CS 475 - Project 4: Vectorized Array Multiplication and Multiplication/Reduction using SSE

Name: Luke Scovel
Email: scovell@oregonstate.edu
Project Number: 4
Project Name: Vectorized Array Multiplication and Multiplication/Reduction using SSE

Machine Information

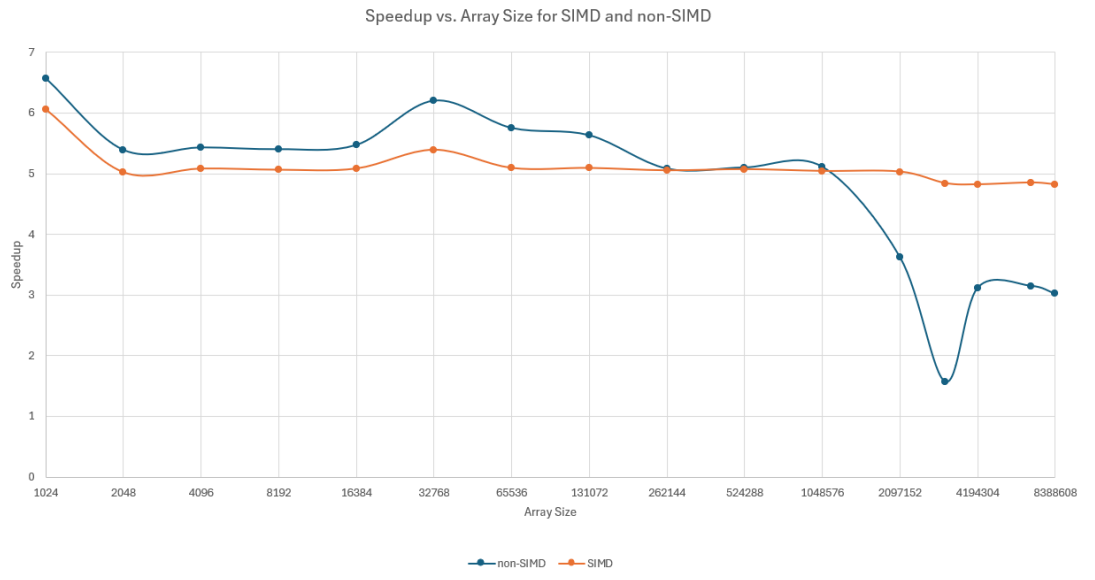
- rabbit.engr.oregonstate.edu
- CentOS Linux release 7.9.2009 (Core)
- g++ compiler

```
LEVEL1_DCACHE_SIZE                32768
LEVEL2_CACHE_SIZE                 262144
LEVEL3_CACHE_SIZE                 20971520
```

Performance Data

bSIMD	ARRAYSIZE	SPEEDUP
non-SIMD	1024	6.57
SIMD	1024	6.06
non-SIMD	2048	5.39
SIMD	2048	5.02
non-SIMD	4096	5.43
SIMD	4096	5.08
non-SIMD	8192	5.4
SIMD	8192	5.06
non-SIMD	16384	5.47
SIMD	16384	5.08
non-SIMD	32768	6.2
SIMD	32768	5.39
non-SIMD	65536	5.75
SIMD	65536	5.09
non-SIMD	131072	5.63
SIMD	131072	5.09
non-SIMD	262144	5.08
SIMD	262144	5.05
non-SIMD	524288	5.1
SIMD	524288	5.07
non-SIMD	1048576	5.11
SIMD	1048576	5.04
non-SIMD	2097152	3.62
SIMD	2097152	5.03
non-SIMD	3145728	1.57
SIMD	3145728	4.84
non-SIMD	4194304	3.12
SIMD	4194304	4.82
non-SIMD	6734976	3.15
SIMD	6734976	4.85
non-SIMD	8388608	3.02
SIMD	8388608	4.82

	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576	2097152	3145728	4194304	6734976	8388608
non-SIMD	6.57	5.39	5.43	5.4	5.47	6.2	5.75	5.63	5.08	5.1	5.11	3.62	1.57	3.12	3.15	3.02
SIMD	6.06	5.02	5.08	5.06	5.08	5.39	5.09	5.09	5.05	5.07	5.04	5.03	4.84	4.82	4.85	4.82



Performance Analysis

- What patterns are you seeing in the speedups?

The speedups generally average around 5x, with the non-SIMD code being about .25 higher until it hits the L2 cache size (65536 * 4 bytes/float = 262144 bytes, 256 KB). After that, the speed up is about the same for both until it hits 1048576 bytes (1 MB), where the non-SIMD code starts to slow down faster than the SIMD code, dipping to 2x before averaging around 3x speedup while SIMD stays around 5x.

- Are they consistent across a variety of array sizes?

SIMD is very consistent across all array sizes, while non-SIMD hits some highs and some lows.

- Why or why not, do you think?

The non-SIMD code is not consistent because it is not optimized for the cache sizes, while SIMD is optimized for the cache sizes. The SIMD code is also more efficient in terms of memory access patterns, which leads to better performance.

What I did

I simply implemented the c++ code required for array multiplication and created a script to run through it and output the results to a csv file.