Introduction

This is the story of what I observed, participated in, learned, and did across the 60 years I worked in the computer industry. Across that period, I worked on pretty much every component of a computer system from the application code and the operating system to the hardware itself including processor architecture, RTL, and processor microcode. I worked at IBM in its heyday, Dell in its early days, a startup (Centaur Technology) that designed Intel-compatible processors, and finally, Intel itself. I am an inventor on more than 300 U.S patents, have been an individual contributor, and have also managed teams ranging in size from a few people to more than a thousand. But all this happened after I was a failure at age 21.

The story of this early time is this: I was a star student in high school—I skipped a grade, took college math classes at UC Berkeley in my senior year, and graduated HS at age 16, and then enrolled at UC Berkeley as a math major. However, the environment at Cal combined with my immaturity soon resulted in my not paying attention to classes and I flunked out of Cal after three years. But that was ok with me because I had a new career planned; I soon started work as a commercial helicopter pilot and instructor, but that career ended after only one year due to my poor eyesight. So, in the summer of 1963 I was 21 with no college degree and no obvious useable skills, and no guiding interest or ideas about a career.

But I quickly got a job as a lab technician at Shell Development which turned into an assignment programing a newly arrived IBM 7044 mainframe, and thus I discovered my true love: computers. After almost two years primarily programming scientific applications in both Fortran and assembler, I returned to school (CSUEB) and in two years received my BS and MS in Mathematics. Even though my degrees are in real math (number theory, set theory, foundations), most of my time in the second year was spent programming the school's IBM 1620 computer. As part of that computer work I received a graduate research grant from UCLA and I also worked part-time as a computer consultant to a paper manufacturer. It was this consultancy job that introduced me to IBM's real-time process-control computers, which I loved, and conveniently were developed in nearby San Jose. Thus, upon graduation in 1967 I joined IBM as an Associate Programmer. And, the rest is the interesting history.

At IBM I led the design and managed the development of several innovative and important IBM products. I received several major corporate IBM awards and was appointed an IBM Fellow. In the 1980's I was a close witness to some of the events that helped lead to the downfall of IBM's dominance. The last straw for me was a idea I had as an IBM Fellow that would have potentially changed some subsequent IBM failures. After a year of fighting IBM executives who were stuck in the mainframe mindset, I quit IBM in disgust in mid-1988; one of the few IBM Fellows to ever quit such a great job.

After leaving IBM, I next worked directly for Michael Dell for over five plus years as Senior Vice President of the Product Group and CTO of Dell Inc. where I managed product development and, at various times, world-wide manufacturing,

procurement, IS, technical support, and facilities. Since I was in charge of products, I spent a lot of time studying Intel processors and concluded that there was an opportunity for a new competitor. I left Dell in early 1994 with the dream of starting a company to develop a “better” Intel-compatible microprocessor. Three others from Dell joined me and we worked as contractors to MIPS while developing our ideas and looking for funding.

After a year, we obtained funding and founded a microprocessor design company (Centaur Technology Inc.) as a subsidiary company of IDT, and later as a subsidiary of VIA Technologies. I was the President for 24 years and along the way, I was the inventor on over 300 US patents and was closely involved in two important and successful patent suits (with Intel and Apple). After retiring in 2019 as President and moving to California for family reasons, I still worked remotely for Centaur doing hardware design and was also a Professor-of-the-Practice at the Naval Postgraduate School.

At the end of 2021, Intel effectively bought Centaur Technology and I was hired into the Labs division as a Senior Fellow. There I developed a new highly secure RISC-V processor that made it into silicon. But the new Intel management closed the Labs and killed the project and I retired at the end of 2025.

So, why am I writing this history? First, the real “secret” to my successes were "right time, right place, and—especially—right people"; I want to recognize these people. Second, my personal philosophy over this period was “never compromise technical excellence.” This has worked well (for example, at IBM) but also not as welI at other places. I hope that there are lessons or motivations for others in this story.

Another topic is what I have learned about managing technical groups—good and bad—culminating in the unique and successful approach that I used at Centaur. My extensive technical management experience has given me very definite ideas about how technical groups should be—and should not be—managed.

Finally, I want to highlight some of the early accomplishments in computer design—some that I observed, and some I participated in. Today’s processor chips are very complex and contain many billions of transistors. However, in spite of this complexity and size, the design task of a processor is easier than computer design in the early years. The early designers had to deal with severe physical technology limitations: tubes, discrete transistors and components, core memories, massive power and cooling requirements, and so forth. In addition, tools such as compilers, CAD design tools, simulators, emulators, and verification tools were primitive or non-existent. And there was little or no base of computer design or computer science theory; almost everything had to be invented from scratch. The computer technology and tools that students today take for granted required many innovations and great engineering work. This early work should be remembered.

In particular, I will summarize the instruction sets of the various processors associated with my career. Today, the hardware instruction set of a computer processor (the "assembly language") is not a critical factor since most software is written in high-level languages and compiled or interpreted into the low-level

hardware instructions. Furthermore, there are now only a few (two or three depending on your bias) modern instruction sets that dominate the market and the performance difference between them (based on the instruction-set type) is not significant.

However, for at least half of my career, the hardware instruction set choice was the *primary* factor determining the function, performance, and cost of a computer. The hardware was very expensive, and each system had to be tightly tuned for a particular application type and price target. For example, when I started programming on an IBM 7044 mainframe in 1963, IBM was shipping many different processor architectures, each having a different instruction set: the IBM 704x, 707x, 7030, 7080, 709x, 7740, 7750, 1448, 1401, 1410, 1440, 1620, 1130, and 1710 systems. Almost all of the operating system code, and many of the applications for these machines was written in assembly language and was thus machine dependent.

Furthermore, the instructions sets were designed by engineers who, at that time, had little understanding or concern for software but were trying to minimize hardware costs. Trying to change this hardware-centric approach was a major focus of my IBM life and led to my biggest accomplishments at IBM.

Because of the importance of processor architecture, and because I was especially involved with it throughout almost my entire career, and because I really love it, I will provide brief summaries of the various architectures that my career intersected.