CS 475 Parallel Programming: SSE Vector Arrays

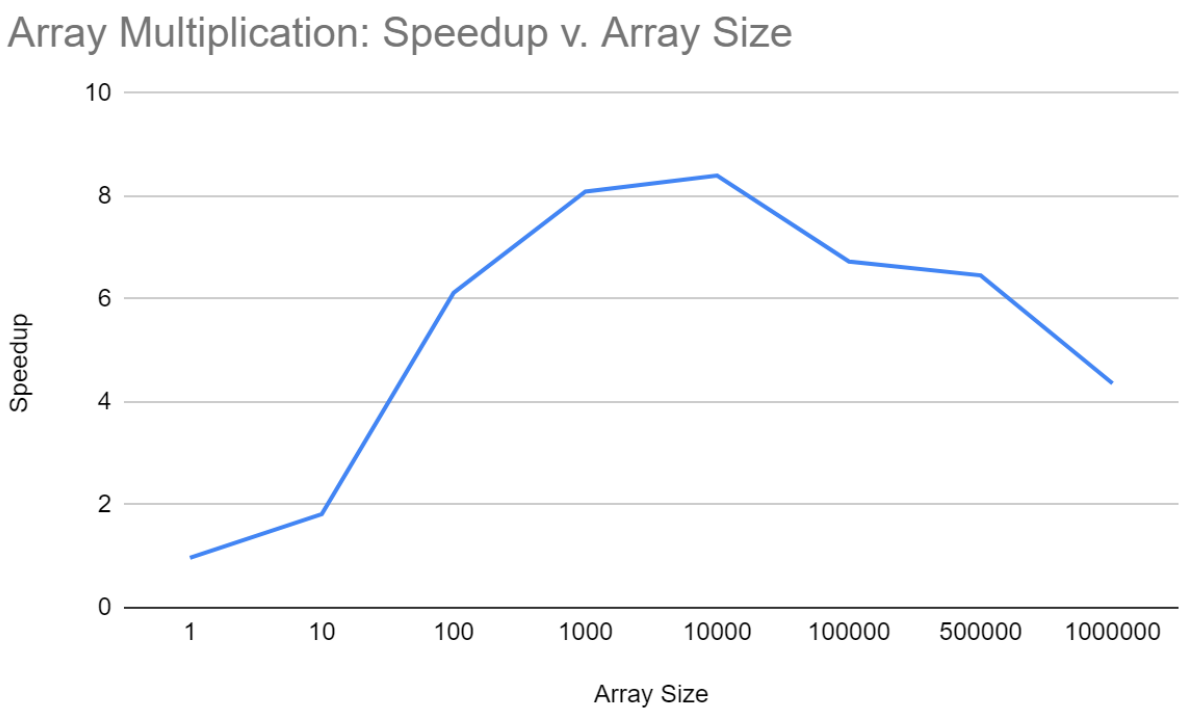
Conner Rhea
rheac@oregonstate.edu

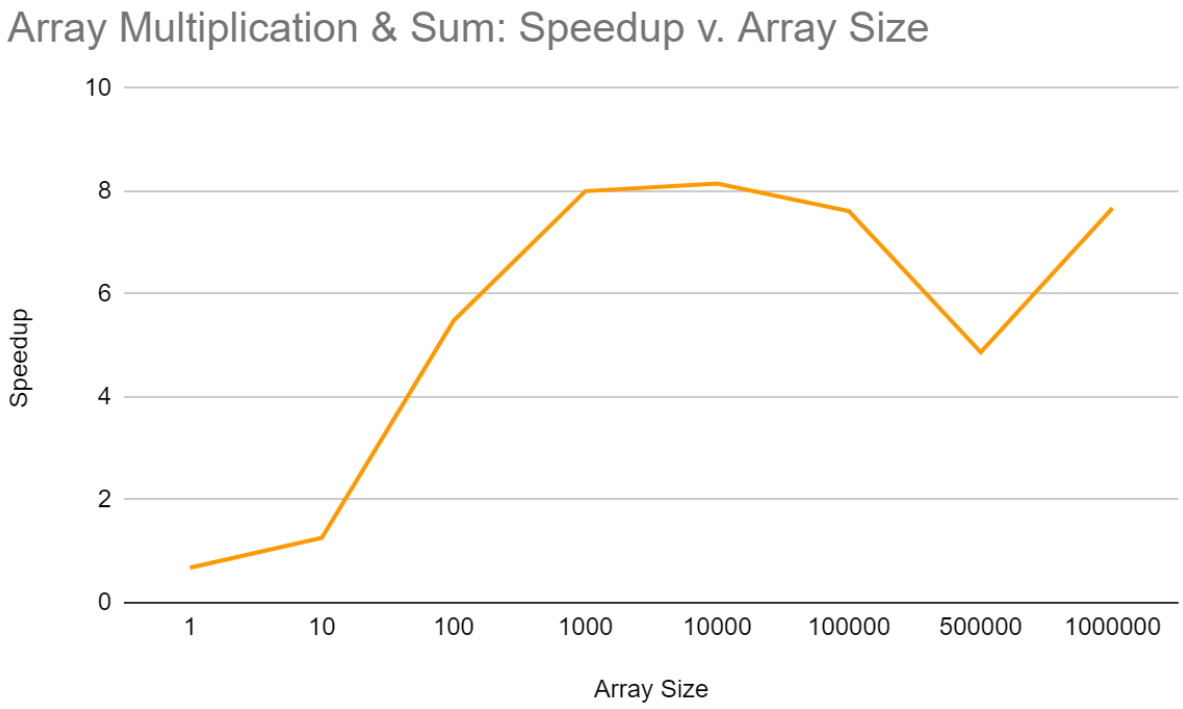
Write-Up:

