Interview of Richard (Richie) Lary, part 1

Interviewed by:
Tom Burniece

Recorded: August 14, 2015
Mountain View, California

CHM Reference number: X7581.2016

Tom Burniece: Hello. I'm Tom Burniece, a volunteer at the Computer History Museum, and we're here today with Richie Lary, who was a long time DEC (Digital Equipment Corporation) …

Richard Lary: employee.

Burniece: … engineer, programmer and architect, who has some really interesting stories to tell. For full disclosure, I want you let you know that I was his boss at DEC for a while. So I know Richie pretty well from the '80s, when he was working for me. But we're going to focus on the story from the beginning plus a ways past his time at DEC. So I'm going to start with Richie. Why don't you introduce yourself a little bit and talk about where you grew up and …

Lary: OK.

Burniece: … what you thought about early on in life?

Lary: Sure. My name's Richard Lary. I was born in Brooklyn, New York in 1948. I grew up in a very classic family. My mom and dad were married for their entire lives. I had two younger sisters and went to the New York public school system.

One of the good aspects of the New York Public school system was that they had a series of special science and math high schools. They also had art high schools and music high schools.

But they had some science and math high schools and I was mathematically inclined and mathematically talented from a young age. My parents were not particularly technical, but they were bright -- my mom had a college education. My father had to drop out of college because of the depression but was also very bright. [They] encouraged, more than encouraged - required my education and pursuing my educational dreams, which was to go into mathematics.

So I wound up at Stuyvesant High School in Manhattan, which was one of the really good science and math high schools in the country. Luckily, I was in the right place at the right time for the first time in my life, first of many. And the second time I was in the right place at the right time was when I was a senior in Stuyvesant and the NYU Courant Institute, which was less than a mile from Stuyvesant, as the crow flies, got themselves a supercomputer.

It was a CDC 6600. I'm sure the museum has documents on it if not a partial copy. It was Seymour Cray's first really big computer.

The Atomic Energy Commission (AEC) bought it on a grant for NYU Courant, which was an Institute of Mathematical Physics, probably to do research on atoms, probably weapons related. And the NYU Courant Institute discovered that they could only use a fraction of that machine. So a guy named Henry Mullish, who was at Courant Institute at the time, and affiliated with the AEC - rumors were he was [also] affiliated with the CIA - convinced them to open up that supercomputer to high school students.

Of course, Stuyvesant being right next door, so to speak, we heard about it and a bunch of us started trooping down after school to NYU to use what was at the time the fastest computer in the world. So talk about being spoiled and talk about being in the right place at the right time.

So we learned Fortran but never really went beyond Fortran, because we were all kind of math geeks. We'd write mathematical programs in Fortran to figure out prime numbers and do other puzzles and things like that.

But it was a wonderful introduction to computing and it was completely unstructured. They provided a big bullpen for us and we would all kind of exchange ideas and do a lot of one-upmanship, who could do this better, who could do this faster …

Interestingly, a number of us went on into the computer industry, surprise, surprise. So I was there and a guy named Steve Rothman was there, who later wound up working on the VAX architecture with me. A guy named Bob Frankston was also there, who later became the co-inventor of VisiCalc, the first spreadsheet.

I don't remember a lot of the other names who went there but I'm sure they all went on into the computer industry. So that's how I spent my senior [year] in high school. By the time I graduated high school, I was already an accomplished Fortran programmer, right, and knew a fair amount of math.

A friend of one of the people at Courant had a little company that did statistical analysis. It was called General Analytics. His name, I'm pretty sure, was John Robbins and he used to get a lot of government contracts to analyze health data for the Department of Health and educational data for what was then the equivalent of the Department of Education (I don't think the Department of Education existed then). He also got demographic data for the Office of Economic Opportunity. So he hired the lot of us in the summer of 1965, because he could get talented programmers for $2 an hour.

Working at General Analytics was just a continuation of fooling around at Courant Institute, except that this time we had some goals set for us. So the first thing that we did was to write a statistical package that they would use to crunch data for all these government organizations. I got to write something called oblique rotations, which is an arcane statistical technique, involving a lot of matrix inversions and things like that.

Other people wrote analyses of variance, regression and things like that. We either would submit our stuff through a remote batch terminal, which was the rage then in the mid '60s; it was basically a card reader and a printer, with some kind of hard network connection; or, we would use an IBM 1130 computer, which was IBM's equivalent of a mini-computer at that time. It was a small scientific computer that still ran off of punch cards, the standard IBM batch philosophy.

I went to college at Brooklyn Poly, (Polytechnic Institute of Brooklyn), which is now NYU's School of Engineering, in Brooklyn, New York, but I had actually qualified to get into MIT. I was accepted by MIT and had a Merit Scholarship but money was still tight, so we couldn't really swing MIT financially. Meanwhile, New York State offered some scholarships, where I could live at home and go to Poly, so it was a lot easier to just go to Poly.

I went in there and they had a computer center with a machine, that was a relative of the one that we were submitting our remote batch jobs to, which was an IBM 7040 (big 36-bit IBM iron, scientifically oriented). So I started hanging out at the computer center and then I ran into a bunch of people at Poly.

A lot of them later moved on to Digital (DEC) or other places in the computer [industry]. Some of them were Jack Burness, who later became a computer graphics guy at DEC, and Lenny Elekman, and Len Shustek, now Chairman of the board of the Computer History Museum and one of the people responsible for moving it to California. [Note: Others included Joe Fischetti and Hank Maurer, plus a bunch of other people].

There were multiple kind of circles at Poly. The original one started out at the computer center, where there was that 7040. We also had a line[several, actually], I'm not sure if it was initially or later on, to a time-sharing system called GE Time Sharing, which used a variant of Dartmouth BASIC and allowed you to access their GE-645 mainframe. And at some point, we also had a time-share connection to a system that ran APL, which was Ken Iverson's very radical computer language, mathematically oriented and very geek friendly. So there was a circle of people around that stuff.

