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

Bachelor's Thesis

Execution of Rust Programs on Sulong

Student: Arif Celik
SKZ/Matr.Nr.: 521 / 01455412
Email: k1455412@students.jku.at

Advisor: DI Manuel Rigger, M.Phil.
DI Dr. Matthias Grimmer

Start date: -

Sulong [1] is a high-performance LLVM IR interpreter built on the JVM using the Truffle language implementation framework. Until now, implementation efforts focused on supporting the execution of C/C++ programs compiled to LLVM IR using the LLVM front end Clang.

Rust [2] is a systems programming language that provides strong guarantees about isolation, concurrency and memory safety which are enforced at compile time. Its memory management model prevents data races, that null pointers are dereferenced, and that dangling pointers are accessed.

The goal of this thesis is to add support for running Rust on Sulong by using the Rust compiler to generate LLVM IR from the Rust program. We expect that the generated LLVM IR differs from the LLVM IR generated by Clang for C/C++ programs. Missing features and bugs in Sulong and Truffle should be investigated and either implemented, fixed, or documented. Further, support for common functionality provided by the the Rust Standard Library should be added (e.g., by intrinsifying Rust's core functions or by compiling the standard library to LLVM IR).

Overall, Rust's goals of safety are similar to those of Safe Sulong, an extension of Sulong that adds memory safety features [3]. However, Rust also allows to write unsafe code segments (called “Unsafe Rust”), that operate with raw pointers, where Rust's strong memory safety guarantees no longer not hold. As such code fragments could be executed by Safe Sulong, it should be discussed how lost memory safety could be regained at runtime for unsafe Rust.

The scope of this thesis is as follows:

- Adding support for running small to middle-sized Rust programs on Sulong that use common standard library functions.
- Extending Sulong’s test suite to cover Rust test cases.
- Comparing the performance of Sulong running Rust with machine code that was directly generated by the Rust compiler.

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Explicit non-goals are:

- Completeness with respect to the Rust language specification and the Rust Standard Library.
- Reaching an execution speed that is close to machine code generated by the Rust compiler.

[1] 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

[2] https://www.rust-lang.org/en-US/

[3] Rigger, M.; Schatz, R.; Mayrhofer, R.; Grimmer, M.; Mössenböck, H.: Sulong, and Thanks For All the Bugs: Finding Errors in C Programs by Abstracting from the Native Execution Model Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18), Williamsburg, VA, USA, March 24 - 28, 2018 (accepted for publication).