Miguel Young de la Sota

(787) 548-0156 / mcyoung@mit.edu

I'm Miguel, an engineering leader with a decade of industry experience, focusing on compilers, optimization, and education. I believe that systems programming should be accessible, welcoming, and useful, and I bring this philosophy to every aspect of my work. This means I am passionate about mentorship, leadership, and growing the technical skills of others, and building software tools that meet users where they are. My academic background is in mathematics; I have a BSc in pure math from MIT.

Career Experience

Senior Principal Compiler Engineer, *SambaNova Systems* (2023–)

- **Graph Compiler**, *Tech Lead*. SambaNova is a silicon design company in the ML/HPC space. The graph compiler is the high level component of its optimizing ML compiler, which focuses on floating-point optimization, loop refactoring, and some instruction scheduling.

  The graph compiler team (and to a lesser extent, the compiler org) was lacking in outside experience with compiler design. I was hired to build a more sustainable software engineering culture, particularly to address velocity issues stemming from our compiler's architecture. In a sense, the mission is to win hearts and minds.

  This role requires multiplexing the often incompatible goals of product readiness (enabling a particular ML model) and building for velocity (making the compiler produce good output for *all* models). Other teams frequently call for my advice on solving compiler design problems in a way that minimizes new tech debt.

  I am also driving the development of RW, a C++-embedded language for writing linear algebra programs on our hardware. This is a new frontend for our compiler drawing inspiration from SYCL, CUDA, Fortran, and numpy. RW is a collaboration among many senior engineers: my role is to provide a coherent strategic vision for our language and toolchain.

  In addition, I have become the go-to for most compiler engineers when it comes to complex and subtle C++ or host-side performance. I have started multiple active Slack channels for discussing linting, C++, and software design, to encourage public discussion and foster collaboration and psychological safety.

  Key Accomplishments

  - Developed new design review and code review practices for our team jointly with the team's manager, focusing on enabling inter-IC collaboration.
  - Helped start a weekly MLIR reading group for engineers to learn more about the framework our compiler uses.
  - Deployed project-wide lint and formatting enforcement tools. Engineers throughout the org have contributed dozens of lint rules.
  - Lead the "SnBase" refactoring project to extract critical-but-abandoned C++ utilities into a high-quality company standard library.
  - Designed the novel programming model used by RW to express host-side and accelerator-side computation and data motion, which focuses on predictable performance.
  - Wrote almost all user documentation for RW, including slideware presented to VIP customers.
  - Mentored several engineers throughout the job ladder on technical leadership, C++ best practices, sustainable software architecture, unit testing, and compiler IR design.
  - Built positive working relationships with the managers and tech leads of the other major components of our software stack.
  - Contributed performance-critical concurrency primitives and knowledge of the Clang and GCC toolchains to help achieve customer-visible performance numbers.

Senior Software Engineer, *Google* (2018–2023)

- **Protobuf**, *Tech Lead*. Protobuf is Google's premiere evolvable schema language and data format, the foundation for RPCs and data storage at Google. Small Protobuf improvements often result in major company-wide resource savings.

  I joined the Protobuf team to lead the *Editions* project, an ambitious ecosystems-wide effort to bring incremental language evolution to Protobuf. I was responsible for the design and execution strategy, as well as the org-level OKRs.

  I was simultaneously the lead for *Protobuf Rust*, a first-party implementation of Protobuf for emerging Rust users at Google. I was responsible for the delivery of an implementation that met our evolvability and performance-critical needs, as well as the technical growth of senior- and staff-level contributors.

  I also contributed optimizations and code health cleanups to our open-source C++ project as spare time allowed.

  Key Accomplishments

  - Led design of Protobuf Editions, which was reviewed and approved by all ecosystem stakeholders, including Sanjay Ghemawat.
  - Ramped up a staff engineer to handle tactical execution of the Editions roadmap.
  - Defined norms and expectations for design contributions on Rust Protobuf, with a focus on psychological safety.
  - Held daily Rust seminars for new senior contributors to learn advanced Rust concepts and techniques.
  - Implemented and deployed a new JSON codec[1] for Protobuf C++, including extensive tests and bug-compatibility. This codec is used by all Google production services that process JSON input in C++.
  - Refactored the core codegen utilities of `protoc`, improving compiler developers' velocity.
  - Designed and implemented Protoscope[2] as a 20% time project. It is a simple, human-editable format for inspecting and modifying potentially invalid Protobuf wire format blobs. This tool is used extensively by optimization engineers to debug wire format codecs.
  - Participated in the on-duty rotation, keeping CI green and answering user questions.

