

Jonathon Sonneborn

Ann Arbor, MI | (269) 601-2900

sonnejm@umich.edu | linkedin.com/in/sonnejm | github.com/JoltedJon

EDUCATION

University of Michigan, College of Engineering

B.S.E. in Computer Science, Minor in Electrical Engineering

Ann Arbor, MI

Expected Graduation: April 2024

TECHNICAL SKILLS

Languages: C++, SystemVerilog, Rust, Scheme, Java, JavaScript, HTML, SQL, Bash, Python, RISC-V, ARM, x86

Tools: Git, Linux, Docker, SQLite3, Make, REST API

Frameworks: Ncurses, React, Flask, Boost

Classes: Operating Systems, Web Systems, Database Management, Computer Architecture, Computer Security, Data Structures and Algorithms, Computer Pragmatics, Computer Paradigms

Operating Systems: Ubuntu Linux, Arch Linux, Windows, MacOS

WORK EXPERIENCE

EECS DCO Computer Consultant

University of Michigan, Ann Arbor

January 2023 - Present

- Provided technical support to faculty, staff, and students on computer and networking issues
- Assisted in the setup and maintenance of computer labs and classrooms
- Managed thousands of user's accounts and permissions on Windows and Linux systems

PROJECTS

Nintendo Entertainment System Emulator - Currently developing a Nintendo Entertainment System (NES) emulator using Rust. Fully functional 6502 processor emulator with correct cycle. Will have video and audio capabilities, and ability to interface with controllers.

Technologies Used: Rust, SDL2, Memory Mapped IO

Fully Synthesizable P6 N-Way Superscalar Processor

Collaborative Project with a team of 7

- Contributed to the development of a fully synthesizable Out of Order P6 N-way superscalar RISC-V processor, emphasizing high performance and efficient execution. Developed working branch predictor, 32-bit multiplier, Reorder buffer, functional units, Common Data Bus, and Early Branch Resolution.
- Independently developed a visual debugger for the processor in C++, utilizing Ncurses to enhance debugging capabilities. Employed the use of linux pipes for inter-process communication between the SystemVerilog code and the C++ code, enabling real-time data exchange.

Technologies Used: SystemVerilog, C++, Ncurses

Network Filesystem

- Engineered a server filesystem in C++ enabling client command processing via sockets. Implemented dynamic memory allocation for upgradeable reader-writer locks, facilitating multiple readers or exclusive writer access. Dynamically allocating the locks, making use of RAII design principle and smart pointers.

Technologies Used: C++, Sockets, TCP, Smart Pointers, Threads, Boost Library

Thread Library

- Designed and implemented a functional thread library. Dynamically created contexts that could be swapped to. Included feature to acquire mutexes, and atomically put context on waiting queue and drop mutex lock by implementing a Condition Variable class. Could properly handle multiple processors attempting to access the thread library by utilizing atomic guards, and spin locks.

Technologies Used: C++, Interrupts, Library Implementation, Multiprocessor, Object Oriented Programming

Memory Page Table

- Created an OS page table that allocated space in RAM for pages. Could handle swap-backed and file-backed pages. Would evict pages from the page table in order for the computer to allocate more space in RAM. Processes that were forked would share pages, and would enact a copy-on-write policy to efficiently use the RAM space.

Technologies Used: C++, Smart Pointers, Object Oriented Programming

Instagram Clone with React and JavaScript

- Developed a client-side dynamic Instagram clone using React and JavaScript, featuring infinite scroll. Integrated a REST API and Flask for backend communication, with SQLite3 for database management.

Technologies Used: Python, JavaScript, React, SQLite3, Flask, AWS

Scheme Lexer

- Built a Lexer for a subset of R5RS Compliant Scheme. Could produce tokens that could be parsed and interpreted.

Technologies Used: Scheme, Functional Programming

Web Security

- Simulated attacking a provided insecure website using SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Learned to spot common vulnerabilities and the weaknesses of naive defenses to avoid using them in personal web applications. - Examined and exploited C code that was vulnerable to buffer overflow attacks. Utilized Ghidra to decompile executable to spot any potential vulnerabilities.

Technologies Used: Python, Javascript, Ghidra, C