



Migrating to z/OS and 64-bit Architecture—A Conversation with Dave Cole of Cole Software LLC

By Bob Shimizu

COLE Software produces the Assembler language debugger, z/XDC™. This tool helps programmers work on simple or complex applications in a wide array of environments. z/XDC works on TSO commands, batch programs, multitasking programs, APF-authorized programs, SVC routines, PC routines, system modifications, and exits into the operating system itself.

Many of the world's mainframe software developers use z/XDC. So, when IBM announced z/OS, Cole Software had to respond.

Dave Cole has been a software developer and systems programmer for nearly forty years. Dave has been the architect of the XDC product line throughout its service life. This interview reveals the challenges Dave faced in migrating an extremely complex 31-bit application into the 64-bit world. If your team plans to exploit 64-bit architecture, then Dave's work will help you.

Q: When did you first begin to work on z/XDC?

A: I first became aware of 64-bit architecture in February of 2001, when IBM blind-sided me with the news. The G.A. date was only a month or so later! Our product depends on structures in the OS that historically have remained stable over decades. In z/OS, however, these structures changed. I had no idea that the effort of building a compatible version of the product would take two and a half years. Nor did I expect to add 110,000 lines of code to the product! z/XDC has 160 modules and now represents over a half-million lines of code. It may be the largest ESTAI or ESTAE ever built.

Q: What were the initial challenges you faced in building z/XDC?

A: First, I had to support about 160 new machine instructions, which it turned out, involved a bit more than just updating some tables. It took from February to July of 2001 to support the new instructions, primarily because of split op-codes. Years earlier, IBM introduced new instructions for the vector processor that put

an op-code's first byte at the beginning of the instruction and the second byte at the end. IBM had a good reason: Splitting the op-code gets the second byte out of the way of the EX instruction. Trust me, that's a good thing!

When I first saw the split op-code instructions, I realized, "One day I'll have to deal with this, but not today." In a sense, I was right: the vector processor and all of its nasty machine instructions did go away. However, the concept of split op-codes didn't. With z/Architecture, they came back with a vengeance, and there was no getting around them. I had to support them.

Anyway, it wasn't until July 2001, before I was able to get the product to assemble and run again. That was the easy part. The more difficult part was writing the actual 64-bit addressing support. This effort took the remaining two-plus years of development.

64-bit addressing touches z/XDC in *thousands of places*. Once I started the second phase, it took **fifteen months** before the product would even assemble again. Yes, that's fifteen months before it would get so far as to produce its initial display!

Q: What major enhancement did you put into z/XDC?

A: You mean in addition to 64-bit addressing support. Well for one thing, I totally re-engineered the FIND command. When a user searches virtual storage in the 64-bit environment, there are huge "holes" where nothing resides. A linear search through "infinite" storage takes an "infinite" amount of time. So, I added support that allows the user to control the range of storage to search. I also enhanced support for skipping empty chunks of storage, and I implemented the ability to search by intervals. z/XDC can now do skip searches through storage, looking for matches that are either half-word aligned or page aligned or **whatever** aligned. Now a user can look for particular bit patterns, not just rigid byte strings. Users can stop the search at the n^{th} occurrence of a match (not just the first) and generate equates that label each match.

I also added support for “wide” displays in z/XDC so that registers, storage and source code can take advantage of the larger screen geometries supported by today’s terminal emulation software. I run my personal copy of z/XDC on a 62 x 130 display, so I view 32 bytes of storage per row instead of the 16 permitted by the old displays. By the way, I ought to thank Ed Jaffe (of Phoenix Software), who pushed for that idea.

I added support for the z990’s Long Displacement Facility, which utilizes 20-bit wide displacements. And I closed a “hole” in the earlier XDC versions wherein deferred breakpoints could not be set in a PDSE resident module. Also, I made z/XDC downward compatible. z/XDC runs on OS/390 systems, so users get the benefit of many of the new features in older releases of the OS.

Q: What technical advice would you give to those who are considering exploiting 64-bit architecture?