And then eventually, after a year or two, the administration at Poly replaced the 7040, which they really allowed almost hands-on access to, with an IBM 360 Model 50, a commercial mainframe computer. They decided that, as befitting the exalted status of the IBM 360, they would create more of a robed priesthood to do the care and feeding of the computer.

They kind of pushed us amateurs a little bit out of the computer center behind glass and we were doing things [remotely], which made some of us unhappy. So we kind of wandered around the school and found in the corner of an EE lab at Poly a PDP-8 that I believe Digital had donated to Brooklyn Poly and was basically sitting there unused.

So we decided to use it. We kind of claimed it for our own. It had 4 thousand 12-bit words of memory, so that's a massive 6 kilobytes of memory. It had a Teletype that had a paper tape reader and punch connected to it, so that was the human interface. And it had a single DECtape. OK. After a little bit of programming it through the paper tape reader and punch, we discovered why it was sitting there unused. It was almost unusable.

So the first thing we needed was some kind of operating system that let us use the one tape drive and I actually wrote that. We called it the RL monitor and later it became the PQS monitor [after the Poly Question Society, one of the circles I mentioned before]. Some people continued to develop it after I left Poly and you can look that up on the web.

There's still a guy who develops and promotes it. His name's Charlie Lasner. He was one of the group and you can find websites of his about the PQS monitor that used to be called the RL monitor. But it was pretty crude. It had the world's crudest file system. All files were a fixed length, it wasn't interrupt driven, and it just had a couple of commands.

We took the digital assembler, which you used to have to load in binary on paper tape, and stuck it out on the DECtape, along with the image of the operating system. That was basically it. And then, of course, we had to figure out a way to copy tape.

So we created a tape copying utility. The good thing about DECtapes is the reels were only about this big and it wasn't some complex vacuum system. It was just two reels and the tape ran overhead.

So the way you copied tape was you ran this program and it read from one tape. You grabbed both reels, yanked them off and put them on the table. You would grab another tape, both reels, and stick it on the reels. It would write that. Then you'd yank that off and put it on the table and grab the original one. You would do that about 50 times and you would d have copied the original tape. It was slow but it worked, right? This gave us the responsiveness of not having to read your program on paper tape anytime.

We wrote an editor that would read in one of these little fixed-size files. Of course, it would fit in the memory of the PDP-8. When I say fixed size, I mean like 4,000 bytes.

You could then edit that file and put it out. When you went to run an assembly job, you'd list out all the files that concatenated together, form the source that you wanted to assemble, and the quote "operating system" would sequence through those files and kind of feed them into the assembler, which thought it was reading them from paper tape.

Then it would take the output of the assembler and stick it in some other file. So that allowed us to actually do software development, without spending your lifetime playing with punch tape. A bunch of us, - the core there (I'll probably leave out names) - was myself, Jack Burness, Lenny Elekman, and a guy named Charles [Kamen], who was a graduate student at Poly. It was kind of his machine.

He wasn't as quite as technical as us but he was included in the group, because we had to keep him happy, since it was his machine. In fact, he later wound up at Digital also. It's amazing. A guy named Roy Folk, who was a graduate student, also wound up at Digital and then came out here [to Silicon Valley].

Poly did not offer a computer science degree, so I was a math major.

Jack and Lenny were EE majors. There were some computer science courses offered but at the beginning, I think there were hardly any. As we went from 1965, when I arrived there, to 1969 when I left, Poly started to recognize that computers were important and got some professors in who could teach computer science language courses but no real operating systems, networks or anything like that.

So we were largely self-taught. The Poly library had access to documents and we'd go out and read about the hot languages at the time, which were things like LISP and SNOBOL. Then we'd implement them in 6,000 bytes on a PDP-8 with our one tape drive.

Burniece: Ok, now Richie, I have to ask you, did you ever go to class?

Lary: So …

Burniece: What were you doing to graduate?

Lary: Ok. This is a little embarrassing. I basically wound up working 60, 70, 80 hours a week on that PDP-8. I lived there. The whole bunch of us did.

Of course, I was a math major and I enjoyed math, so I did go to my math courses. As for the other courses, in my spare time I had actually joined a fraternity at Poly. Because I recognized that I was in danger of becoming a serious nerd and I didn't want to become a serious nerd. So I joined a fraternity.

It was a fraternity of actually fairly bright, nerdy guys, who also didn't want to be nerds completely. But one of things that happened is that three separate people in that fraternity figured out three separate ways to get master keys to the school.

So I had access to a master key. For my non-math courses, I think I took a EE course or two, just because it seemed like a good idea, but for the rest of the courses, I cheated. Ok.

I'm not advocating that as a lifestyle but I consider that I got an incredible education at Poly. But it was more of a, I don't know, Waldorf School kind of self-directed thing, rather than a curriculum-driven education. In fact, one of the big time wasters, as far as I was concerned, at Poly was a gym class.

You had to go to a gym class to get graduation credit and you can't cheat on gym class. There's no final. So to actually graduate Poly, I bought an insurance policy from the fencing coach and, in return, he retroactively put me on the team for two years. I've never fenced. But this was Brooklyn and you could do this sort of stuff.

So it's not the kind of thing I want my kids to be doing, unless they found a passion like I did that wasn't available at their schools. But it's what I did and it worked out great for me. I don't advocate it for everybody.

Burniece: It's a great story. And I've heard it before in some pieces, but a lot more detail this time. So tell us how you parlayed that experience from the time you were in high school all the way through Brooklyn Poly, basically becoming an expert in programming and understanding …

Lary: Well, self …

Burniece: …. computers, into a [career]

Lary: Expert's a bad word, because academically my credentials are actually fairly poor. I did graduate with a bachelor's degree in math and I was in a four-year master's program at Poly. I had skipped a grade in elementary school and had 28 credits towards a master's.

Burniece: Ok.

Lary: But I never did get the master's. After four years, I was done. I wanted to build things. I was building things all the time at Poly. I did most of the work on our SNOBOL.

Burniece: What's SNOBOL?

Lary: SNOBOL was a string processing language, kind of the PERL of its time. It allowed you to do pattern matching and replacement on strings. It was a very powerful language at the time.

