Report

Table for three runs using n=8192

mmvec	1	2	3
User Time	151.207017	148.595638	148.534394
System Time	0.312006	0.271992	0.343996
Maximum Resident Size	1050408		

mmnovec	1	2	3
User Time	348.251613	346.941634	344.531697
System Time	0.283999	.372001	.343995
Maximum Resident Size	1050544		

Speed Up	2.32

1) Discussion of your implementation and results, especially why you may not have achieved the theoretical max speedup

To get a good vectorization implementation for the multiplication of the matrices I first tried to tell the compiler that data was aligned using the #pragma before the first loop of the outer loop, however, the compiler tried to vectorize the matrix multiplication, but since data was not aligned It did not improve performance. So, I realized that data should be aligned for the compiler to make a good vectorization and take advantage of Single Instruction Multiple data (SIMD). To improve the performance and make sure data was aligned I took the transpose of the second matrix, so with the transpose the compiler could take advantage of data alignment and perform SIMD. And in addition to transpose I put the #pragma vector aligned before the last inner loop because this is the for loop that would iterate the most, so I it had to take advantage of SIMD instruction. The transpose of the second matrix improved performance by a lot and user and system time improved and were faster. I believe the reason that I did not achieve the theoretical max speed up it's because the operations performing by multiplication both matrices are not fully being transformed into SIMD instructions, and the speed up is slower because of this reason, and since the size of the array is 8192 speed up also depends on how many operations are using SIMD.

2) theoretical maximum speedup due to vectorization and your reasoning:

AVX has instructions that operate on 256 bits, and SIMD allows to do the same operation on several data with SIMD memory is not fetched slowly and the same operation can be done in once. So, the theoretical maximum speed up depends on the instructions made by using SIMD and proportional to vector width. The maximum speedup I can get is 4x due to the CPU, due to vectorization. So, due to vectorization multiple elements are being processed simultaneously instead of one.