- **OpenTitan**, *Software Engineer*. The OpenTitan project is an open-source root of trust SoC, consisting of both silicon designs and firmware to run on the device. I responsible for our low-level support libraries and contributed significantly to our build system and cryptography libraries.

  I was also responsible for the *Manticore* project, an implementation of Microsoft's Cerberus attestation protocol in Rust.

  Key Accomplishments

  - Migrated the entire project from ad-hoc Make scripts to Meson; two years later, contributed a significant portion of the migration from Meson to Bazel.
  - Established a long-term relationship with Microsoft senior staff engineers working on Cerberus, including a mutually agreed-upon RFC process for Cerberus.
  - Unified our linker scripts and assembly files under one project style.
  - Implemented and maintained core libraries, such as C runtime support and optimized math utilities.
  - Developed and implemented methodology for unit-testing driver code off-device.
  - Designed fault-injection and power analysis attack mitigation strategies for use in high-assurance privileged code; multiple patents were filed based on this work.
  - Engaged with customer teams on cryptographic primitive requirements, balancing legacy use-cases with modern security practice.
  - Provided technical mentorship for multiple new hires, including our cryptographer.

[1] `https://github.com/protocolbuffers/protobuf/tree/main/src/google/protobuf/json`
[2] `https://github.com/protocolbuffers/protoscope`

- **20% Work**. In addition to my assigned work, I dedicated significant time to community efforts, primarily focused around the theme of education.

  Key Accomplishments

  - Joined the admin group of *C++ Readability*, a mentorship program that teaches C++ best practices to engineers via randomized code review. I also performed approximately 500 such mentorship reviews.
  - Taught multiple sessions of *Comprehensive Rust*[3], a multi-day Rust 101 course; student feedback was overwhelmingly positive. I also organized initial development of a revised edition of the course materials.
  - Wrote entries for the *C++ Tip of the Week* best practices publication.
  - Kicked off the Rust style guide working group, contributing strategic vision for Rust best pratices at Google.
  - Designed *moveit*[4] a novel move semantics library for Rust, bringing custom move operations to the language. This design was adopted by Google's Rust/C++ interop tooling for bridging C++ move constructors to Rust.

Software Team Member, *Harvard–MIT Mathematics Tournament* (2016–2017)

HMMT is a mathematics olympiad for attended by hundreds of high-school students from around the world. HMMT uses custom scheduling and scoring software that is critical to a smooth show.

I developed an Android app in Kotlin that provided teams and coaches with personalized schedules, directions, and notifications on tournament day. I also designed the REST API used by the Android and iOS apps to communicate with HMMT's services.

Lead Developer, *Octagami's Omniverse* (2013–2016)

Octagami's Omniverse was a small MMORPG with sandbox game aspects. We served over 500 simultaneous players and surpassing 250K unique users. Omniverse's unified game world used a distributed machend to overcome performance bottlenecks.

As the lead for server software development, I was responsible for the backend game server architecture, which leveraged existing open-source proxy technology. I also helped develop new gameplay features, ensuring that they could scale to our player count while leveraging our architecture. I was also responsible for JVM performance tuning of game server nodes.

Personal Projects

I maintain a personal blog at *mcyoung.xyz*, where I post long-form explainers on advanced systems-programming topics, written for an intermediate-level audience, with a focus on making the material accessible.

Most of my larger presonal projects are some kind of toolchain or programming language. *Alkyne*[5] is a pure scripting language inspires by Starlark and Jsonnet, intended as a Starlark alternative; *snasm*[6] is a full Super Nintendo (MOS 65816) toolchain, including an assembler, a disassembler, and a linker; *jas*[7] is a JVM bytecode assembler.

Other smaller projects include *0x*[8], a binary dumper with colorization capabilities and an RPN calculator, and *voltorb*[9], a Picross-like game with graphics and animations that runs inside of a terminal.

[3] https://github.com/google/comprehensive-rust
[4] https://github.com/mcy/moveit
[5] https://github.com/mcy/alkyne
[6] https://github.com/mcy/snasm
[7] https://github.com/mcy/jas
[8] https://github.com/mcy/0x
[9] https://github.com/mcy/voltorb