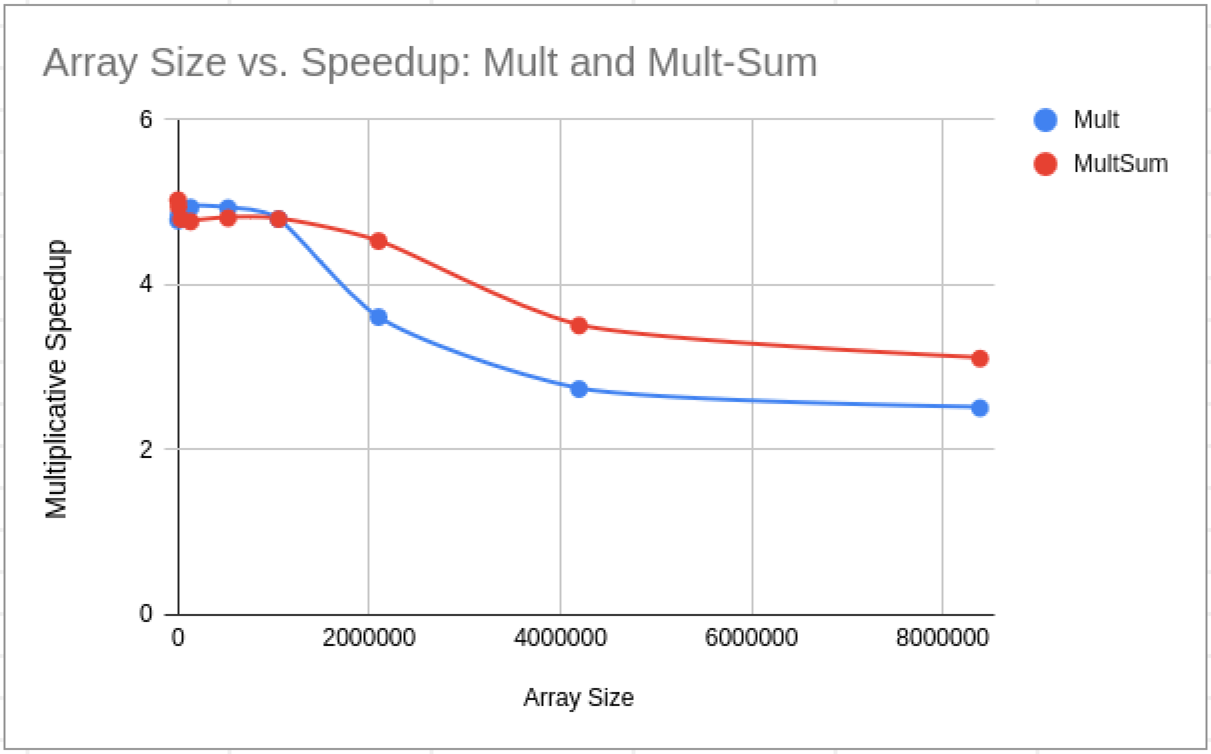
CS_475 Project #4

Benjamin Anderson II
anderbe2@oregonstate.edu

1. I ran this on the school’s flip server, as my own machine seg-faulted when I tried to run it locally (potentially an issue with them being att instructions, but I don’t quite know)
2.

Array Size	NonSIMDMultMega	SIMDMultMega	MultSpeedup
1024	397.22	1897.35	4.78
4096	391.02	1892.04	4.84
32768	390.51	1875.8	4.8
131072	388.65	1919.23	4.94
524288	386.85	1905.45	4.93
1048576	384.75	1846.74	4.8
2097152	377.56	1362.03	3.61
4194304	371.7	1017.76	2.74
8388608	369	924.83	2.51

Array Size	NonSIMDMultSumMega	SIMDMultSumMega	MultSumSpeedup
1024	395.65	1988.27	5.03
4096	397.44	1965.61	4.95
32768	398.29	1910.01	4.8
131072	398.18	1900.34	4.77
524288	396.51	1905.34	4.81
1048576	395.39	1897.43	4.8
2097152	394.22	1785.1	4.53
4194304	391.11	1371.32	3.51
8388608	388.75	1209.34	3.11

3. 

4. The only patterns I can really see are:
   a. The speed ups seem to be considerably better at lower array sizes and seem to taper out after around 4 million elements or so. They are still 2x the speed of no SIMD though, it's just not as drastic as the 5x from ~1K to ~500K.
   b. The speed-up for Mult-Sum is better (at limit) than the Mult.
5. (As answered above) No, the speed-ups are not consistent across a variety of array sizes.
6. To explain why I think this is, I'll use the toaster metaphor from the slides. The more slots a toaster has, the more bread it can toast, but the more bread you get (say 8 million slices), the more diminishing returns you get on having a 4 slot toaster vs. a 1 slot one. It'll still be faster, but the ratio falls off, and limits toward a certain value. There could also be issues with prefetching with larger datasets, but I'm less certain about the specifics of why that would be the case.