JONATHAN PERRY
5/20/18

PROJECT #5

For this assignment, I created a bash script to help test my program with several different combinations of parameters –such as setting the array size and setting up which functions I specifically want to run. To run the script, adjust the permissions to execute the script and then simply enter in the following command: ./script

1. Tell what machine you ran this on
   - A Linux system on OSU's FLIP server was used to run Project #5.

2. Table of Results and Graph

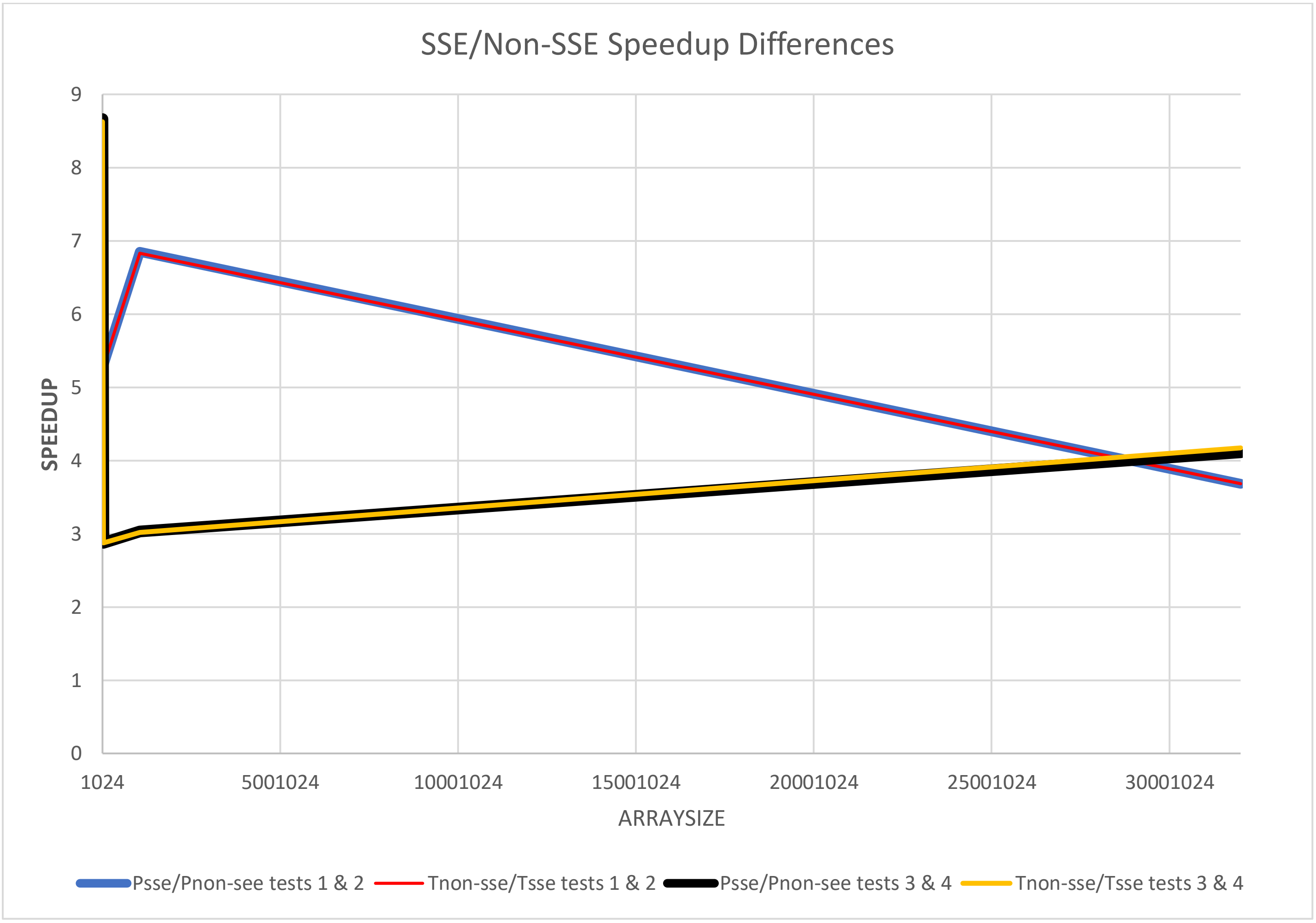
MegaMults/Sec				
ARRAYSIZE=====>	1024	32768	1048576	32000000
test 1 - SSE	617.01	649.84	1045.66	1123.21
test 2 - Non-SSE	173.45	122.69	152.67	304.92
test 3 - SSE	1607.47	666.67	644.05	1388.99
test 4 - Non-SSE	185.57	231.85	212.66	337.7

Time Elapsed				
ARRAYSIZE=====>	1024	32768	1048576	32000000
test 1 - SSE	0.00000166	0.00005055	0.00100564	0.02848968
test 2 - Non-SSE	0.00000621	0.00026709	0.00686839	0.10494413
test 3 - SSE	0.00000064	0.00004915	0.00163252	0.02303831
test 4 - Non-SSE	0.00000552	0.00014133	0.00493081	0.09615397

Speedup				
ARRAYSIZE=====>	1024	32768	1048576	32000000
Psse/Pnon-sse tests 1 & 2	3.557278755	5.29660119	6.849151765	3.683621934
Tnon-sse/Tsse tests 1 & 2	3.740963855	5.283679525	6.829869536	3.683584021
Psse/Pnon-sse tests 3 & 4	8.662337662	2.875436705	3.028543215	4.11308854
Tnon-sse/Tsse tests 3 & 4	8.625	2.875483215	3.020367285	4.173655533

JONATHAN PERRY
5/20/18

PROJECT #5



3. The patterns I'm seeing with the speedup is that the graph for Psse/Pnon-see and tnon-sse/Tsse both tests 1 and 2 run parallel to each other, as do tests 3 and 4 run parallel to each other too. I'm also seeing an increase in speedup from array size 1024 to array size 32768 with both psse/Pnon-sse tests and Tnon-sse/Tsse for tests 1 and 2, followed by a steady decrease in speedup after. Lastly, I'm seeing a heavy decrease in speedup from array size 1024 to array size 32768 with both psse/Pnon-sse tests and Tnon-sse/Tsse for tests 3 and 4, following by a steady increase in speedup after.

4. The speedup is consistent with the fact that speedup for all tests with all array sizes is greater than 1. However, as mentioned above, there's a decrease in speedup for tests 1 and 2, but an increase in speedup for tests 3 and 4 as the array size increases.

5. SSE allows for four floating point operations to happen simultaneously, while the non-sse code only allows for 1 floating point operation to happen at a time. We see that our speedup is greater than 1 because the SSE code can handle more floating-point numbers at a time than non-sse code.

6. Knowing that SSE SIMD is 4-floats at a time, we see a speedup of less than 4 with a small array size, then a speedup greater than 4 for medium size arrays, and finally a speedup of less than 4 with large arrays. This is happening because we are traversing through the array so fast that we're violating temporal coherence with the non-sse code. We're not reusing same values multiple times, instead we use the value, flush the cache line, then at a later point we reload the cache line.

7. Knowing that SSE SIMD is 4-floats at a time, we see a speedup of less than 4 with a small array size, then a speedup of less than 4 for medium size arrays, and finally a speedup greater than 4 with large arrays. This again is happening because we are traversing through the array so fast that we're violating temporal coherence with the non-sse code. With an array size of 1024, we are more than likely getting a lot of cache hits, but as the array size gets bigger, we start to experience more cache misses.