Algorithm Engineering Lab Assignment 10

Brian Zahoransky (brian.zahoransky@uni-jena.de)

February 15, 2021

1. Name and explain some useful compiler flags during development.

The **"-Wall"** flag enables all compiler's warning messages such as declared values which are never used. These warnings will be shown during the compiling process.

The **"-g"** flag generates debug information which may be used by a debugger and profiler. Debuggers allow going through the program step-wisely that might simplify the debugging process. Profilers are checking the program for specific bugs by themself. The intel profiler, for example, provides functions to find deadlocks and race conditions, among others.

The following "-fsantize=[arg]" flags are not available for the intel compiler.

The **"-fsantize=address"** flag looks after out-of-bound and use-after-free access, and memory leaks. Out-of-bound access means memory access out of the range of the allocated memory. Use-after-free access occurs when a program tries to access deallocated memory. Memory leaks are caused by not deallocating memory at the end of the program.

The **"-fsanatize=undefined"** flag is searching after undefined behaviour at run-time. An example in the lecture was an integer overflow caused by increasing a value multiple times by adding random numbers.

2. How could Intel oneAPI help you write better programs?

The Intel one API contains various subprograms for tuning programs. During the lecture, the Intel inspector was discussed as well as the Intel VTune Amplifier. The **Intel inspector** provides functionality to perform memory, threading and customisable error-analyses. The **Intel VTune Amplifier** enables executing a performance analysis. Based on that, the programmer can start more specific analyses on several aspects of the program. These specific analyses look at the aspects: algorithm, parallelism, accelerators, micro-architecture, input/output, and platform analyses.

3. What can we learn from the following quote? Premature optimization is the root of all evil (Donald Knuth).

The quote from Donald Knuth says that optimization should follow the analysis. Blind optimization without analyzing where are the bottlenecks is not reasonable. First, a programmer should implement a stable algorithm which solves the problem adequately. Therefore, he can use debugger and profiler. Second, the developer can use performance analysis tools for finding critical sections and bottlenecks. Third, he/she can implement optimizations. Fourth, the program should be checked if it still solves the problem. Does this apply, step two, three, and four can be repeated till the program reaches the desired performance.