4.	Fachgebiet Architektur eingebetteter Systeme **Advanced Computer Architectures**	SoSe2018

Assignment start: 26.06.2018 Submission deadline: 16.07.2018

Assignment 4 - OpenMP Multithreading **20 Points**

In the final exercise you will build on the multi-threading concepts acquired in the lectures and more specifically you will get hands on experience using OpenMP to parallelise sequential programs.

OpenMP is parallel programming model for shared-memory multi-processor systems. It allows the programmer to specify a portion of code should that should be executed in parallel using compiler directives. OpenMP implementation of multi-threading uses the fork-join model, whereby a master thread forks a number of slave threads dividing work among them. The threads then run concurrently, with the runtime environment allocating threads to different processors core or hardware threads.

In this assignment we provide you with sequential code implementing the LU decomposition algorithm, `https://en.wikipedia.org/wiki/LU_decomposition`. Your task is to parallelize this sequential program with the help of OpenMP. The code tarball is available on the course ISIS page. You can perform the experiments on your local machine but for consistency we will be checking your assignments on an Intel(R) Xeon(R) CPU E5-2680 v3 with 24 cores (48 threads) using GCC compiler version 5.4.0 running on Ubuntu1 16.04.

The `lud_seq.cpp` file contains our sequential baseline implementation. As part of this assignment, you have to perform the following:

- Modify the `lud_par.cpp` file by parallelizing the sequential code using OpenMP directives.
- Modify the `lud_opt.cpp` file to produce your best optimized version. You can make any changes to the the algorithm, use any OpenMP directives, number of threads etc ..
- Produce a graph figure by varying the number of OpenMP threads and showing the speedup of your `lud_opt.cpp` versus `lud_seq.cpp`. Use the 1024.dat input file.
- Write a small report analyzing the produced figure. Also include brief explanations of the modifications you made to both `lud_par.cpp` and `lud_opt.cpp`.

You can test your implementation on the same machine that we will use for grading your submission. The machine is accessible via ssh using: *ssh username@cell1.aes.tu-berlin.de*. To copy the source tar file for example you can use: *scp aca-openmp.tar username@cell1.aes.tu-berlin.de:/~*

Your deliverable should only contains 3 files `lud_par.cpp`, `lud_opt.cpp` and your brief report `report.pdf`. The grade for this exercise will be determined by the performance of your implementations (15 points) and your explanation (5 points).