# Uzair Nawaz

512-244-3928 | uzair.nawaz@utexas.edu | github.com/uzairnawaz

## EDUCATION

**University of Texas at Austin**                                                                 Austin, TX
*B.S. Computer Science Honors (Turing Scholar), GPA: 3.95/4.00*                    *Graduating Spring 2027*
Courses:  Honors Algorithms, Honors Computer Security, Honors Operating Systems, Programming for Performance, Honors Computer Architecture, Data Structures, Linear Algebra, Probability and Statistics, Discrete Math

## TECHNICAL SKILLS

**Languages**: C/C++, Python, Java, x86/ARM/RISC-V Assembly, Rust, Verilog, Bash, JavaScript, HTML/CSS, SQL
**Tools**: Git, CMake, Makefile, GDB, JIRA, Confluence, Unity Game Engine, Arduino Microcontroller

## EXPERIENCE

**Google**                                                                                     May 2025 – Aug 2025
*Associate Software Developer Intern - Compilers, Runtimes, and Toolchains Team*          *Sunnyvale, CA*
- Contributed to LLVM-libc, an open-source standard library for C which is part of The LLVM Project.
- Designed, implemented, and tested a system to perform conversion between character encodings in **C++**.
- Contributed to the pthreads implementation, designing a reusable barrier for thread synchronization.

**The University of Texas at Austin**                                                          Jan 2025 – May 2025
*Undergraduate Teaching Assistant (CS 439: Operating Systems)*                              *Austin, TX*
- Planned, graded, and answered questions related to programming assignments and quizzes for about 65 students.

**Keysight Technologies**                                                                      May 2024 – Aug 2024
*Network Visibility Software Intern*                                                          *Austin, TX*
- Developed a staged upgrade process for loading new software on network switches. Allowed network switches to remain online for a large part of the software upgrade process, decreasing downtime by about 40%.
- Reorganized one-shot **Bash** upgrade scripts into logical stages that could be run independently.
- Improved existing web GUI using ExtJS to display software installation status and future actions.
- Implemented new features to **Java** REST API, allowing users to trigger specific stages of the installation process.

**Institute for Computing in Research**                                                        July 2022 – Aug 2022
*Research Intern*                                                                            *Austin, TX*
- Worked with a mentor to create an n-body simulation in **Python** to predict the motion of gravitating particles.
- Used matplotlib library to animate simulations either two- or three-dimensionally.
- Performed benchmarks and generated plots to compare accuracy/efficiency of algorithms (Barnes-Hut vs $N^2$).

## PROJECTS

**LDOS ML Compiler Research Project** | *Python, PyTorch, Triton, C, Linux Kernel*
- Developed a process to compile PyTorch models using Triton and perform inference on them within the kernel.
- Increased the feasability of using ML-based OS policies instead of the existing heuristic approaches.

**Chess Engine** | *C++, Google Test*
- Wrote a UCI-compatible chess engine that applies heuristics and alpha-beta pruning to choose the best move.
- Represented the board as a collection of bitboards (64-bit integers with each bit representing a square), computing moves through efficient bitwise operations. Utilized Google Test framework to verify move generation.

**Object-Oriented FUN Compiler** | *C++, ARM Assembly, Makefile*
- Built compiler to translate a toy language to ARM assembly. Supports conditionals, loops, functions, and classes.
- Implemented compile-time optimizations such as constant folding and tail call elimination by constructing an AST.

**Pipelined Processer** | *SystemVerilog*
- Designed a 6-staged pipelined processor with an ISA that included branches, memory read/writes (including misaligned addresses), and arithmetic. Implemented a branch predictor to reduce pipeline flushes.

**Network Card Driver** | *C++*
- Implemented a driver for the E1000 family of network cards. Supported both transmitting and receiving.

**Chip 8 Emulator** | *C++, SFML*
- Created a Chip 8 interpreter. Supports arithmetic, memory read/write, branch, and graphics instructions.