Machine code disassembling is a process of transforming binary machine code into assembly code. The main purpose of this process is to help people to understand the purpose of the program without knowing its source code. Unfortunately, the machine code produced by compilers is quite hard to read due to numerous optimizations applied to it. One substantially problematic optimization is instruction scheduling which mangles instruction order to increase final performance.

The goal of this thesis is to implement a disassembler capable of reordering individual instructions. This would allow the user to restructure the machine code into a more readable form. To provide such functionality, the disassembler has to be able to understand the meaning of machine code instructions. For this reason, we will design a platform-independent internal representation of machine code and we will translate any machine code into it. This representation will be then used to analyze dependencies between instructions which can be further used in instruction reordering algorithm. At the very end, we will discuss the possibility of platform-independent program emulation based on internal disassembler representation.