Meridian Research

Executive Summary

Why I'm Building Meridian Research

The Idea That Started Everything

I'm 18, and I've been coding since I was 15. But here's what drives me crazy: every time I want to build something, I must translate my perfectly clear idea into some programming language that makes it unnecessarily complicated.

I know exactly what I want the computer to do. The computer is smart enough to understand natural language better than most humans understand code. So why am I still writing functions and classes and debugging syntax errors?

That's the moment Meridian Research was born. I realized programming languages aren't a necessary part of computing. They're just historical artifacts from when computers were too dumb to understand human language.

Well, they're not too dumb anymore.

The Real Problem Everyone's Ignoring

Every programming tutorial starts the same way: "First, you need to learn the syntax." But syntax has nothing to do with the problem you're trying to solve. It's just arbitrary rules someone invented decades ago.

Meanwhile, I can describe a complex algorithm to ChatGPT in plain English, and it understands perfectly. The technology already exists for computers to understand human language better than most programmers understand assembly code.

The problem isn't that we need better programming languages. The problem is that we need programming languages to disappear entirely.

Think about it: you have an idea, you translate it into code, the compiler translates it into machine language. Why not skip the middle step? Why not just tell the computer what you want in the language you already speak?

What I'm Actually Building

Most people think this is about making better chatbots. It's not.

The breakthrough isn't human-computer interfaces. It's machine-to-machine communication through natural language.

Right now, when your phone talks to a server, they exchange JSON or XML. When AI systems coordinate, they pass around vectors and matrices. When microservices work together, they use rigid APIs with strict schemas.

But what if they just talked to each other like humans do?

What if your banking app could say "John wants to transfer $500, but he's cutting it close on his checking balance - should we suggest he wait for his paycheck to clear first?" And the server could respond "His paycheck usually hits on Fridays, but it's only Tuesday. Let's warn him and ask if he wants to proceed anyway."

This isn't science fiction. The language models exist. What's missing is someone crazy enough to rebuild computing around conversation instead of code.

Why Everyone Says It Won't Work (And Why They're Wrong)

"Natural language is too ambiguous." Humans coordinate incredibly complex tasks through natural language every day. The Manhattan Project. The Apollo Program. Building skyscrapers. If language was too ambiguous for complex coordination, civilization wouldn't exist.

"It'll be too slow." Only if you think this means running everything through GPT-4. We're talking about purpose-built hardware that processes language as efficiently as current chips process math.

"The industry will never adopt it." The industry said the same thing about personal computers, the internet, and smartphones. Every major shift looks impossible until someone proves it works.

Industry Experts who validate my approach:

Jensen Huang, CEO of NVIDIA, said "English will become the universal programming language." He's right, but he's thinking too small. English won't just be how humans program computers. It'll be how computers program each other.

Dario Amodei talks about single researchers creating billion-dollar companies with AI tools. That's exactly what's happening here. The tools now exist for small teams to tackle problems that would have required entire industries.

The timing is perfect. AI capabilities make natural language computing possible. Developer frustration and accessibility demands make it necessary.

How I'm Going to Beat Everyone Else

I'm 18. That means I'm old enough to understand the technical challenges but young enough not to care that everyone thinks it's impossible. I don't have 20 years invested in the current system. I don't need to protect existing businesses built on programming languages.

I built a research system that works 24/7. While competitors are stuck in meetings and managing teams, my AI agents handle everything except the breakthrough thinking. I spend 99% of my time on actual research.

I'm targeting the foundation, not the surface. Everyone else is building better development tools. I'm eliminating the need for development tools entirely.

I understand both hardware and software. This isn't just a software problem. You need processors optimized for semantic operations, not arithmetic ones. I'm building the full stack.

The Business Reality

I'm not building this as an academic exercise. This needs to make money and change the world.

The commercial opportunities are massive:

Patents that matter: Own the fundamental IP in natural language computing protocols. Every company that wants to build post-programming systems will need licenses.

The platform everyone uses: Build the development environment for natural language computing. Capture the market that's currently split between dozens of programming languages and IDEs.

Hardware that makes it fast: Partner with chip companies to build processors optimized for language instead of math. Enable the performance that makes adoption inevitable.

Tools for the transition: Build the migration systems that help the industry move from code to natural language. Make the transition painless instead of painful.

The 4-Phase Plan

Phase-1: Prove it works Build prototypes that beat traditional programming for specific tasks. Not just match performance, exceed it. Get developers to choose natural language because it's better, not because it's trendy.

Phase-2: Make machines talk Design the protocols for machine-to-machine natural language communication. Show AI systems coordinating complex tasks through conversation instead of APIs.

Phase-3: Build the hardware Integrate purpose-built processors with our software stack. Prove that natural language computing can be fast, reliable, and efficient enough for production systems.

Phase-4: Start the transition Build the tools that help the industry migrate from programming languages to natural language. Train developers. Support early adopters. Make it obvious that this is the future.

What Success Looks Like

Short term: Developers choosing natural language over code for new projects because it's faster and less error prone.

Medium term: Computer science programs teaching natural language computing instead of programming languages.

Long term: A world where anyone can build software by describing what they want, regardless of technical background.

This isn't just about making programming easier. It's about making programming unnecessary.

The Funding Opportunity

I need resources to build this full-time without distractions. Dedicated computing infrastructure for my research agents. Hardware prototyping partnerships. Ability to hire specialists when I need specific expertise.

This is a chance to fund research that could eliminate programming languages entirely. The early investors in personal computers, the internet, and smartphones saw similar opportunities. This is bigger.

Why This Will Actually Happen

Programming languages exist because computers used to be too stupid to understand human language. That's not true anymore.

The only question is whether the transition happens gradually over 20 years, or rapidly over 5 years with the right breakthrough.

I'm betting on rapid. I'm building the breakthrough that makes it happen.

The future doesn't need programming languages. It just needs someone willing to kill them.

That someone is me.

Meridian Research: Where code goes to die and ideas come to life.