It was a non-computational language, dealing with abstractions, you know: strings, text replacement, and text searching. So it was interesting. Somebody else did the LISP for the PDP-8, a full implementation of LISP in 6K bytes.

Some of us in the computer center, which we hadn't completely abandoned (it was just not the center of our orbit anymore), had gotten into something called a Stromberg-Carlson 4020, which was a very early graphics device. It was non-interactive graphics.

You could go back into the tiny room where it sat and there was a screen maybe this big (sorry, this big) -- a high-resolution cathode ray tube. And right above it was a camera. The purpose of this graphic thing was to produce microfilm. Right?

So you could do movies. They'd wind up on the microfilm. But it was no good for interactive graphics, since there was no input terminal anywhere near it. So you couldn't play Space War on it or anything.

But Jack Burness kind of took the lead in playing with the graphic style. Lenny and I were his assistants. In fact, one of the things we did at the time - this is kind of an early example of open source – was the University of Waterloo had this Fortran compiler called WATFOR (Waterloo Fortran). I think one of the professors at the school actually suggested that we do this. He said, you're so interested in graphics. Fortran isn't really designed for graphics but why don't you add some extensions to Fortran to implement like a vector data type? .

Then you could have a plot command that would plot a vector on the screen. Because that's all the 4020 could do, put a dot on the screen at a particular point. So in fact, we got the sources to Waterloo Fortran and turned it into a graphics version of Fortran. We called it WATFIVE, because we were college students at the time. We couldn't think of anything more clever. Actually, [Lenny] wound up presenting a paper on it. [Note: the Waterloo folks later did their own WATFIV, no relation to ours]

Burniece: Which Lenny are your talking about? Is that Len Shustek?

Lary: This was Lenny Elekman. Although, talking to Len Shustek earlier, I believe he credits me with teaching him how to program. He said the vehicle for that was the matrix multiplication routines that we needed.

He remembers it as being in a BASIC. I remember it as being in WATFIVE, because if you can add vectors as an extension to the language, you needed to do the dot products and cross products. Also you needed to multiply vectors by matrices.

Burniece: Just to add a little context to that by way to the video here, Len Shustek is the Chairman of the Board here at the Computer History Museum and was a few years behind Richie at Brooklyn Poly.

Lary: Right.

Burniece: Len met with Richie for a few minutes just before this interview.

Lary: He is also one of the people who was instrumental in bringing the museum to Silicon Valley.

Burniece: Absolutely.

Lary: You know, it was good to have some shared history with him.

Lary: We'd been using this timeshare version of BASIC, because BASIC was a language that was easy enough that we never took a course in compiler design. We learned, for example, from WATFOR how to do compiler design.

We, mostly I, decided to do a BASIC for the PDP-8 that was an actual compiled BASIC. So I wrote what became Poly BASIC, which was a full-featured BASIC data compiler. It didn't compile into PDP-8 assembly language, because that was pretty much impossible.

PDP-8 assembly language - I'll get back to that - was a marvel of what you could do with a 12-bit instruction. But it was not easy to generate machine code from a high level language. So we created a little interpretive language that we compiled BASIC into that was low overhead interpreter, kind of threaded code; if you go back, you'll find it [Threaded Code] had a different meaning back then than it does now, and that took quite a long time. That probably occupied me for most of a semester, along with all the other crap that I was doing.

But I wrote this compiler. It was a multi-stage compiler that produced code and then there was an interpreter that ran the code.   There was also the runtime system that supported the code.

Ultimately, when I graduated, that was what I took with me on my one and only job interview post-college, which was to Digital Equipment Corporation. It was the only company that I would consider working for, because I wanted to work for a company that built such a cool machine.

I also brought my resume, which was actually in matrix form. It was the machines that I had programmed and languages I had programmed in. There were Xs in the places where I had programmed in that particular language on that particular machine. That was my resume, plus a DECtape with Poly BASIC on it, and they pretty much hired me on the spot.

Burniece: Who did you interview with? Do you remember?

Lary: Probably my first bosses at Digital, which was a guy named Chuck Conley, who ran the PDP-8 software group, and a guy named George Thissell, was involved in PDP-8 software, but also PDP-12 software. The PDP-12 was a lab machine that Digital built, which was a PDP-8 with a laboratory coprocessor on it that had A to D converters and things like that.

And probably with a guy named Larry Portner, who headed up software in general--

Burniece: Right.

Lary: Those are the people that I logically would have [interviewed with]. I don't remember those interfaces[interviews?]. There was also a guy named George Berry, who was responsible for another part of PDP-8 software and who, in fact, I did eventually work for at one point in my career.

Burniece: Do you remember what your first job was there? What you actually worked on first?

Lary: Yeah. So my first job was to take Poly BASIC, spiff it up a little and stick it in the DECUS library - We decided not to productize it but called it DECBASIC.

DECUS was the Digital Users Society and there was a lot of code sharing going on there. So we just made a freebie to give away to people.

Before I head on into the Digital stuff, there are maybe a couple of things more about Poly to mention. However, I'm going to sit there and stare into space for a while, because I've gotten derailed.

Burniece: You can also fill that back in a little later but if you do have a couple more things about Poly, [go ahead.] That sounds like one of the more interesting colleges in the country at that time frame, letting its students actually

Lary: Yes. They were …

Burniece: … have hands-on access to computers for [experimentation and self-learning].

Lary: … either incredibly lax or incredibly far-sighted in doing what they did, which was to give a bunch of undisciplined teenagers access to what at the time was a very expensive piece of equipment. That PDP-8 cost $18,000.

Burniece: Wow.

Lary: The other thing that they had and I'm much less proud of this in retrospect. In fact, I'm saying this in the Computer History Museum, so I may be executed, but Poly also had a collection of older computers at the time.

The one I remember was a Bendix G-15, which was a drum machine. It had maybe four locations of real memory and all the other memory was on a drum. You programmed it by writing an instruction and then also specified where the next instruction was, so you could optimize your program with the next instruction under the head of the drum just as this one finished.

