Abstract of presentation of lecture

Course: **Compiler Construction (VYPe)**
Variant: **Compiler Design in C – Chapter 7 (Optimization Strategies)**
Authors: Odaloš Matej, Bc. xodalo00@stud.fit.vutbr.cz
Polesný Ondřej, Bc. xpoles01@stud.fit.vutbr.cz

The presentation of optimization strategies will show optimization techniques from different categories based on where the optimization takes place. We will present the basics of each technique (based on our selection as the time allocated for the presentation is limited) and show its results on our own examples. There are three categories in which optimizations can be easily divided: parser optimizations, linear peephole optimizations and structural optimizations. We will present and describe each category.

For parser optimizations *instrinsic functions* should be mentioned as they are managed by the parser itself and are very simple.

For *linear peephole optimizations* we point out that parser cannot do them by itself, because examining several blocks of code is necessary. This is what the word „peephole“ represents. We will mention types of these optimizations, such as *strength reduction* (replacing an operation with a more efficient one yielding in the same result), *constant folding and propagation* (reduction of counting constants and replacing variables in right side of assignments with constants in case base variable does not change in the process). Finally we will mention *dead variables and dead code*, which manages removal of variables and code, which will never be processed. We may mention a few things to know about *hardware problems*.

For *structural optimizations*, we will mention how they differ from the previous techniques and what kinds of problems they have to face. This part of presentation will explain two intermediate languages – *postfix* and *reverse-polish notation* - and present examples of their application. Finally we will present a few techniques of structural optimization such as *Common-Subexpression Elimination*, *Register Allocation*, *Lifetime Analysis*, etc.