

HONGYU CHEN

✉ hongyc4@uci.edu  [Hongyu Chen](#)  github.com/hongyu-dev

Education

Ph.D. in Computer Science, University of California, Irvine

Sep 2023 - Mar 2027 (Expected)

Research in binary translation.

Advisor: [Professor Michael Franz](#)

M.S. in Computer Science, University of California, Irvine

Sep 2021 - Jun 2023

B.S. in Computer Science, University of California, Irvine

Sep 2016 - Dec 2020

Skills

Languages: C/C++, Assembly(x86/64, AArch64), Python, Rust, Linker Scripts, Bluespec, SQL, Shell Scripting

Frameworks/Tools: LLVM, MLIR, Clang, GCC, Docker, Ghidra, CMake, Ninja, Valgrind, Maven, Git, TensorFlow

Publication

[Deterministic Fully-Static Whole-Binary Translation without Heuristics](#)

To appear in ACM Transactions on Architecture and Code Optimization (TACO)

Selected Experience

GPU Code Generation

Jun 2026 - Sep 2026

Intern, Backend Compiler Team, NVIDIA

Binary Ninja Decompiler Infrastructure Enhancement

Jun 2025 - Sep 2025

Intern, Vector 35, Melbourne, FL

- Implemented Xtensa register windowing with a window rotation system to analyze calls with overlapping register sets.
- Designed the ILTransparentCopy attribute, letting SSA analyses trace data flow through architectural register copies.
- Fixed the ELF loader's section-to-segment mapping so NOTE sections no longer produce an incorrect memory layout.
- Added constant folding for double-precision instructions in the IR, improving decompiled code quality.
- Architected a scalable calling convention model built on register classes, enabling TriCore support and removing the fallback to heuristics for parameter identification.

[Improve Linker Script Handling in LLD](#)

Jun 2024 - Sep 2024

Software Engineering Intern, Google LLC, San Jose, CA

- Enhanced the lexer to handle complex linker scripts more efficiently, reducing errors and improving maintainability.
- Replaced lambda-based expression handling with structured expression types, improving potential performance, reducing memory overhead, and enhancing debuggability.
- Collaborated with upstream maintainers to ensure smooth integration of changes, addressing bottlenecks in the review and feedback process.

[Intel VTune Support to LLVM JITLink](#)

Nov 2023 - Feb 2024

- Integrated Intel VTune profiler support into JITLink for enhanced performance analysis.
- Enabled profiling of JIT-compiled code using VTune's API.
- Developed a test based on existing LLVM JIT profiling tools.

SMPL Compiler

Jan 2023 - Mar 2023

- Developed an optimizing compiler in C++ for the SMPL programming language.
- Constructed a recursive descent parser that generates a Static Single Assignment (SSA)-based IR.
- Implemented copy propagation, common subexpression elimination, and redundant array load elimination over SSA.
- Developed a global register allocator that tracks live ranges, builds interference graphs, and performs graph coloring.

Microbenchmark for Manycore system HammerBlade

Jun 2022 - Sep 2022

- Explored and designed a runtime system for HammerBlade to facilitate swift task migration and reduce memory-to-icache transfers.
- Implemented a [microbenchmark](#) in C to assess the trade-off between cache misses and core hopping.
- Profiled and compared the performance difference between the normal execution model and the core hopping model.

Service

Student Volunteer Co-Chair, [PLDI 2026](#) and [PLDI 2027](#)

Artifact Evaluation Committee, [EuroSys 2026](#)

Student Volunteer, PLDI 2022 and PLDI 2023