This was like from the 1950s and they had several of that ilk machine up in a unused room, which they also allowed us to use as kind of a bullpen. As kids will do, we developed a game called chair hockey, which involved a basketball and a bunch of rolling chairs with backs removed.

You go sliding around the room, kicking the ball trying to get a goal and we broke all the switches off of these ancient museum piece computers. Because we were from Brooklyn and we had no respect. I feel much more ashamed about that than I do about cheating on my non-math courses.

Burniece: I think your apologies are long since accepted.

Lary: Ok. As things come up about Poly I'll …

Burniece: That's well worth it, because that's a really rich story, much deeper than I've heard before.

Lary: Ok. so when I first joined Digital, PDP-8s were in production. Digital was working at the time on the PDP-8E, which was the integrated circuit successor version, I think. Or maybe PDP-8I was the integrated circuit successor to the PDP-8, plus much smaller.

You know PDP-8s were the size kind of maybe a small home refrigerator cut in half vertically and weighed about 250 pounds.

Oh, here's a combination DEC and Poly story. After I was at Digital for about three years, I went back to Brooklyn Poly on a recruiting trip, because I still knew some people at the school and still had some cachet there. Digital was very hungry for talent.

So I went back to preach about how wonderful it was to work at Digital, which it was, to the new students at Poly. I brought my checkbook with me, because I was going to buy that PDP-8 and take it home with me on that trip, because it had basically changed my life.

I was going to be a math major but it wasn't clear how you made your living as a math major. My dad wanted me to become an insurance actuary, because he knew that was a career that needed a lot of math, and he was very disappointed when I left before getting my master's. It took about 8 to 10 years before he finally grudgingly admitted that I probably didn't do so bad after all. He wanted me to get my doctorate.

So I went back with a checkbook, gave my talks, and then went over to the EE Department that owned the PDP-8 and announced to them that I wanted to buy it, at which point they informed me that it had been stolen. Somebody had walked out of the school with 250 pounds of historic computer equipment. This was Brooklyn.

So anyway, back to Digital. At the time, and it turns out this has driven an awful lot of decisions, memory technology improved and the cost of memory had come down. The PDP-8 was a 4K machine. This was the minimum amount of memory you could get on it.

Because it was a 12-bit machine, 4k 12-bit words was the maximum amount of memory you could address. A 12-bit pointer only goes up to 4k. It did have a memory extension command set, which would let you manually specify the next three bits of the address, so it could technically go up to 32k.

But memory was so expensive that this was kind of theoretical, except that memory prices started to come down, so we started to ship PDP-8s with more than 4k of memory. But we had a really primitive operating system for the PDP-8, quote "operating system" with no interrupts. It was more of a program development system.

Now that PDP-8s were shipping with more than 4k, we needed a program development system that would utilize up to 32k. If the system could use 32k and the assemblers and the compilers would support 32k, then maybe people would buy up to 32k of memory. So Chuck Conley's group, that I was part of, was tasked with doing that.

Our first idea was to go out and see if there was anything being done at the universities. It turns out that the University of Michigan had a PDP-8 and also had a big IBM 360, model 67 time-sharing system. One of the hackers there had built something at Cooley labs called CPS, the Cooley Programming System. OK.

It was something that would use 8k of memory and kind of emulate what it was like to be on a terminal of a 360-67 time sharing system, kind of culturally compatible. They sent me out there to evaluate that, so I went out there and played with it.

It was pretty nice, so we bought it. We actually traded them some hardware for it, so was pretty cheap. We then took the sources back, looked at them and determined that it just wasn't suitable for us. Everything was hardcoded for a particular hardware way of doing things, using 8k. but it couldn't use any more than 8k.

At the time, Digital's flagship machine was the PDP-10. I fell in with a bunch of people who were the hardware guys and the software guys. There was a guy who graduated a year before us at Poly named John C. Green Jr, who was hired by the PDP-10 engineering group. He was friends with guys like Alan Kotok, Dave Gross, Bob Clements and Alan France. These were all PDP-10 hardware, software designers - a very collaborative bunch of very bright people.

They were known as the MIT mafia, because they came kind of from MIT over to Digital as a group. Those of us who went from Poly to Digital, which included myself, Jack Burness, Lenny Elekman, a guy named Stan Rabinowitz, and a guy named Herb Jacobs, were the Poly mafia.

We were the second wave of college kids who were going to come in and transform the company. So I started hanging out with the PDP-10 guys. And the PDP-10 was an incredibly impressive machine to me. It had time sharing. Yeah.

Burniece: Before we lose this track, you mentioned a number of people I know you really respect a lot.

Lary: Yes.

Burniece: Pick a couple of those names, like Alan Kotok or someone like that.

Lary: Right.

Burniece: and just talk a little bit about what they're all about and why they influenced you. As you and I both know, Alan was one of the key guys that created Star Wars on the PDP-1 and ….

Lary: So interestingly enough ….

Burniece: … he's now gone.

Lary: Yeah. I figured you'd want to know about that kind of mentors.

Burniece: Yeah, right. So who are the ones you really respected?

Lary: Digital didn't really work on mentors, because it was[n't?] hierarchical in that sense, right? You just found people whose style you really liked and whose intelligence you really respected. Then you'd try to emulate in certain ways.

I'll be coming back to this, because it will be piecemeal. My mentors at Digital were guys like Gordon Bell and Alan Kotok and later, a guy named Tom Stockebrand who was actually a mechanical engineer. But the interesting thing I wanted to bring out about that is Gordon Bell and Alan Kotok designed the PDP-8. Tom Stockebrand designed DECtape. So it's like form follows function.

Burniece: Yeah.

Lary: It's like before I ever knew these guys, I knew how bright they were--

Burniece: You knew what they'd done.

Lary: I knew what they had done. I knew their work but I don't know if the work influenced me to look like the guys. No.

They were all very likable. They all projected a certain energy and intelligence. They were also the kind of guys that if they believed A and you believed B and you were going to be in a room with them arguing whether to do A or B, you'd better have your facts straight around B.

You better think long and hard about why B is better. Because they had their facts straight about why A was better. Everything was very well thought out in advance.

