Geoffrey Corey
CS475 Project 7

- **Test Platform**

  Personal Laptop.

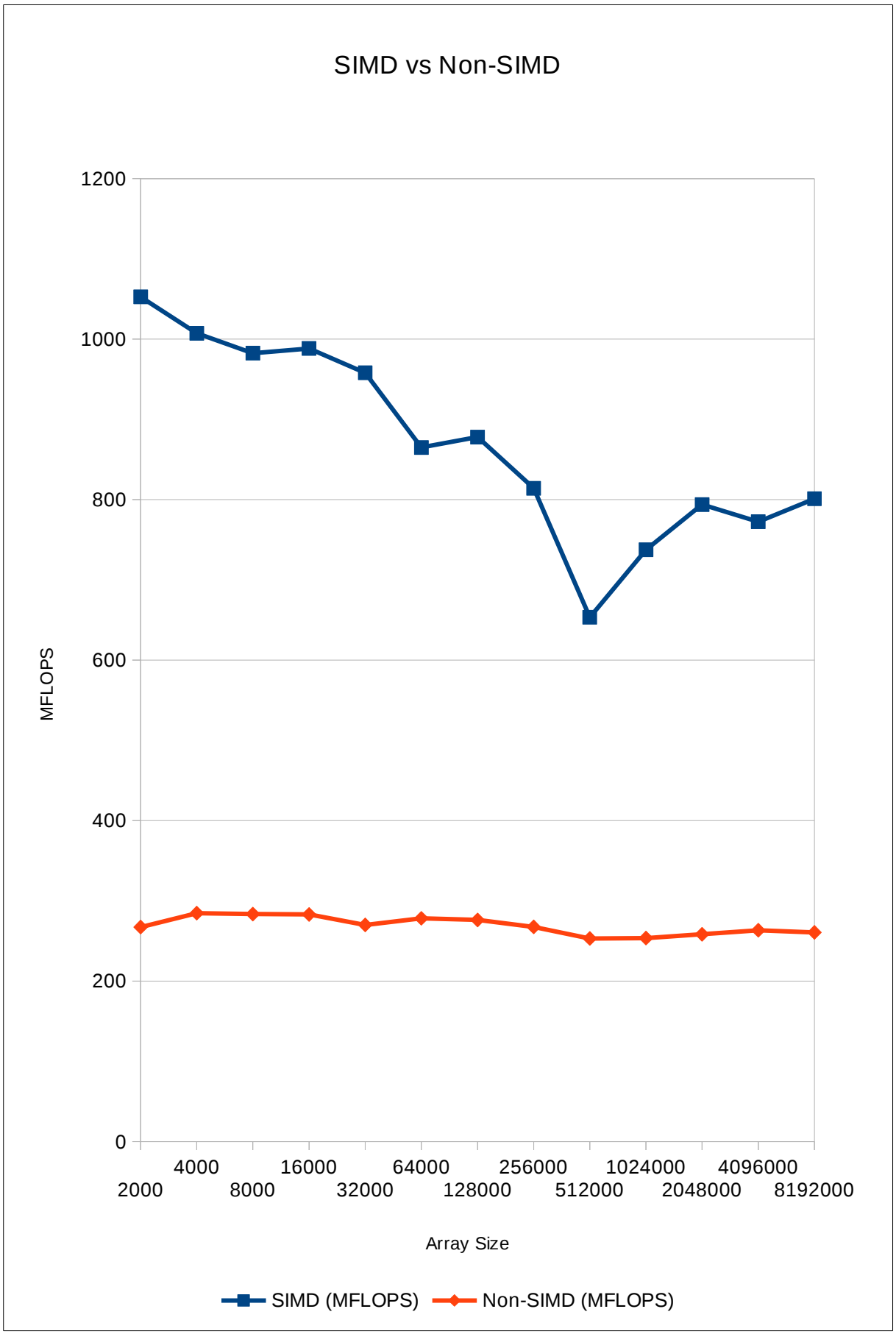
  Core i3-2330m

  Ubuntu 14.04

  Added in a 1 second sleep call to allow system to “quiet down” before running benchmark.

- **Data**

NUM	SIMD (MFLOPS)	Non-SIMD (MFLOPS)
2000	1052.632	267.308
4000	1007.252	284.643
8000	982.499	283.549
16000	988.307	283.092
32000	958.078	270.031
64000	865.006	278.275
128000	877.966	276.153
256000	814.096	267.539
512000	653.445	253.073
1024000	737.488	253.596
2048000	793.898	258.401
4096000	772.497	263.421
8192000	801.002	260.707


SIMD vs Non-SIMD
1200
1000
800
600
400
200
0
MFLOPS
2000
4000
8000
16000
32000
64000
128000
256000
512000
1024000
2048000
4096000
8192000
Array Size
SIMD (MFLOPS)
Non-SIMD (MFLOPS)

- **Speed Patterns**
  It appears that as the array size increased, to overall performance began to decrease for the SIMD code. However, the overall performance for the Non-SIMD code (while much lower) seemed to remain within the same range throughout all of the array sizes.

- **Speedup Patterns**
  The SIMD code had a sharp dive when the array size was 512000 and then increased less sharply back to around 800 MFLOPS. Other than this anomaly, it overall leveled to about 800 MFLOPS as the array size increased towards 8192000.

  For the non-SIMD code, there was not an increasing or decreasing pattern over the size of the input array. The overall speedup pattern remained roughly 0.
- **Consistency**
  The non-SIMD code has very consistent performance while the SIMD code had a decreasing performance trend.
- **Commentary**
  The non-SIMD code being consistent in performance across the varying array sizes is not surprising as the code was using the hardware to its full potential for conventional floating point array multiplication. However, the inconsistency of the SIMD code is very surprising and even lacking a plateau as shown in the in-class slides.