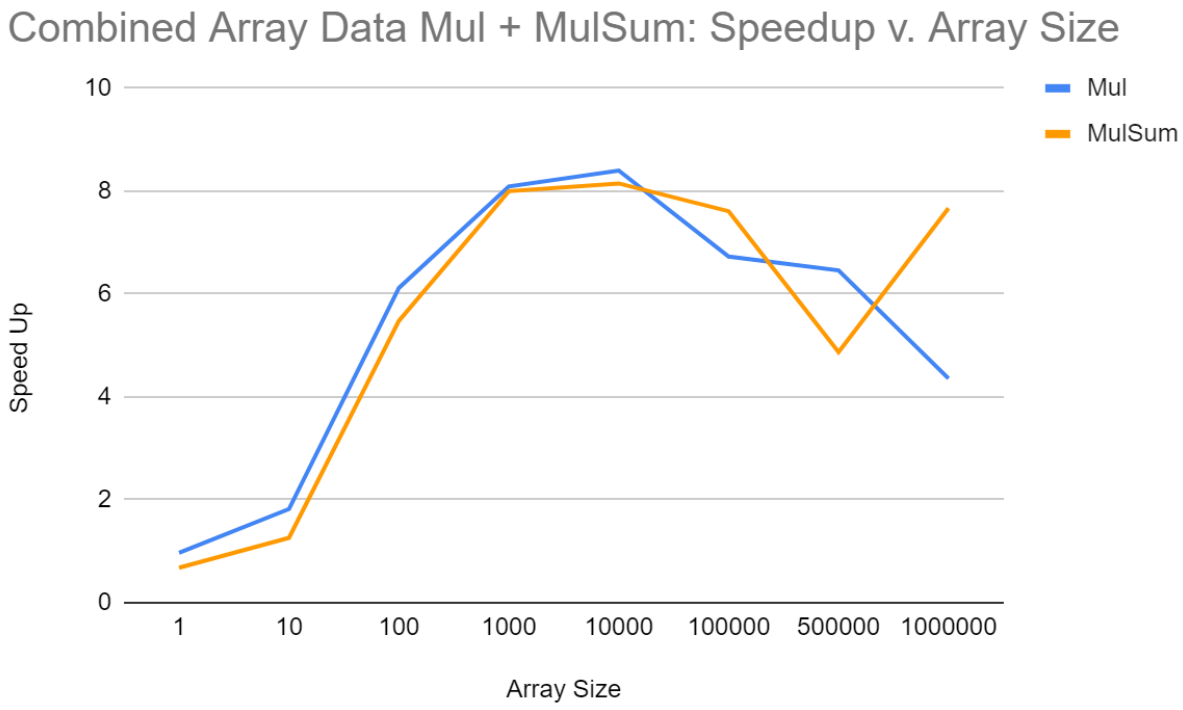
1. I ran this project's code on the Flip1 server provided by OSU.
2. Tables Created from Data:
   Output.csv created from loop.bash

1	N	28.63	S	27.71	0.97	N	27.8	S	18.88	0.68
10	N	107.64	S	196.12	1.82	N	104.25	S	131.75	1.26
100	N	160.26	S	980.59	6.12	N	161.56	S	885.56	5.48
1000	N	172.5	S	1394.7	8.09	N	172.65	S	1381.24	8
10000	N	120.88	S	1015.74	8.4	N	123.21	S	1004.73	8.15
100000	N	120.84	S	813.31	6.73	N	235.74	S	1794.37	7.61
500000	N	199.66	S	1289.87	6.46	N	203.86	S	993.03	4.87
1000000	N	219.98	S	959.31	4.36	N	233.73	S	1792.93	7.67

3. Graphs Created from Data:







4. Based on the graphs created from the data, SSE has very strong initial speedup on multiplication operations when data sizes begin to increase, however as those array sizes begin to get very large (1 million and above) the speed-up begins to fall off though still providing a tangible benefit (a

speedup value of 4). Comparatively, on Multiplication/Reduction operations, the speedup is fairly is similar, however after falling off as it approaches the 1 million mark, it jumped back up instead of continuing to fall, which might imply that it has certain break points which it struggles with or that I perhaps got a lucky outlier.

5. The Speedup seems to create a consistent bell curve like shape for both problems, however the speedup itself seems to vary depending on the Array Size, having a noticeable decline after the array reaches roughly 500,000 size.
6. This seems consistent with the SIMD notes, where the graphs begin to decline as array sizes get larger due to temporal coherence, the cache lines are not fast enough to serve the true speed of SSE, so it ends up having to wait for the cache lines to be available to continue.

Google Sheets Link: CS 475 SSE Array Multiplication