They were all, I'd say, more systematic thinkers than I was, especially Gordon and Alan, who came from more academic backgrounds. I really was part of just a bunch of guys who would spur each other on to learn things, as opposed to being mentored by computer science professors from the previous generation. So it was those kinds of interactions with these people that caused me to learn and not so much a formal mentoring relationship.

Burniece: I know for a fact you also won their respect - you certainly did …

Lary: Yeah, sure. Because we were young …

Burniece: and did something with their machine.

Lary: We were young hot shots and the good thing about the DEC culture was there was no resentment [among] the older guys around the young hotshots. There never really was. You go, you go, right? And it was my turn to do that when more young hotshots came up.

And so while we're talking about mentors, I'll do names and we can come back to them. So for sure Bill Strecker later on, especially in terms of VAX architecture and other things, plus a guy named [Jega Arulpragasam], who I don't know if you ever knew, Tom.

Burniece: No.

Lary: But I did some advanced development work with him and he taught me a lot about hardware, because it was a hardware and software advanced development project. A guy named Mike Riggle, who I know you're very, very familiar with, and there will be more as I go on.

The thing about this was they were all people who had deep technical knowledge and at the same time had a refreshing lack of arrogance around [their] technical knowledge. [They also had] an ability to bring the technical truth out collaboratively in a meeting without butting heads, and stuff like that, or making the people who weren't at their level feel bad about the fact that they weren't at their level. I learned a lot from them - not just technical stuff, but also how to deal in a community of people with different technical ideas and different levels of technical skill, [then] bring the laggards up to speed without making them feel bad. So the focus was on getting the job done, rather than being the smartest guy in the room, et cetera.

Burniece: I can absolutely relate to that, knowing most of these people, and I know they also could think outside the box.

Lary: Yes.

Burniece: Every one.

Lary: Many of them had invented a lot of things in the past and broken with tradition in their own way and it was a great bunch of people.

Burniece: OK.

Lary: Anyway, going back. Sorry, it was a big digression, but we used [a PDP-10], in the PDP-8 group, even though I had this little operating system that I did [at Poly]. The PDP-10 had a real operating system, with a real file system and real tools, so I would do most of my development on the PDP-10 for the PDP-8.

Probably the CPS experience was in my mind, because when CPS turned out to be not suitable, I decided that I would take the one thing from them that was really valuable, which was the idea that a standalone minicomputer terminal should look just like a time-sharing terminal on its big brother computer. So I set off to implement a system that would later become OS-8.

Burniece: Was that a unique idea at the time?

Lary: Well, CPS certainly did it before me. I don't know that it was widespread. There weren't an awful lot of companies that were making interactive computers that spanned any kind of range. I mean, Digital was really a leader in interactive computing.

All the IBM computers that I had used, including all the 360s, were batch machines. You'd queue up batch jobs. So the 360-67 was IBM's kind of redheaded stepchild in that regard. And later, IBM’s VM [Virtual Memory? Yup, an O/S for the IBM 360 series] stuff was another redheaded stepchild that ultimately caught on and you know, VM was a success, too.

I set out to build an operating system that one, would handle a variety of devices, so it had to have, for instance, loadable device drivers and a formal device driver interface, which my old operating system didn't have and CPS didn't have. Two, it would have a real file system - not super real, I had 6k to work with or, in this case, 12k, 8-bit words minimum, and it would look just like being on a time-shared terminal on a PDP-10.

Because I was 20 or 21 at the time, I called it the First Upward Compatible Keyboard Monitor, which has an obscene acronym. We were going to do the same thing for the PDP-9 and call it the Second Upward Compatible Keyboard Monitor, which had a matching obscene acronym that wound up in comments of the code. In fact, the DEC library knew that if the code was coming from the Poly mafia, they would run it through a text editor and search for the obscenities and take them out of the comments and the variable names. I got the hint and stopped doing that. Jack Burness started writing his obscenities vertically.

Burniece: So they couldn't track the acronym.

Lary: The first letter of each comment would spell it out as acronyms and the text editors couldn't find that but I just figured it was a losing cause, so gave up and cleaned up my act.

So what started out as F**KM, ultimately became the PS-8, the programming system for the PDP-8. It was remarkable in a couple of ways. 8K is not a lot of memory. So the kernel of the PS-8 used [just] 256 words of memory.

That was it. You could use all of memory, other than those 256 words, and those 256 words were basically some translation tables that abstracted device names and things like that, plus a disk driver. One of the things about the PDP-8, and this will come up later, was very true at Poly.

In the PDP-8 architecture, and this is no failing on the part of Gordon Bell and Alan Kotok, the way you had to do things, if you were going to design an architecture for 12-bit machine, is you had some very restrictive addressing. The machine was organized into 128-word pages - not pages, like virtual memory pages, but pages form the point of view of addressing.

If your code fit in 128 words, it was great, but if your code fit in 129 words, it got a lot more awkward, because you had to do this inter-page addressing to get the pieces, which would then blow up your code a lot more.

So, if it was 129 words, it might as well be 150. We only had a limited amount of memory, so all of us guys at Poly became very good at taking n lines of code and turning it into n minus 1 lines of code and repeating, as necessary.

So when we were writing things at Poly, and then later when I had these guys help for PS-8, I'd write a version of a routine. Also, this architecture encouraged modularity, so you'd write your code in chunks that would fit in 128 words, which means they'd be small.

We were into modular programming before it became common. The architecture forced us into it, so I'd go and write a routine and it would be 132 words long. I'd sit and scratch my head and look at the code. I'd get it down to maybe 131 words and then I'd give it to Jack. He'd sit there and scratch his head and figure out a way to get it down to 130 words. He'd give it to Lenny Elekman and he's get it down to 129. Finally, it would come back to me and I'd look at what they did and scratch my head some more and get it down to 128, so it would fit. Then we'd have a routine that worked.

Burniece: How long did that all take?

Lary: A lot of time but, on the other hand, the result was much improved and we all became really good at writing dense, fast, code. That has warped my engineering design philosophy.

