# Introduction to Parallel and Distributed Computing Exercise 2 (May 16)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

The result is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file which contains

- a single PDF (`.pdf`) file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - a section with the source code of the program benchmarked, the output of the compiler, and an explanation of the output,
  - a section with the raw data of the benchmarks,
  - a section with a summary table and graphical diagrams of the benchmarks.
- the source (`.c`) file(s) of the programs.

## Exercise 2: Shared Memory Programming with OpenMP

The goal of this exercise is to solve the “all pairs shortest paths problem” presented in Exercise 1 with OpenMP in two versions.

### Verson 1 (65P): OpenMP Loop Parallelism

Parallelize the program by annotating the outermost loop of the “squaring” operation with OpenMP pragma `parallel for` such that this loop gets executed in parallel by distributing its iteration range among multiple threads (do not forget to “privatize” variables whenever necessary). Compile the program with options `-O3 -openmp -openmp-report2` and explain the compilation output. Experiment with at least two different scheduling strategies (clause `schedule(runtime)`, environment variable `OMP_SCHEDULE`) and choose the better one for your benchmarks (describe your experiments and justify your choice).

### Version 2 (35P): OpenMP Task Parallelism

Parallelize the program by rewriting the “squaring” operation to a recursive function that processes the iteration range $i \in [begin, end[$ (left-closed, right-open interval) of the index variable $i$ of the outermost loop. This function performs the work of $n = end - begin$ iterations of the loop in a divide and conquer fashion:

- if $n < 1$, no work is performed.
- if $n = 1$, row $i := begin$ is processed.
- if $n > 1$, the iteration range is split into two halves $[begin, mid[$ and $[mid, end[$ for $mid = \lfloor (begin + end)/2 \rfloor$ which are processed recursively (in parallel).

Implement this algorithm using the OpenMP `task` pragma (see the last slide of slide set “OpenMP”, which provides a sketch of above algorithm). Compile the program as in the first version and explain the compilation output (if any).

Tasks are scheduled dynamically; there is no need to explicitly deal with scheduling. However, you may append to each task pragma a clause `if (`$m$ `>=` $M$`)` where $m$ is the number of rows to be processed by the task (i.e., $m = mid - begin$ or $m = end - mid$); in this case the task will only be executed in parallel if $m$ exceeds the given threshold $M$. Please experiment with some values for $M$ and report your best choice (which may also be to not use the `if` clause at all).

### Benchmarking

Benchmark each version of the program as in Exercise 1 and present the same results (execution times, absolute speedups and efficiencies) as in Exercise 1.