Formally Verified Advanced Optimizations for RISC-V

David Monniaux, Sylvain Boulmé, Léo Gourdin*
Université Grenoble Alpes, CNRS, Grenoble INP, Verimag
https://gricad-gitlab.univ-grenoble-alpes.fr/certicompil/Chamois-CompCert

Avoiding Bugs in GCC & LLVM (cf. [1])?

CompCert¹ solution (ACM Software System Award 2021): the first formally verified (= machine-checked mathematical proof of correctness) compiler optimizing safety-critical software [2, 3].

https://www.absint.com/solutions/

Our General Purpose Translation Validator

The oracle takes source program S and yields its optimized version T along with a certificate. A verified symbolic execution interpreter then ensures semantic preservation, and aborts compilation in case of failure.

S: Source program
Interpreter
Oracle
T : Target Program
Verifies
Specifies
Certificate
Trusted (Coq)
Untrusted (OCaml)

Defensive Symbolic Simulation

For each pair of loop-free blocks $(B_{src} \in S, B_{tgt} \in T)$, we compose the symbolic states $(s_{src}, s_{tgt})$ resulting from their symbolic execution.

The certificate contains invariants propagating information between blocks.

Compile Times That Scale

Validating the Lazy Code Transformations Oracle


• Search for reducible multiplicative operators;
• Based on data-flow analyses performed by an OCaml oracle;
• Supports decomposed patterns like a left shift + an addition;

Optimizing in two steps:
1. Lifting the multiplication out of the loop;
2. Inserting compensation code in the loop body;

Optimized Generated Code That You Can Trust

Comparing w.r.t. Official CompCert over five test suites
Percentage gain in execution time, higher is better

References


Git Repo

This work is supported by the French National Research Agency in the framework of the ‘France 2030’ program (AMR). RISC-V Compiler contributed by the Labex M2RhôneAlpes (ANR-11-LABX-0063).