Because I have a bias towards dense, fast, code, it also kind of reinforced an existing tendency that I had to not so much deal in abstractions as in concrete things. We used to joke later on when object programming and high level languages came in. Although, I use high level languages these days, people just don't understand the value of a bit and the value of a microsecond anymore.

So anyway, our management, which was very cost conscious, didn't really want us developing an operating system for the PDP-8. So [with] Chuck Conley, who was actually a very kind of neat, buttoned down guy, I explained to him what I was doing and I showed him an early prototype.

He went up to management and bold-faced lied to them - told them that we were making some modifications to CPS - but because of the limitations that we had found in CPS, it was going to take a little while, a little longer than we thought to modify CPS into something that Digital could offer to its customers.

Meanwhile, we had completely thrown the code away and wound up offering them this thing that I wrote, which was called first PS-8. Then that we changed the name to OS-8, not because it had actually grown into a full-fledged operating system.

It never did turn interrupts on; it was really a program development system. The reason it changed its name from PS-8 to OS-8 is that this was around 1972 and Nixon had imposed price controls, wage controls. You could not raise the price of a product, without going through all kinds of federal paperwork. So we changed the name and then we could raise the price, because it was a new product. Ok?

Burniece: That's a great story.

Lary: So anyway, we did that. Some of the other things I worked at in my early days at Digital - you've got to understand at the time I did have a girlfriend in New York, but I'd only go in there about every second weekend or third weekend.

In the weeks that I didn't go into New York, I was working 90 to 100 hours. I lived less than 100 yards from the entrance to the building. Digital was built on a hill and at the very top of the hill, there was a parking lot. On the outside of the parking lot were two little houses and I was in one of those houses. I rented one of those houses, along with at various times, John C. Green Jr, Jack Burness and Lenny Elekman.

Burniece: Now, you're not talking about Maynard?

Lary: I'm talking about Maynard.

Burniece: So were you 100 yards from Maynard?

Lary: I'm talking about Thompson street in Maynard.

Burniece: Wow.

Lary: It was off of the Thompson street parking lot at the top of Digital. I would get up in the morning and would roll down the hill into “the mill” building. I wouldn't leave sometimes until the next morning.

Because it was all-consuming, it was such fun, productive, satisfying and all of that. Because it was early enough days in the computer industry that we could be doing stuff that hadn't been done before, it didn't take a team of 50 to do it. And it didn't take as much standing on the shoulders of giants, as happens now where everybody's building on things that happened before. We could do things in an assembly language on bare iron that had never been done before and it was great.

Burniece: Ok, now, I want to get to the VAX in the next 15 - 20 minutes. But before we do that, any other stories in that first few years at DEC from the time you joined it in '69 until you started on the VAX, about the PDP-8, PDP-10, or anything else that makes some sense?

Lary: Sure.

Burniece: [INAUDIBLE] people?

Lary: I got one or two stories about it. So Tom Stockebrand …

Burniece: Yeah.

Lary: … was a mechanical engineer. And he became a friend. He's still a friend. He's 83 years old now. And he still radiates energy.

He doesn't do anything without gusto and so he hated software, because it was complicated - he hated things that were complicated.

He used to tell software guys, you guys always wanted to one plus everything. I ask you to do this and you don't do a good job on that but you give me six features I don't want, blah, blah, blah. He didn't have a software group that helped him out. When he needed software, he'd kind of troll the halls, you know, looking for people who might be willing to use some of their copious spare time to help him.

So I volunteered one time and I went over to interview with him. Tom was a good-sized guy and he is a mesomorph. His face looked like, at the time, chiseled out of stone and he had the heavy brow kind of Neanderthal brute look to him.

He comes striding into the room, grabs a chair, sets it backwards in front of me, straddles the chair, leans into me, and says, what are you good for? This is his interviewing technique. Homina homina... [old Jackie Gleason schtick from The Honeymooners indicating being struck speechless]

Burniece: I know Tom well. This is fun.

Lary: Huh?

Burniece: I know Tom.

Lary: Homina homina homina and I wound up doing some work for him. He had this monstrous high-speed paper tape punch that would do 1,000 characters a second or something like that and you had to wear headphones around it to block the noise.

He needed somebody to write some software for it to get it to do right, because, of course, the tape was zooming through there so fast that it could screw up, right? Anyway, I did some work for him on that and later, just as an aside, Tom went on, became our video terminals guy. He built a terminal called the VT 50, which was kind of a failure but it worked pretty well.

It was a little too expensive and didn't have the right resolution and things like that. We sold a bunch of them but then one of things about Tom is, you know, he's always learning. He learned from that and built something called the VT 52, which became I think the best selling terminal in the world at the time, and then followed that up with the VT 100 ….

Burniece: Which was?

Lary: … which was the dominant computer video terminal in the world. Then he got tired of the quote "politics" in Maynard and moved out to Albuquerque. I lost track of him until I moved out to Colorado and then it was only a five hour trip and we'd kind of go see each other periodically. He's still one of my heroes.

Burniece: Yeah, I know a number of great stories about him, which I won't get on tape today, but what I want to ask you about, was I always told that he used to …

Lary: He is available for a video interview.

Burniece: Oh, we'll probably do that.

Lary: He'd be a great subject.

Burniece: We definitely want to do that. One of the stories I was always told, and I wasn't around in the days, was that he would canoe into the …

Lary: Yes.

Burniece: … the mill and go in through the windows.

Lary: Tom was an alternative energy guy from way back and iconoclast from way back. He drove an electric Volkswagen in the '70s and he liked to canoe, so he would put his canoe into the mill pond and canoe over to where his office was, where he'd canoe up there, open the window, climb in, tether the canoe outside his office, and [go inside]..

In fact, Digital, because we did manufacturing in that plant too, security was sometimes a little obstreperous. They thought they ran the company, so Tom bet them that he could get into the building any time without going through security.

They took his bet and he just canoed up to his window, went up there to security from the wrong direction, and said, "hi, you guys". I took a page out of his book later in Colorado by taking the hinges off of doors that were supposed to be locked.

Burniece: I remember that.

