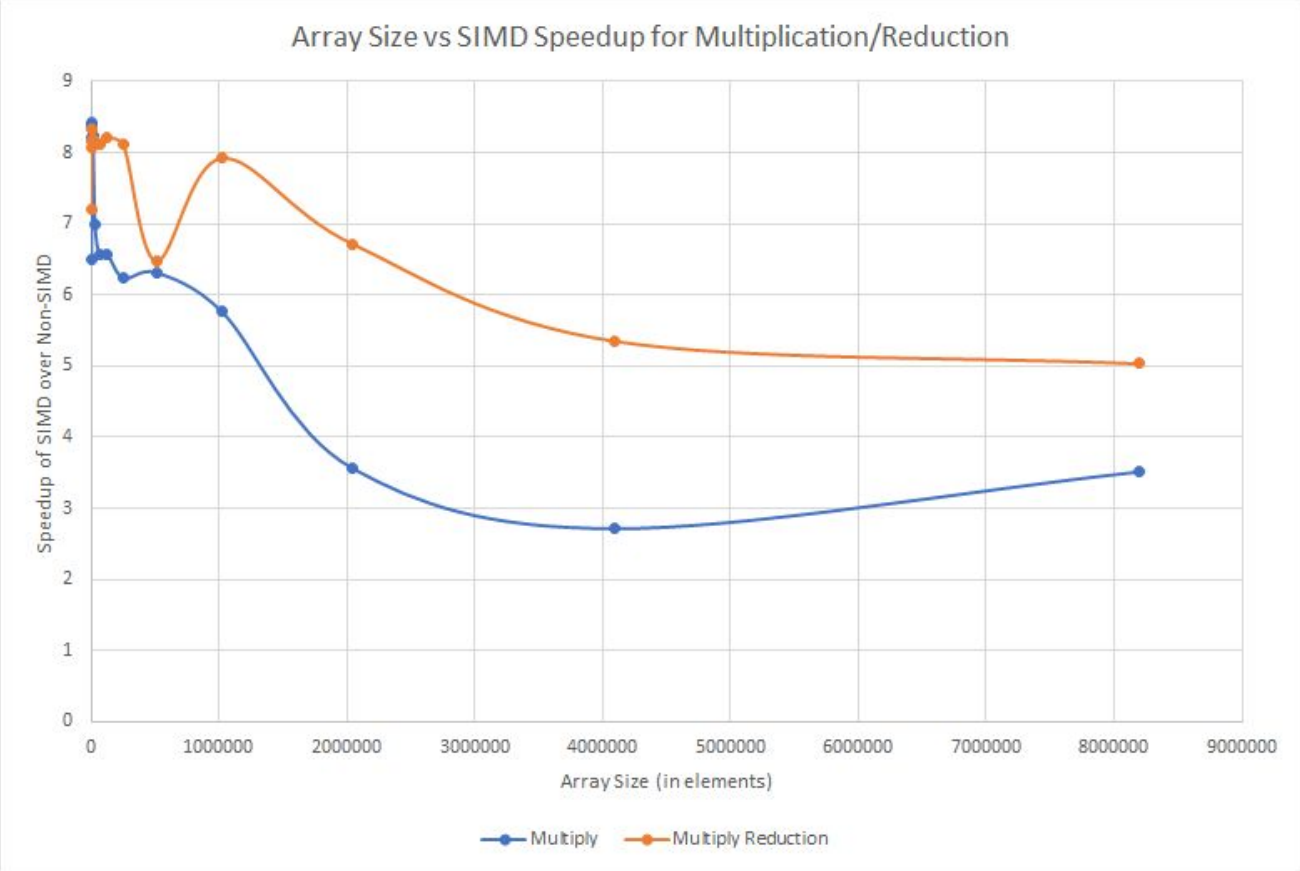
Kevin Harvell
harvellk@oregonstate.edu


**Project 4 - Vectorized Array Multiplication and Reduction using SSE**

1. **What machine you ran this on**
   This project was run on flip with uptime ~5.
2. **Show the table and graph**

| Array Size | Multiply | Multiply Reduction |
|---|---|---|
| 1000 | 8.201761 | 8.072958 |
| 2000 | 8.417009 | 8.319385 |
| 4000 | 6.509729 | 7.211365 |
| 8000 | 8.369192 | 8.176077 |
| 16000 | 8.230019 | 8.183855 |
| 32000 | 6.998756 | 8.136038 |
| 64000 | 6.575403 | 8.114446 |
| 128000 | 6.568631 | 8.212957 |
| 256000 | 6.237034 | 8.10918 |
| 512000 | 6.313782 | 6.479531 |
| 1024000 | 5.767848 | 7.928118 |
| 2048000 | 3.564031 | 6.719186 |
| 4096000 | 2.721002 | 5.35178 |
| 8192000 | 3.516195 | 5.033701 |

Kevin Harvell
harvellk@oregonstate.edu
Project 4 - Vectorized Array Multiplication and Reduction using SSE




3. **What patterns are you seeing in the speedups?**
   For multiplication SIMD speedup, speedups start out high with lower array sizes. As array sizes get bigger, the speedup drops significantly until array size gets to about 4,000,000. Then speedup starts to increase slightly, but remains below 4.
   For multiplication reduction SIMD speedup, speedups also start out high with lower array sizes. As array sizes get bigger, the speedup drops significantly until array size gets to about 4,000,000. Then speedup starts to flatline at around 5.
   Multiplication Reduction appears to speed up using SIMD significantly more than Multiplication.
4. **Are they consistent across a variety of array sizes?**
   The patterns noted above are explained mainly by referring to array sizes. With both Multiplication and Multiplication/Reduction, they both start out with high speedup numbers with smaller array sizes. As array sizes get bigger, both speedups slow down significantly until around 4,000,000 array size. It appears as array sizes get very large, Multiplication and Multiplication/Reduction start to converge between 3-5. With even larger array sizes, it would be interesting to see if this convergence holds true.
5. **Why or why not, do you think?**
   Speedups are very high for smaller array sizes because the core is able to multiply 4 floats at a time without bringing in a significant amount of data from future cache lines. Basically, the core doesn't have to work as hard to get the cache lines before it is finished processing the array. However, as the array

sizes get much larger, speedup slows down drastically because there is only so many cache lines the core can get before it has to wait for what is being processed. In other words, the data is so much that the core can't keep up and has to work a lot harder to do so.

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-multiplication?**
   Speedup is <4.0 for large array sizes because there is only so many cache lines the core can get before it has to wait for what is being processed. Basically, the data is so much that the core can't keep up and has to work a lot harder to do so.
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-multiplication-reduction?**
   Speedup is >4.0 for large array sizes because the intermediary step of adding the result of the matrix multiplication allows for the core to catch up on fetching new cache lines to multiply. This effect is diminished over time due to the sheer amount of data that needs to be fetched.

**Summary:**

This assignment showcased SIMD very well. It is very apparent to me that using SIMD can drastically speed up a program. I am not sure that I'm clear on what actually SIMD does in order to get the results I got above. I understand that SIMD takes multiple numbers or data and performs one operation on them all at once. However, it is still unclear to me exactly why speedup is <4.0 for array multiplication and >4.0 for array multiplication-reduction. Above explanations are my best, most educated guesses as to why these are occurring, but I'm not exactly sure what is really happening with the cache lines behind the scenes if I'm being honest.