CS 475/575 -- Spring Quarter 2022

Project #4

Vectorized Array Multiplication/Reduction using SSE

Aasritha Jasti

934386327

jastia@oregonstate.edu

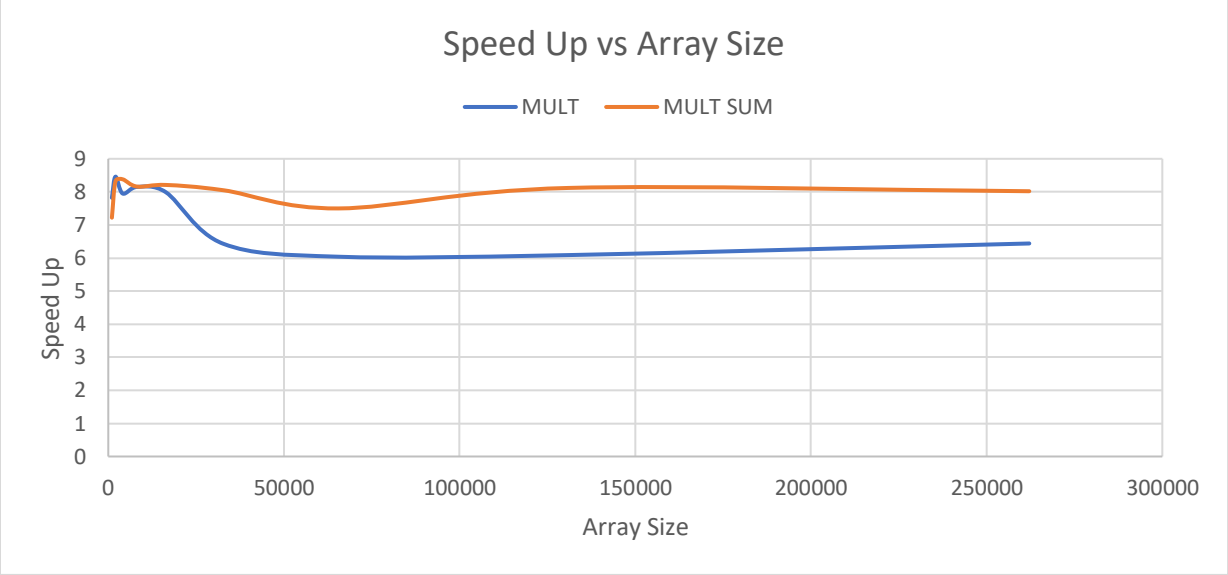
1. What machine you ran this on

   I used flip server

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

ArrSize	NON-SIMD Mul	SIMD	MULT	NON-SIMD Mul Sum	SIMD Mul Sum	MULT SUM
1024	201.17	1572.98	7.82	213.56	1542.09	7.22
2048	222.06	1879.51	8.46	224.24	1866.74	8.32
4096	222.17	1765.57	7.95	225.09	1885.95	8.38
8192	221.51	1804.88	8.15	225.82	1842.89	8.16
16384	221.77	1771.08	7.99	225.94	1855.32	8.21
32768	221.44	1424.15	6.43	225.9	1818.6	8.05
65536	221.16	1335.02	6.04	225.79	1692.35	7.5
131072	219.06	1334.97	6.09	222.71	1808.34	8.12
262144	119.89	772.21	6.44	224.86	1804.09	8.02

3. Show the 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?

   As seen in the above graph speed up vs Array size, the multiplication sum (MULT SUM) is constant for all array sizes whereas Multiplication (MULT) is also constant in starting at the level with MULT SUM but decreases a little and further remains constant.

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

   Matrix multiplication sum was constant all across, but Matrix multiplication was constant initially and started decreasing between 1000000 to 5000000 and came to a constant state after 5000000

MULTSUM is constant throughout the array size whereas MULT was in the same line with MULTSUM but gradually decreased in between 0 to 50000 after that it remained constant.

6. Why or why not, do you think?

   Multiplying the array and storing the values in another location might decrease the speed up, but when coming to Multiplying sum just multiply and adds them to a locally declared value, which might be stored in the cache or a register and can add things up in a faster and increase the performance.