# Viranchee Lotia

hire@viranchee.com | linkedin.com/in/viranchee | github.com/viranchee | +1 7438372837

EDUCATION:

**North Carolina State University** - **MS**, Computer Engineering (CGPA: 3.6) Jan 2022 - Dec 2023

**Mumbai University** - **BE**, Electronics Engineering Aug 2014 - May 2017

WORK EXPERIENCE:

**Passive Logic** - Compiler Intern, Utah (Remote) Aug 2023 - Dec 2023

- Generating code coverage reports for differentiable part of the Swift compiler [link].
- Automated deploying coverage publicly via CI/CD: Github Actions and Cloudflare Pages CI [link].

**Qualcomm** - Software Engineer Intern, San Diego May 2023 - Aug 2023

- Modified the compiler, linker, and heap allocator algorithm to add ASAN (Address sanitizer) support for chipsets.
- Performed crash-dump and live, JTAG, memory & stack trace debugging via Trace32, and gdb.
- Analyzed and cut project build time by 30% by optimizing a symbol searching algorithm for an ELF file.

**Multiple firms** - iOS Software Engineer, Mumbai Nov 2018 - Oct 2021

- Shipped Popviewers, boosted Zalora's app-engagement by 30% using A/B testing, UIKit, and Jenkins.

**Earth Energy EV** - Embedded Engineer, Mumbai Jun 2017 - Oct 2018

- Programmed electric bike ECU and BMS peripherals over I2C, SPI, UART, USB.

PROJECTS:

**Autodidax** Understanding JAX core mechanisms: May 2024

- Implementing simplified features of the DL framework JAX including JVP, VJP, Vmap, Jaxpr, JIT tracing.

**Compiler induced Fault Tolerance**: Recreated a 2005 paper: Apr 2024

- Implemented middle-end code transformations which improves system reliability through code replication.
- Combined static and dynamic analysis of compute value correctness and cfg branch correctness at runtime.

**Compiler Optimizations** (**TVM**, **MLIR**, **LLVM** passes and analysis): Mar 2022 - present

- Tuned ResNet-50 v2 using TVM Compiler optimized for Apple M1, achieving a speedup of 2.5x on inference.
- MLIR pass written for constant folding 2 shift lefts, for the arithmetic dialect using OpRewritePattern.
- LLVM Implemented Code optimization techniques: CSE, Dead Code removal, Instruction simplification, Load hoisting, duplicate store removal pass over LLVM IR. ~20% instr. removed with a leading M2R pass.
- Implemented a Loop Invariant Code Motion (LICM) pass. Moved 0.2 - 5 instr. on avg. outside the loop using LICM for LLVM benchmarks, with net reduction of 1238 instr.
- Built Availability analysis, Value numbering, Constant propagation passes for an academic compiler infrastructure.
- Meta-programming feature for Swift to enable private properties in Swift protocols [link].

**Bitwise Processing Language** (**LLVM Front end**, Docker): Feb 2022

- Implemented LLVM front end conversion from a simple PL specification to LLVM SSA using Flex and Bison
- Stood in the top 10 rankings of the class, for least amount of LLVM IR instructions generated.

**GPGPU Simulator** (C++): Jan 2023 - May 2023

- Hand wrote GPU kernel for quantum gate simulation, utilizing CUDA shared memory.
- Enhanced GPGPU simulator by bypassing L1 cache for Cache Unfriendly benchmarks using Address frequency heuristics, achieving +12% on average improvement on IPC.

**Dynamic Instruction Scheduling for out-of-order Superscalar processor** (C++): Nov 2022

- Developed a simulator for 9 stage OOO superscalar pipeline processor for RISC-V ISA.
- Analyzed the effect on IPC by varying ROB size, Issue Queue size and width of the instructions to be fetched.

**CPU Simulators** (C++): Jan 2022 - Dec 2022

- Cache & Memory hierarchy, Branch Prediction (bimodal, gshare), Cache coherence protocols (MSI, MESI, Dragon)

**Xinu Operating System** (C, Rust, QEMU, x86 Assembly) Aug 2022 - Dec 2022

- Implemented Exponential Distribution scheduler and Linux-like scheduler; Wrote concurrent reader, single writer locks
- OS in Rust + QEMU with Paging, UART, Heap allocator design, Exception handling.

**Languages & Frameworks**: C++, C, Python, Swift; MLIR, LLVM, PyTorch, JAX; OpenMP, MPI, CUDA; pthreads, Metal

**Domain Knowledge**: Compilers, Parallel Computing, Machine Learning, GPU / CPU Architecture,