

Mainframe Hall of Fame

An Interview with John Ehrman of IBM

By Robert W. Shimizu



I'VE KNOWN JOHN EHRMAN AS "MR. ASSEMBLER" FOR YEARS AT IBM. WE see each other at SHARE, where he confers with Dave Cole, my boss. John also supports the "Assembler Boot Camp" at SHARE, which seeks to expose new people to Assembler language. Given this high level of visibility, it seemed natural to approach John for this series.

Q: John, when did you get into computers and how?

John: I'm showing my age, but in 1958, I was a graduate student in Physics at the University of Illinois. I took a programming class on the original ILLIAC I. I was HOOKED! I like to say that I caught the "original computer virus!"

Had the university offered it, I would have changed my major to computational physics, but it was much too early for that, so I finished my degree in physics. I've stuck with computers more or less entirely since then.

Q: What do you consider your outstanding achievement?

John: There's nothing that I'd call "outstanding." After I left the University of Illinois, I worked for 17 years at the Stanford Linear Accelerator Center, known as "SLAC"—a high-energy physics lab. Having done a bit of teaching at University of Illinois, I also did some occasional teaching of S/360 Assembler language on the Stanford U campus.

At the time the tools for teaching Assembler really didn't exist. I and a group of students at Stanford wrote a student Assembler/Interpreter. It was called a "Single Pass Assembler or "SPASM." It got pretty wide distribution through the SHARE program library, and it helped beginners

to learn Assembler without doing damage to the operating system, which was quite easy to do if they ran on the bare system. You see, the operating system at that time didn't have very strong "defenses."

I also wrote a multiple precision floating point package, which was actually used by IBM in developing their original support for extended precision hexadecimal floating point in the early 1970s.

At SLAC, I wrote mostly in FORTRAN. When I joined IBM, I worked in the FORTRAN group. At the time they were working on the vectorizing FORTRAN compiler.

Around 1990, the planner for the Assembler retired, and having taught Assembler, I was interested in learning what could be done to improve the Assembler and its language. So I began to dig around in internal databases of customer requirements and requests, and found that there was a huge backlog of them that management wasn't aware of. This was probably because through much of the '70s and '80s people thought that Assembler was long-gone—nobody did that any more. They still think that way sometimes!

So, I put together a document that tried to collect and collate these requirements and it grew to 100 pages. I was lucky to have the help of a lot of people at IBM—support that led to the approval to build the High Level Assembler, which came out in 1992.

Q: So, you'd say your crowning achievement was the High Level Assembler?

John: I'd say the most useful achievement. I've always been interested in tools that leverage the mental effort that programmers have to put into writing programs. Improving any of those tools produces better results with less effort.

Q: So, today you are pretty much responsible for HLASM? You're the Team Leader?

John: It's hard to describe. I think that within IBM I would be called a "Product Planner." It's like being the Chief Cook and Bottle Washer. Correction: I'm not the cook! There's a highly skilled team in Perth, Australia who do the actual coding and service work, the documentation and so forth. But I have my fingers in all sorts of little areas, and handle a lot of the bureaucratic responsibilities.

Q: Do you have anyone you would consider your "mentor?"

John: Yes, my primary mentor was the guy who taught my very first programming class, a guy named Lloyd Fosdick. After he left the

University of Illinois, he went to the University of Colorado at Boulder, and is retired now.

Then, when I was at SLAC I worked under William Miller, who went on to become the Provost of the university (Stanford) and a fellow named Charles Dickens. Both Miller and Dickens were very encouraging and allowed a fair amount of free experimentation, which I guess was consistent with the lab's mission. It was a very energizing work environment.

Q: What's your normal work-week?

John: I really don't have a "normal" work-week. It varies between 30 and 80 hours a week depending on what needs doing. I "do" planning, handling customer requests and requirements, answering questions, a little bit of coding or prototyping and giving SHARE presentations—which is how we met.

One of the really interesting aspects of the job is that I get to work with groups in all other areas of the company like hardware architecture, operating systems, subsystems, programming languages and so on. It's the variety of these different activities that makes my job both fun and satisfying.

Q: Where do you see the industry going?

John: This is a real "fuzzy" crystal ball, but I do see increasing and pervasive reliance on computing technology and increasing concerns with information security and privacy. I don't think those are terribly novel ideas.

Q: Do you think we'll actually achieve cross-platform communications, like a Sun talking to a Linux talking to a VM talking to a z/OS talking to a Wintel box?

John: Yes, all of the above. The industry is still struggling with the question of standards, so I think as time goes forward, there will be more customer pressure for standards that allow better inter-connections and inter-communication. I think it's a necessity. For example, you can't sell vacuum cleaners that run on 330 volts in this country.

Q: What would you do differently if you had to start over?

John: I consider myself very lucky. I sort of fell into this and I'm really very happy with where I wound up. It was not a conscious choice, so when you ask, "What would I have done differently?" that might imply that I had to think about it. And I didn't; it just happened that way! I'm grateful.

Q: What advice would you give to people who are just getting into mainframes?

John: Just getting into mainframes? It's the same advice I'd give to any beginning programmer, and that is start with a machine or an Assembler language. That's really the key to understanding what's happening under the covers of any programming language. Some of the more modern languages are certainly more powerful, but they also insulate the programmer from understanding the impact their code might be having on system resources, since there is such a large distance between the code and the actual execution environment.

It often comes as a surprise to university students to discover that computer resources aren't "cheap"; they aren't free. And that some commercial applications have to handle hundreds of millions of transactions

per day; hundreds of them per second. They have to be able to recover from data corruption and from system crashes, in very short times and with very high reliability.

I don't think that's the kind of thing that is very high on the educational priorities in the student's environment, where they usually work on either their own computers or some kind of workstation where if something crashes, that's just a "bug" in their homework.

Q: When you say they should "Learn the Assembler," that's a rather cunning trick, because if they're going to learn the Assembler, they're going to HAVE to read the "POO," won't they?

John: I've seen teaching methods where students stick with high-level languages and they never really have to understand the impact of doing things one way or another. If things run slow, then that's because the machine is slow, and they never realize that they've made an unfortunate choice of algorithms or languages or file organization and so on.

Something that exposes them to the architecture of REAL machines makes them see how the "motor" works, if you like.

Q: So your overweening comment is "Learn the architecture?"

John: Learn AN architecture and how to program at that level.

Q: Is there anything we can do to stimulate people coming into the mainframe industry?

John: That's a very broad question. One of the best stimulants is the availability of jobs. There was plenty of evidence of that in the dot-com boom of the '90s where computer science departments at colleges were being overwhelmed with applicants. It's off now that the boom has deflated somewhat.

To go back to your earlier question about where the industry is going, the fact that computing in its many shapes and forms is with us and *will continue to be in ways we can't anticipate* means that there WILL be a need for people who understand the technology and can manage it.

If you're talking specifically to high-end environments, like the mainframe, I think that there certainly needs to be an appreciation at the educational institutions of the enormous complexity of these environments.

Q: What else would you like to expand upon? What else would you like to tell our readership?

John: I think that the computing world offers a lot more opportunity for creativity—for injecting some sense of your own personality into what you do—than many other kinds of jobs and work environments might. The fact that it varies so greatly from year to year means that it's never dull. It may be taxing and demanding, but that's not a bad thing. 🎧



NaSPA member Robert Shimizu provides first-level technical support for Cole Software.