A: There are a couple of things: First, you can use “above the bar” storage only for data. z/OS doesn’t yet support the execution of programs in high storage. Second, be careful how you use high storage. It’s one thing to think that you have trillions of bytes, but it’s quite another to actually use that much. Storage doesn’t actually exist until you reference it. It isn’t until then that z/OS actually assigns a page of real storage (and eventually a slot on a real page dataset). If you have some sort of scan routine that touches an obscene number of pages, you will bring the system very rapidly to a grinding halt!

Q: Do you see a great movement to 64-bit architectures in the industry?

A: I don’t see a great deal of companies moving in that direction yet. For instance, most Fortune 500 companies are going to z/OS to fulfill throughput requirements, not programming necessities. But there are several reasons why IBM came out with 64-bit architecture:

1. Marketing: PC chips already supported 64-bit addressing. In the numbers game, “64” sounds better than “31.” One “IBMer” told me that the hardware guys wanted 63-bit addressing, but the marketers cringed at having to explain, “half as much storage [eight quintillion bytes] was just as good.” (They’re right! But that’s hard to explain.)
2. Simplification: Previously, IBM went “horizontal” with memory by enabling methods for accessing more and more data spaces. Now they’re also going “vertical” with fewer but much larger spaces. It’s easier to understand, and it’s easier to do.
3. Compilers will probably make use of high storage, possibly at compile-time and probably at run-time as well. Certainly, there should never again be a need to have those cumbersome //SYSUTn work area spill files.
4. Database systems and transaction processors that need to support an unlimited number of people, processes, and accesses coming off the Internet will probably stuff a lot of their work areas and data above the bar.

The mainframe has become a small cog in the huge machine that is the Internet. In order for the mainframe to support many millions of transactions per second, it needs vast resources, and the availability of more fast access memory is becoming increasingly crucial.

Finally, IBM already is exploiting its 64-bit architecture and dragging software vendors along with it. Since those software vendors are our primary customers, they’re dragging us along as well.

Q: Who helped you during your work on z/XDC?

A: Well, Bob Rogers of IBM offered insights from time to time. Peter Harris of Software AG offered me a testing site, but it was two years before I was ready to use it. Still, I appreciated the offer.

I got great support from Chris Blaicher of BMC, Ed Jaffe of Phoenix Software and Sam Knutson of Allen Systems Group. Chris had an immediate need for 64-bit support and put it right to work. Both Ed and Sam took my fixes, installed them, and gave me quick feedback. Ed, in particular, made a number of suggestions that I implemented. I also had help from the folks at Candle Corporation, Systemware, and the rest of our beta customers.

I also got support from the small number of extremely sophisticated programmers—those thousand or so people who support most of the world’s mainframe development—who use z/XDC and the products before it. My interaction with them drives the evolution of z/XDC. It’s a small but very enthusiastic user community.

Q: Can you recall any special “AHA!” moments during the project?

A: No, not really. I get a great deal of satisfaction out of the initial design. I like an initial design that achieves a result in a simple, direct way—an elegant solution. Unfortunately, no one ever sees that elegance. It’s often hidden too deeply in the code.

But once I’ve built the framework, it’s just a matter of persistence to implement it. That can be a long process. You start at one end and work to the other. With 500,000 lines of code to deal with, that distance is very long indeed. To be a good programmer, you have to cope with a LOT of tedium, sometimes for months or years at a time. You just have to believe in the result and keep the goal in mind.

Q: What did you learn about yourself in this new release effort?

A: (Laughs) Well, I learned that it’s possible for me to become bored with coding! I learned that in the future I’m going to have to focus on training successors, people who will carry on the work while I fill an advisory role. Towards that end, I’ve hired two younger programmers who are learning the product and the mainframe world. That was a difficult decision to reach, but I want z/XDC to keep evolving. It’s time to bring more people onto the team.

Q: Where do you see z/XDC and Cole Software heading from here?

A: We’d like to get z/XDC involved in reading system dumps. Also, I expect to support other high-level languages and to run under other operating systems. And I would like to move into the GUI world.

We can consider new directions now that z/XDC is out the door. We’ll continue listening to our customer base as we move forward.

Cole Software’s Web site URL is www.colesoft.com.

For more information, please contact Bob Shimizu at (800) XDC-5150 or send e-mail to bshimizu@colesoft.com.