Kailey Cozart
CS-315-1
November 1, 2018

Part 1: *Is there a correlation between any of the above? I might suggest thinking about the maximum threads per block, the warp size and the specifications for your GPU (you are going to have to look those up externally).*

The first thing is that there is a correlation between the number of SMX's and the number of cores. The number of SMX's is equal to the number of cores divided by the warp size. For the GPU, there is a maximum of 1024 threads per block and the warp size is 32. The maximum block dimension is 1024 by 1024, so you have all the threads vertically or horizontally, or a combination of both. However, you cannot have both all vertical and all horizontal.

Part 2: *Perform a number of performance tests on your code and report what happens with various sizes and configurations of threads and blocks. Did you find a correlation at all? Remember the architecture of the GPU is based on warps!*

Grid Size	**Block Size**	**Time (Milliseconds)**
N/1 = 65536	1	.49488
(N+1)/2 = 32768.50000	2	.253152
(N+14)/15 = 4370	15	.254304
(N+15)/16 = 4096.93750	16	.041376
(N+16)/17 = 3856	17	.044288
(N+30)/31 = 2115.03225	31	.031808
(N+31)/32 = 2048.96875	32	.03072
(N+32)/33 = 1986.90909	33	.028256
(N+62)/63 = 1041.23809	63	.018272
(N+63)/64 = 1024.98437	64	.019264
(N+64)/65 = 1009.23076	65	.020512

As the grid size decreases and the block size increases, the time decreases. In this chart, you can kind of see that the multiples of the warp size, 32 and 64, are slightly better than the numbers around them. For example, 32 is faster than 31. Also, 16 is better than 15 and 17. We

would probably be able to see the better performance of multiples of the warp size more clearly if our program was doing more.

I chose the block size based on multiples of the number of warps, which is 32. To find the grid size, I used the formula (N + (Block Size - 1)) / Block Size in order to ensure that all of the data would be given to an SMX if N wasn't perfectly divisible by the Block Size. The time in milliseconds was found using CUDA's event timing functionality.