Lary: Yeah. You were there at the time. You had to support me in that.

Burniece: I remember that very well.

Lary: But, anyway - yeah, [we went] through the early history really fast. Again, at Digital it just became another continuation of learning by doing. Digital decided they wanted to get into the commercial software business.

There was a guy, and I wish I remembered his name, who invented a cut down version of COBOL called DIBOL, Digital's Interactive Business Oriented Language, and he wrote the first compiler for it. Then I and a guy named Stan Rabinowitz, who was another Poly grad, and one or two other people were tasked with turning that into a commercial system, so we migrated it over to OS-8 and I wrote a sort for it. Stan wrote a field-oriented graphics input that emulated these kind of IBM terminals that you could only type in certain fields and, if the field required a number, you couldn't type anything but numbers in - that kind of stuff. So we built something called the Commercial Operating System, COS, which was a variant of OS-8 for commercial users.

Burniece: Was that the very first Digital commercial operating system?

Lary: Commercial product? Well, on minicomputers, yeah. I think they had stuff - had a COBOL on the PDP-10 - for people who wanted to do that kind of stuff but minicomputers and commercial were kind of non-intersecting worlds at the time.

So it was moderately groundbreaking there. And again, a little digression for my sins. After I moved to Colorado, I became involved in a rental company, a place that rented tools, skis and stuff like that we wanted to go automate. We found some company in Kansas that wrote software for rental companies and we got their software. It included a thing saying we'd get the source if they went under. Sure enough, they went under and they shipped us their source. It was in DIBOL and for my sins, I got to maintain a large commercial package written in this language that I helped promulgate.

Burniece: That's great.

Lary: So I did that - did a real-time system for the PDP-8 - because early on in its history, we decided PDP-10 did time sharing, the PDP-8 should do time sharing. A bunch of engineers at Digital, and I wish I remembered their names, because they were all very good, implemented a time sharing system on the 8 [Nathan Teichholz and Mark Bramhall were two of them]. In order to do that, they had to add in the equivalent of a hypervisor bit to the PDP-8, where any I/O instructions would trap down to a kernel.

Burniece: Just as a reminder, we will footnotes some of the names, if you remember them later …

Lary: I'll try to come back to them.

Burniece: … into the transcript, because that's some good stuff.

Lary: I have some people I can call. You know, there's a chain.

Burniece: OK.

Lary: And so I built a real-time system for the 8 [This was in 1974]. Because I'd never built a real-time system before, and PDP-8s were being used a lot in real-time, it was all ad hoc.

So I built something called RTS-8 and I got this great idea - I used that hypervisor bit and you could run OS-8 as a background task for RTS-8. You could do the development for RTS-8, while you were running RTS-8 in the background, so I got all kinds of experience that didn't come out of lectures and books. But it was like, hey, why don't we do this? I'd go research how to do this, then go do it, and eventually, I got involved in some of the PDP-11 stuff.

Burniece: OK. Talk about that a little bit.

Lary: There were other people doing PDP-11 software, a guy named Bob Bean, a guy named Brad Miller, and a guy named Mark Bramhall, who was really the spark plug behind the time-sharing system for the PDP-11 named RSTS. So I got involved, not in a primary role, but kind of in a helping out role there. I was never a primary PDP-11 programmer. The PDP-11 was another Gordon machine - Gordon Bell machine.

Burniece: There are a number of people who'll listen to this or will look at the transcript who don't know a lot about the difference between a PDP-8, a PDP-10, and a PDP-11. Can you kind of give a quick …

Lary: So at the beginning …

Burniece: … comparison?

Lary: … of the computer industry, there was a kind of chaos around word sizes. Most of the machines actually around were multiples of 6 bits in length. The common numbers were 12, 18, and 36. And there were some at 24. Ok?

If your multiple is 6 bits in length, you'd use a 6-bit character set. The problem with a 6-bit character set is you could hold the capital letters, the numbers and a bunch of punctuation marks but you couldn't get [both] capital and small letters.

Burniece: To differentiate, that's not what we call a byte today, which is 8 bits.

Lary: Correct. It was a character and when the [IBM System] 360 came out there were other machines from smaller companies. Xerox had their SDS series and a couple of small companies whose names escape me in Florida and other places also built 32-bit machines and 16-bit machines and they had 8-bit bytes. You could do a lot more with 8 bit bytes; plus you had control characters. You had small letters. You had a lot more punctuation stuff. You had escape characters that would let you get into even bigger character sizes.

It became very clear that that was the way the industry was going, plus, you could address a lot more memory. Again, a lot of this was driven by memory cost. So Digital decided to build up a 16-bit machine and they were doing it, I think, as I joined in 1969, June of '69. Right around Woodstock was when I joined. I missed Woodstock, because it was my first week at Digital.

I think they shipped their first PDP-11 around 1970, maybe '71, and again, it had kind of a weird little disk operating system that was very slow.

Some of the PDP-11 software guys said, hey, we got the system on the PDP-10 that looks like this and we've got the system on the PDP-8 that's culturally compatible with it, has the same commands, the same file name syntax and stuff. Why don't we do that on the PDP-11?

That's how this system called RT-11 was born. I didn't have anything to do with it. It was built by guys like Bob Bean, Brad Miller, and I think a guy named Anton Chernoff, plus a couple of other guys who were the PDP-11 hacker group.

Burniece: Was Dave Cutler involved in that one?

Lary: Not initially, not initially. He came a little later and he was more of a real-time guy. RT-11, despite the fact that it had real-time in its name, was really more of a simple program development system.

It did have interrupts. You could write interrupt-driven programs but it didn't have a concept of multitasking or anything like that. When Dave came along, he brought with him a legacy of designing priority-oriented, real-time systems.

I'd list him on a list of mentors, mainly because he could code like ringing a bell. And his code was clean, small, and fast. And it was all things I admired in code.

Burniece: Expand a little bit more about Dave, because he's a key player here in the next part of the story.

Lary: Yes.

Burniece: So tell us a little bit more about Dave – what was he like, what he did, and the capabilities he had …

Lary: What Dave was like?

Burniece: … his personality.

