JOHANNES KEPLER
UNIVERSITY LINZ

DI Manuel Rigger, M.Phil.
Institute for System Software

T +43 732 2468 4356
F +43 732 2468 4345
manuel.rigger@jku.at

Secretary:
Birgit Kranzl
Ext 4341
birgit.kranzl@jku.at

Master's Thesis / Project in Software Engineering

A Truffle-based Interpreter for x86 Binary Code

Student:	Daniel Pekarek
Advisor:	DI Manuel Rigger, M.Phil.
	DI David Leopoldseder
Start date:	01.03.2018

GraalVM is a virtual machine that supports the execution of various languages including Ruby, JavaScript, R, and LLVM-based languages such as C/C++ [1]. Its individual language implementations are based on Truffle, a language implementation framework [2].

To call precompiled binaries, where no source code is available, GraalVM provides a native function interface, called Truffle Native Function Interface (Truffle NFI). However, the call overhead of Truffle NFI is significant, since data that crosses the language boundaries needs to be converted.

To provide an efficient alternative to Truffle NFI, a Truffle-based x86-64 interpreter should be implemented. Due to Truffle’s efficient language interoperability, the overhead of calling precompiled libraries is avoided. The implementation of the interpreter requires mapping the semantics of individual x86-64 instructions to executable Truffle nodes. To implement unstructured control flow, Sulong’s bytecode-interpreter-like approach should be used [3]. Further, both an abstraction of memory and a subset of system calls need to be implemented.

Compilation on GraalVM relies on the assumption that function boundaries are explicitly represented. However, functions are no longer present on the machine code level. Thus, a heuristic should be developed that enables compilation nevertheless (e.g., by heuristically identifying blocks of code as functions).

The scope of this thesis is as follows:

- Implementation of a Truffle-based x86-64 interpreter that can execute at least 2 SPEC CINT2006 benchmarks and 5 benchmarks of the Computer Language Benchmark Game.

- Supporting compilation of the generated Truffle ASTs by the GraalVM.

- An evaluation of the interpreter’s completeness and peak performance.

Explicit non-goals are:

- Completeness with respect to the x86 instruction set or system calls.

- Peak performance that is competitive with state-of-the-art approaches.

The work's progress should be discussed with the supervisor at least every 2 weeks. Please note the guidelines of the Institute for System Software when preparing the written thesis.

[1] Thomas Würthinger, Christian Wimmer, Christian Humer, Andreas Wöß, Lukas Stadler, Chris Seaton, Gilles Duboscq, Doug Simon, and Matthias Grimmer. 2017. Practical partial evaluation for high-performance dynamic language runtimes. SIGPLAN Not. 52, 6 (June 2017), 662-676. DOI: https://doi.org/10.1145/3140587.3062381

[2] Thomas Würthinger, Andreas Wöß, Lukas Stadler, Gilles Duboscq, Doug Simon, and Christian Wimmer. 2012. Self-optimizing AST interpreters. In Proceedings of the 8th symposium on Dynamic languages (DLS '12). ACM, New York, NY, USA, 73-82. DOI=http://dx.doi.org/10.1145/2384577.2384587

[3] Manuel Rigger, Matthias Grimmer, Christian Wimmer, Thomas Würthinger, and Hanspeter Mössenböck. 2016. Bringing low-level languages to the JVM: efficient execution of LLVM IR on Truffle. In Proceedings of the 8th International Workshop on Virtual Machines and Intermediate Languages (VMIL 2016). ACM, New York, NY, USA, 6-15. DOI: https://doi.org/10.1145/2998415.2998416