Original Code	**-O0**	**-O1**	**-O2**	**-O3**
Runtime	204.33 sec	176.96 sec	167.35 sec	158.75 sec
Cache Misses	1,086,233,003	1,081,318,176	1,081,085,741	1,082,005,104
Cache References	22,648,784,843	3,263,331,023	3,261,147,420	3,260,132,609
With best compiler optimization level	**Original**	**Column-major**	**Tiled 16x16**	**Tiled 32x32**
Runtime	158.75 sec	13.36 sec	24.32 sec	23.53 sec
Cache Misses	1,082,005,104	4,084,661	153,356,431	121,793,715
Cache References	3,260,132,609	3,260,423,084	4,399,400,779	4,366,408,701

First off, changing the levels of optimization via the compiler from O0 to O1 greatly decreased the number of cache references and slightly decreased runtime. From O1 through O3 there were slight decreases in cache references but a not insignificant decrease in runtime. Then, still using O3 optimization, changing the program to column-major memory storage for the second matrix made a huge difference in decreasing both cache misses and runtime. Finally, once I implemented tiling for the program, the results depended on what you compare it to. Compared to the original code, it is still a large decrease in both runtime and cache misses. However, when compared to the column-major optimization, there's a slight increase in runtime and cache misses. I'm not sure if that's caused by great code optimization in the column-major version or poor code optimization in my tiling code, I'd probably assume the second, but it's still a major improvement over completely unoptimized code.