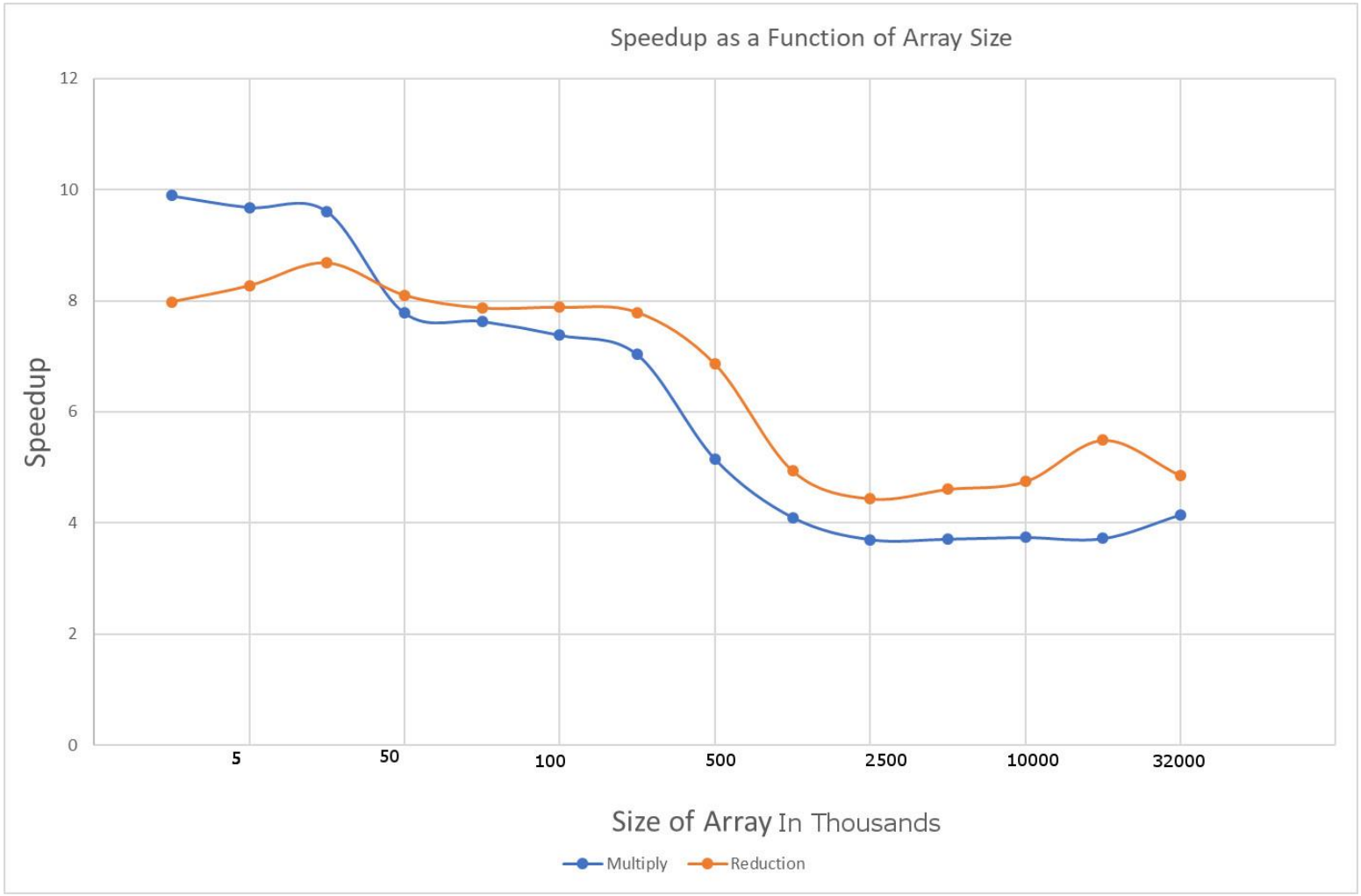
Project 5 Report CS 475 Jacob Leno

1. All tests were run on OSU's server Flip 3.
2. Table and Graph:

	1000	5000	20000	50000	70000	100000	250000	500000	1000000	2500000	5000000	10000000	20000000	32000000
Multiply	9.89	9.68	9.61	7.78	7.63	7.38	7.03	5.15	4.1	3.7	3.71	3.74	3.72	4.15
Reduction	7.98	8.27	8.68	8.1	7.87	7.88	7.79	6.86	4.94	4.44	4.61	4.75	5.49	4.85



3. At the number of elements in the array increases the speedup gained by using SIMD also goes down and hovers around 4. The same behavior is exhibited for both multiplication and reduction.
4. No, for smaller array sizes the speedup is greater when using SIMD but as the array size gets larger the speedup is not as great.
5. The function is going through the array so quickly that it is violating temporal coherence. In other words, at higher values of the array the cache line has to be reloaded so often that it causes a decrease in performance.
6. I think the reason I am getting around 8x speed up for the array multiplication is because SIMD uses only one instruction vs the C/C++ version which would use multiple assembly instructions. The fact that there are fewer instructions and that 4 floats are being multiplied at a time results in the high speedup which decreases for higher array sizes because of a violation of temporal coherence.
7. This would have a similar answer to question 6. The almost 10x speed up is because SIMD uses fewer assembly instructions and thus has less overhead but when array sizes grow larger there is a speedup decrease because of a violation of temporal coherence.