Project4

Vectorized Array Multiplication and Multiplication/Reduction using SSE

Name: Yi Chiun Chang

Email: changyic@oregonstate.edu

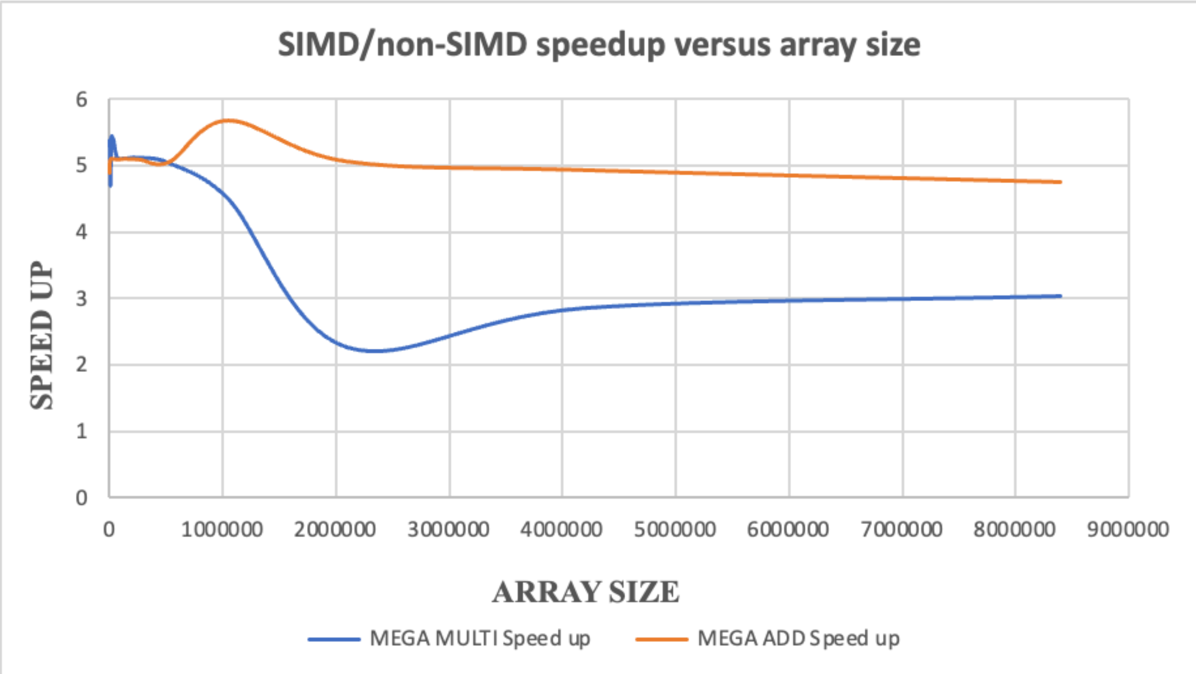
1. What machine you ran this on

CyberDuck(rabbit.engr.oregonstate.edu)

2. Show the 2 tables of performances for each array size and the corresponding speedups

Array Size	NONSIMD MEGA MULTI	SIMD MEGA MULTI	Speed Up	NONSIMD MEGA ADD	SIMD MEGA ADD	Speed Up
1024	144.00	760.22	5.28	153.99	752.95	4.89
2048	145.15	779.30	5.37	154.67	775.76	5.02
4096	154.86	832.01	5.37	165.96	842.80	5.08
8192	153.72	720.49	4.69	177.15	901.51	5.09
16384	175.36	952.67	5.43	199.42	1017.39	5.1
32768	226.11	1219.05	5.39	254.78	1298.36	5.1
65536	327.87	1678.39	5.12	354.56	1806.30	5.09
131072	328.47	1675.02	5.10	354.56	1807.44	5.1
262144	328.16	1675.80	5.11	354.62	1804.41	5.09
524288	326.45	1641.54	5.03	353.75	1787.88	5.05
1048576	326.82	1467.61	4.49	303.76	1724.72	5.68
2097152	310.22	701.01	2.26	352.91	1785.22	5.06
4194304	314.37	895.21	2.85	322.29	1588.86	4.93
8388608	291.76	884.58	3.03	318.86	1515.16	4.75

3. Show the graphs (or graph) of SIMD/non-SIMD speedup versus array size (either one graph with two curves, or two graphs each with one curve)



4. What patterns are you seeing in the speedups?

For both multiplication and addition operations, the speedup starts high (around 5x for multiplication and around 5x for addition). The speedup values remain relatively consistent up to the array size of 1048576, with minor fluctuations. Besides, for both operations, there is a noticeable drop in speedup at array sizes larger than 1048576. The speedup for multiplication drops significantly at 2097152 elements and continues to drop further. The speedup for addition also drops significantly at 2097152 elements, although the drop is less dramatic compared to multiplication

5. Are they consistent across a variety of array sizes?

According to small to medium size, the speedup values are quite consistent across different array sizes. This indicates that the SIMD operations are efficiently parallelizing the computations and taking advantage of the vectorization capabilities provided by the hardware. However, the speedup values of the large size start to decline. This suggests that the efficiency of SIMD operations is decreasing with larger array sizes.

6. Why or why not, do you think?

The observed speedups for both multiplication and addition operations show a consistent pattern for small to medium array sizes, up to 1048576 elements. During this range, the speedups are relatively stable, around 5x for multiplication and addition, indicating that SIMD operations efficiently parallelize computations of the hardware's vectorization capabilities. However, as the array sizes increase beyond 1048576 elements, there is a noticeable decline in speedup. For instance, at

2097152 elements, the speedup for multiplication drops significantly, and the trend continues with further increases in array size. This drop in speedup can be attributed to several factors. When arrays are small to medium, they fit within the CPU cache, allowing for faster memory access and higher efficiency for SIMD operations. In contrast, larger arrays exceed the cache size, leading to cache misses and higher memory latency, which diminishes the efficiency of SIMD operations. Additionally, memory bandwidth limitations become more apparent with larger arrays, causing the CPU's computation speed to outpace data fetching from memory, creating a performance backup. Furthermore, the overhead associated with setting up SIMD instructions might become more significant for larger arrays, especially if there are additional memory management issues. Consequently, while SIMD operations provide substantial speedups for smaller arrays, their efficiency diminishes with larger data sets due to cache and memory bandwidth constraints.