I want to do research in topics like programming languages, compilers, and program verification. Through my undergraduate studies, I have gained a solid foundation across disciplines, thanks to my comprehensive curriculum in Computer Science, Electrical and Computer Engineering, and Physics. These diverse areas of study have equipped me with the skills to approach problems from different perspectives. My exploratory projects, such as designing a language framework for typesetting systems and proposing an augmented type system for JavaScript, alongside projects in operating systems and AI, like building a mini kernel and an AI-based storyteller, have deepened my interests and prepared me with the necessary skill set. With a strong interest in further advancing the field of programming languages, I am applying for graduate study to pursue my academic and research goals.

During my undergraduate years, a comprehensive curriculum provided me with both a broad academic foundation and advanced coursework in programming languages, compilers, and systems. In Programming Language Design (CS 422), I created languages in different paradigms using the K framework. In Compiler Construction (CS 426), I developed a functional KOOL compiler, and in Operating System Design (CS 423), I built several practical Linux kernel modules. These projects provided me with a strong understanding of language design, compiler construction, and operating systems. Furthermore, my coursework in Digital Systems (ECE 385) and Computational Physics (PHYS 498) strengthened my analytical and hardware-level thinking, complementing my perspective on systems and computation. Together, these courses have equipped me with the theoretical foundation and practical skills necessary to pursue innovative research in programming languages and compilers.

Building on the foundational knowledge from my coursework in programming languages and compilers, I began an exploratory project to address challenges I encountered with LaTeX during technical writing. The idea of creating a typesetting system to replace LaTeX emerged from my frustration with its poor readability and limited flexibility when writing complex ODE formulas, big-step semantics, and TikZ graphs. Lines of dense macros made source files unreadable, while simple adjustments, like spacing, often required opaque “black magic.”

To address these issues, I began designing a new language that improves LaTeX's readability and expandability. I studied The TeXbook and LuaTeX documentation. It introduces some effective typesetting concepts like TeX's Glue-Box model but relies on an outdated macro system and lacks modern Unicode support. Based on these insights, I designed a Lisp-like domain-specific language, which allows user-defined syntax and semantics to capture frequent constructs, such as those in ODE formulas or big-step semantics. This could fold cumbersome expressions into human-readable syntax, making the language significantly more readable and usable. I also prototyped an IR that generates instructions for rendering in browsers or PDFs according to OpenType font specification. While the project is still preliminary, it has strengthened my understanding of modular language design and broadened my perspective on translating high-level constructs into graphical outputs.

Beyond the typesetting project, I also explored language design through a proposal to address limitations I observed in TypeScript, a typed superset of JavaScript that I frequently use in web programming. While its subtyping system shows promise for refined typing that could better capture the behavior of JavaScript APIs, the lack of type operations often leads to convoluted "type gymnastics" or manual workarounds, undermining its expressiveness.

To address these limitations, I proposed a predicate-based type system, where compile-time predicate functions could be used to evaluate type constraints based on metadata of checked variables. Specific type operations could be implemented in predicate functions, while commonly used types are treated as syntactic sugar, enhancing the language's flexibility and usability. It also unifies imperative paradigms with type checking, where TypeScript's system is often inconsistent. The proposal deepened my understanding of predicate-based systems and its application in practical type system design.

In addition to my major projects, I have worked on several smaller projects that further developed my technical skills. I have built an AI-based interactive storyteller that generates story pieces based on user prompts. In a course project, I implemented a mini kernel for x86 architecture, which deepened my understanding of operating systems and low-level programming. Additionally, I reconstructed the Nintendo NES console on FPGA as part of a digital systems course, applying hardware design principles to a real-world system. These projects complemented my academic and research preparation by broadening my perspective in areas in hardware, systems, and AI.

My coursework and projects have deepened my interest in programming languages and compilers, leading me to apply to the University of Washington to work with faculty members who are experts in the field and to take part in its vibrant research community. I am particularly interested in developing augmented type systems for practical programming languages and using verification techniques to enhance security and maintainability in software and systems. Furthermore, I wish to research universal compiler generators and domain-specific language design to make the process of language creation easier without sacrificing performance. The strong curriculum and research environment of the University of Washington therefore make it the perfect setting for pursuing these objectives. As an undergraduate student applying to graduate school, my research about the research groups and curriculum at the University of Washington shows that it will be the best place for me to pursue my goals in programming languages and compilers.

I am especially excited by the chance to work with Professor Zachary Tatlock, whose work in the formal verification of compilers exactly aligns with my interest in correctness in high-level language tools and systems. Professor Rastislav Bodik's work on program synthesis resonates with my passion for automating and facilitating the programming process, particularly through language tools that marry human intention with machine precision. Besides, I am interested i Professor Michael Ernst's work in programmer productivity and software engineering; it aligns with my interest in type system improvement for better usability and expressiveness, while maintaining software reliability. Moreover, the research of Professor Dan Grossman on programming language theory and his contribution to type systems reveals interesting possibilities for further research on the foundations and applications of language verification. Additionally, the work of Professor René Just on program analysis and testing resonates with my interest in developing reliable and maintainable systems through practical verification techniques. The exceptional faculty and collaborative research environment at the University of Washington make it an ideal institution for me to pursue my research objectives and to contribute to the advancement of the field of programming languages and compilers.