AUTOMATIC VECTORIZATION
Bradley Sundsbarm
CSCI551
Assignment 2

TIMES and Speed Increase calculations

mmvec times
231.408
236.491
241.822
241.017

Avg = 237.6845

novec
476.786
470.691
472.339
473.212

Avg = 473.257

*Chose to stop it around 4 samples per test since my times were are relatively close.

Average speed up from vectorization = 237.6845 / 473.257 = 0.5022…
A tad bit above 50% increase in solution speed.

Theoretical maximum speedup due to vectorization is about 50% given the sample trial data I used. This percentage is exponential however and would only increase if was used on a larger sample size than 8192. Note used the code presented in class for the multiplication of the two matrixes, this method results in less miser per inner loop iteration and averages to 0.25 misses per clock cycle.

My testing was done on my Macbook pro 2017 model running Ubunto 18.04 on VMWARE Fusion. Additionally, I closed most processes on both Operating Systems and allocated 8gb of ram to Ubunto. The flags I used for my Ubunto compiler for vectorization was -O2 and -ftree-vectorize. At first I tried just ftree, but for some reason I was not vectorizing my code and it ran about the same times as no flags, so after further investigation I added -O2, which where if you read the Man Pages of the g++ compiler it turns on vectorization. After that adjustment I saw noticeably more speed. Another touch I added was that when I was calculating the floats for the random matrixes, I took the inverse and multiplied vs dividing. Though the increase was not very noticeable it still made for good practice in terms of speed and performance, like you discussed in class.