JOHANNES KEPLER
UNIVERSITÄT LINZ

o.Univ.-Prof. Dr.
Hanspeter Mössenböck
Institute for System Software

T +43 732 2468 4340
F +43 732 2468 4345
hanspeter.moessenboeck@jku.at

Secretary:
Karin Gusenbauer
Ext 4342
karin.gusenbauer@jku.at

Master's Thesis

A Generic Machine Code Instrumentation Library

Student:	**Alexander Voglsperger (12005568)**
Advisor:	Prof. Hanspeter Mössenböck
Co-Advisors:	DI Peter Feichtiger, DI Michael Obermüller
Begin:	1.10.2024

Binary instrumentation is a technique where the machine code of a running process is modified so it can be analyzed in some way. By attaching so-called hooks to specific pieces of the program, arbitrary code may be run when they are executed. This is particularly interesting for use cases such as performance measurements or monitoring, e.g. by hooking the entry points of specific functions of interest.

The goal of this thesis is to implement a generic C++ library that allows performing this kind of modification on arbitrary function entry points to attach a hook (function pointer or other callable object) provided by the user. The hook should receive all the necessary context for inspecting the program's state, such as register contents and stack pointer. Interpreting the context (e.g. reading local variables) is outside the library's scope, but rather the hook's responsibility.

One straightforward way to implement this kind of instrumentation is to use function trampolines. To instrument a piece of code, its first few machine instructions are replaced by a jump instruction to a so-called trampoline. The trampoline code takes care of preserving the necessary context, calling the actual hook, restoring context, and then executing the original instructions before returning execution to the instrumented function. To achieve this on x86, the library will have to be capable of copying machine code in memory while preserving its semantics.

The library implemented in this thesis should target x86, ideally both 32 and 64 bit. Since x86 is only one of several planned target architectures, the library should be easily extensible to support other architectures such as aarch64. The library need only support the architecture it's compiled for, i.e. running on x86_64 only needs to support hooking of x86_64 functions.

An existing library that might be of interest is Microsoft Detours [1], which implements function trampolines (and the necessary code copying) for Windows. As for a disassembler, it might be possible to use Capstone [2] rather than implementing everything from scratch.

The task requires a solid understanding of machine code and assembly language, and a commitment to implementing a user-friendly and extensible library that can be used by programmers for their daily work.

The work's progress should be discussed with the advisor at least every 2 weeks. Please follow the guidelines of the Institute for System Software when preparing the written thesis. The deadline for submitting the written thesis is 30.09.2025.

References:
[1] https://github.com/microsoft/detours
[2] https://www.capstone-engine.org