Farhan Saif

+1-312-823-5137 | fsaif@uic.edu
linkedin.com/in/farhansaiif | github.com/overlorde
Chicago, Illinois, USA

Education

University of Illinois Chicago — Expected: Dec 2026
Master of Science in Computer Science — GPA: 3.5/4.0 — *Chicago, Illinois, USA*
- Coursework: Compiler Design, Advanced Topics in Concurrent Computing Systems, Principles of Concurrent Programming

Islamic University of Technology — May 2023
Bachelor of Science in Computer Science and Engineering — *Dhaka, Bangladesh*

Experience

University of Illinois Chicago — May 2025 – Present
Graduate Research Assistant
- Built a functioning record/replay and multi-version execution system for stock CPython with no bytecode modification
- Extracted CPython's network unit tests into server and client pairs to broaden coverage of network system calls
- Identified CPython methods that issue system calls by combining `strace` with signal based tracing
- Implemented runtime method instrumentation by exploiting a loophole in `mappingproxy` configuration
- Worked on a novel linearizability violation testing tool aimed at free threaded CPython
- Wrote automation gluing together the tool's randomized testing pipeline
- Benchmarked the tool against CPython 3.14t, Jython, and IronPython
- Found two segfaults in CPython's free threaded pickle and authored the fix merged into CPython as #146470

University of Illinois Chicago — August 2024 – May 2025
Graduate Teaching Assistant
- Teaching Assistant for Machine Organization and Systems Programming
- Organized lab demonstrations and discussion groups, and held office hours

Shikkha — November 2022 – August 2023
Software Engineer — *Dhaka, Bangladesh*
- Built PHP and Laravel 8 backend services covering auth, course management, content delivery and payments for an EdTech platform, shipping to AWS through Jenkins CI/CD

Publications & Posters

- Co-authored paper on linearizability violation finding in free-threaded Python — under submission
- Poster: Performance Evaluation of Biased Reference Counting in Free-threaded Python, GCSR 2026 Workshop, Northwestern University

Skills

- **Languages:** C, C++, Python, PHP, Bash, x86 Assembly, TypeScript/JavaScript
- **Compilers & Runtimes:** LLVM, Yacc/Bison, asmjit, CPython internals, interpreters, multithreading/concurrency
- **Tools:** GDB, Valgrind, Linux, Git, Jenkins, AWS, Nginx, CI/CD

Projects

pyrr-mvx – Multi-Version Execution for Free-Threaded Python *[Python, CPython C internals]*
- Runs a leader and a follower process concurrently, the leader executing real syscalls and streaming results over a lock free SPSC shared memory ring, the follower consuming them for byte equivalent execution with all I/O removed
- Patched immutable CPython module dicts and type caches via a "Leaker trick" exploiting `__eq__` and `PyType_Modified` invalidation
- Implemented follower takeover on leader death, shipping live connections and unaccepted listeners over `SCM_RIGHTS` fd transport so clients never see a break or a rebind
- Passes 263 of 397 CPython test suite files as live leader and follower pairs under automated divergence detection

py-svf-analysis – Static Value-Flow Analysis for Python *[Python]*
- Lowers Python source to a custom SSA form IR and runs field sensitive Andersen points-to over it, building the call graph on the fly because every Python call is indirect
- Layers memory SSA with strong updates on top and builds a sparse value flow graph over heap locations, so taint questions become thin slices that see through aliases and stop at sanitizers
- Analyzes whole libraries without executing them, via a typed harness and an `extapi` style effects database
- Validated by 123 regression tests, differential runs against PyCG, and a dynamic trace inclusion check

CE – A C-like Compiler with an LLVM Backend *[C++, LLVM]*
- Wrote the LLVM backend with the C++ `IRBuilder` API, lowering the typed AST to LLVM IR and native object files
- Designed a nominal type system with full type erasure so every source type compiles to one `i64` word, with phantom typed heap primitives `alloc<T>` and `ptr<T>` and subsumption subtyping
- Implemented Hindley Milner type inference with link based unification to support polymorphic functions

- Added Tofte Talpin style region analysis to reclaim heap memory without a garbage collector or manual frees