Lecture 9: OpenMP – Homework

- Use OpenMP (not MPI) to implement GEMM in parallel
  - No need to implement all variants, assume xGEMM("No transpose", "No transpose", M, N, K, $\alpha = -1$, A, B, $\beta = 1$, C)
  - You can make the dimensions “convenient”
    - Divisible by number of threads, cores, ...
    - Divisible by 100 or other favorite integer
  - Make sure though that the matrix dimensions can grow so you can test weak scaling.

- Use OpenMP (not MPI) to implement TRSM in parallel
  - Simplify accordingly in a similar way as you did for GEMM.

- Comment on performance and ease of implementation in comparison to each other and the pthread implementation
  - Things to keep in mind is scalability, productivity, potential for concurrency bugs.
Details on TRSM

- TRSM = Triangular Solve with Matrix
- Reference implementation: http://netlib.org/blas/dtrsm.f
  - If you don’t know where to start then you can select one of the loop nests in the reference implementation and focus on parallelizing the selected code.
  - To simplify parallelization you should assume that the if statements are predetermined. That’s why I allow you to select particular values for SIDE, UPLO TRANSA, and DIAG input parameters.
- This is a pictorial representation of what the TRSM routine does

\[ A \times X = B \] or \[ A \times X = B \]

- The TRSM routine solves \( AX = B \) system where \( A \) is either upper or lower triangular. The solution matrix \( X \) is returned in the space occupied by \( B \).