I have built a strong foundation in the field of computer science through comprehensive coursework. My academic background encompasses programming languages, compilers, operating systems, and theoretical computer science. These courses in the programming languages designs, compiler construction, and operating systems gave me both deep insight into these subjects and practical experience, like the development of Linux kernel modules and compilers. Furthermore, the interdisciplinary studies in physics and electrical and computer engineering have enriched my technical skill set and perspectives, better positioning me to approach problems with innovation and depth.

My main academic interest is in the field of programming languages and compilers, followed by operating systems and computer architecture. The idea of a universal compiler and meta-language that can generate programming languages automatically purely based on their semantics really fascinates me. Such a vision raises a few core questions:

- How would such a framework achieve performance comparable to handcrafted compilers?
- What kind of abstraction would be flexible enough and, at the same time, precise enough to cover all thinkable languages?
- How to ensure that compilers generated this way provide all the hooks that semantics needs to interface well with underlying hardware architectures and operating systems?
- What are methods to efficiently verify the behavior of languages generated this way?

I am also interested in the design of special-purpose languages for specific needs, such as a verifiable programming language for computer architecture to formally assure chip behavior and a domain-specific language for typesetting that surpasses LaTeX syntax in ease of use and functionality for the user. In the long run, I would like to pursue an academic career, doing research in programming languages and compilers at a university. My long-term goal is to address foundational challenges in this field, developing innovative tools and frameworks that will advance software correctness, efficiency, and usability.

The Programming Systems group at UC Berkeley has an impressive history of influential projects, from UNIX and the PASCAL compiler to recent work in compiler optimization, semantics, and domain-specific languages. I am particularly interested in several faculty whose work closely relates to my research interests:

- Prof. Alvin Cheung: His research spans areas such as data management, programming languages, and building software systems. His group develops techniques to assist end-users in extracting insights from large data and techniques to handle efficient data processing pipelines. Projects include Verified Lifting for inferring program properties and development of tools like QBS and STNG.

- Professor Max Willsey: His work focuses on improving program optimization through the integration of techniques from programming languages, databases, and systems. He leads the development of the 'egg' toolkit, which uses e-graphs to construct new program optimizers.

- Prof. Sanjit A. Seshia: His interest in formal methods for trustworthy systems involves a range from mathematical specifications and software to electronic substrates. He creates fast algorithms and effective tools to solve problems such as satisfiability modulo theories (SMT) and syntax-guided synthesis. His group develops projects such as VeHICaL and LOGiCS that target verified AI and cyber-physical systems.

Working with such faculty members would provide a unique insight into the design of high-performance programming languages and compilers. I am excited about contributing to their groundbreaking research while exploring the intersection of theory and practice in trying to solve open challenges in the design of programming languages and compiler optimization.