Lary: Originally, I think he must come from a non-interactive programming background.

Burniece: I think he came from a biology degree or something …

Lary: Yup.

Burniece: … or something like that.

Lary: He would actually write his code on coding sheets, which were pieces of paper with little tick marks in them like you see when you've got to enter your name on a form that's going to be transcribed into a computer. He would code on something else and somebody else would enter the code.

But he was one [of the] great operating system guys - understood a lot kind of intuitively about how to synchronize different tasks going on, in a uni-processor, later in a multi-processor, and had a very real-time approach to things. So if you needed to build something, he'd usually build a real-time kernel for it, and then build all the other functions up on …

Burniece: Just for a footnote on that.

Lary: … top of the kernel and he was an extremely forceful guy. We used to call him a "drill Sergeant".

Burniece: Right.

Lary: [As for] his attitude, he eventually [moved into] a supervisory role and he had a very, very clean management style. He basically said, you know, if you can't do your job, I'll do your job and my job, too. Which was a very powerful way of motivating people.

Burniece: I heard that many times. Just a footnote here, Dave is probably considered today one of the most famous operating system guys of all time, because he did not only RTS, or whatever it was, …

Lary: …and VMS.

Burniece: …. but he did the VMS …

Lary: … and Windows.

Burniece: … kernel and he yes also did the NT kernel for Windows

Lary: Right. Right and he branched out. He did some languages.

Burniece: Right.

Lary: He did some languages for Digital. He did some CPU design. He's a full-spectrum guy. It's just that what he's known for is operating systems.

Burniece: And he's still working on the X-Box.

Lary: And he's still working. He and I didn't intersect that much, because he was always on PDP-11s and I was always on PDP-8s. I got involved in PDP-11s, but in a different way I'll get into.

So I helped out a little. And I do have a story, because you said you wanted some anecdotes. So at Digital, I was helping out on RSTS and one of the ways I was helping out on RSTS is Digital had just built a machine called the PDP-11/70, which was a big address machine, even though a single program could address only 16 bits of memory. This thing could take 4 megabytes of memory, so you could have a lot of different 64K chunks of memory.

It also had what's called I and D space, a separate program from data to get around the 16-bit limitation, which we'll get back to later. I was helping modify RSTS to handle this big address space and, as a result, I spent some time on that hardware.

So this guy named Dennis Ritchie from Bell Labs, who was one of the guys who invented Unix and C - We had a relationship with them, because one of [the Unix] targets was originally PDP-11s - and we told them about this new 11 we were building,

So he came out and we gave him a manual. He made all the changes to Unix to run on the bigger address space and with I and D space.

He [then] came out to Digital to use one of our prototypes to test it out on and I got to be his minder. I met him at the door, escorted him over to PDP-11/70. I left him alone there - gave him some privacy, while he did stuff.

He did his stuff for three or four hours and got it to work. I came over, when it was time for him to leave. I said, you know, you better delete your files, because you don't want to leave this Bell Lab proprietary stuff on our computers.

He said, good point and he executes one of these UNIX recursive commands and deletes every file on the disk. We leave and I go back to the machine. I grab a tape drive and do a physical dump of the disc onto the tape drive, take the tape drive over to a PDP-10.

There was a very sophisticated text editor that we all used called TECO, developed at MIT, and I used TECO to search for the copyright statements at the beginning of every module, and basically got a copy of Unix, Bell Lab's Unix. We never did anything with it.

We never violated that copyright but we kind of looked at it - looked at the internals - and said “you know, this is kind of interesting; this is a kind of new”. We learned something from it. It was lovely, right?

You know, there was a core of about six of us who actually went and read those sources for understanding …

Burniece: Did Dennis ever …

Lary: … to see what the fuss was about.

Burniece: … figure that out?

Lary: No, we never told him. We did invite him back though, to the VAX architecture committee and I'll talk about that.

So anyway, because I was involved on RSTS on the 11, some time in early '75, I went down see Gordon. I had some dumb idea about something to do with RSTS and Gordon was the VP of Engineering at the time, or the Chief Technical Officer, one of those titles. I walk into Gordon's office and obviously, I had interrupted him in the middle of a “Gordon storm”, where he was thinking about ideas.

He had some stuff on the board and he said, listen, we got a problem. Memory's getting even cheaper and the amount of memory that we can support on a PDP-11 just isn't enough. We're going to be in a position where people are going to need more memory than we could provide and not enough of our system cost is going to be in memory. You have kind of a certain ratio of the processor is this much of the cost and the memory is this much of the cost.

He said, we've got to go to 32 bits, so I'm forming a committee and I want you to be on it - ok?. Again, right place, right time.

Burniece: So you actually walked in, interrupted …

Lary: That's how I wound up on the VAX architecture committee.

Burniece: … a Gordon storm and that's how you got invited to VAX?

Lary: I wandered into Gordon Bell's office, just as he was thinking hard about this problem and getting some passion around it.

Burniece: All right. Now, tell me about this [VAX architecture committee]. Who were the key people on it, and a little bit about each?

Lary: Ok - so the people on the original committee - I'll have amend this later, because I don't want to leave out any names. It's important. There was Bill Strecker, a PhD, who at the time I think was not long out of Carnegie Mellon and a very bright architecture guy.

Burniece: And a very structured guy, right?

Lary: Right. There was me, because I had a software background and this was going to be a machine that was going to be friendly to software. Similarly, there was a guy named Tom Hastings, who was a PDP-10 operating system and software guy, [who had] been around a long time He was a generation ahead of me in age.

Burniece: Also a little bit like you and self-taught or was he also …

Lary: You know …

Burniece: … a really structured …

Lary: … I don't know.

Burniece: … academic kind of guy?

Lary: I think so. He may have come out of the MIT community but he was older than the other guys in the MIT community.

Burniece: Ok.

Lary: There was Steve Rothman, who went to high school with me and was one of the hackers at NYU Courant Institute with me. He was a hardware guy and we needed a hardware guy on the committee.

Burniece: But he did not go to Poly, right?

Lary: He didn't go to Poly - he went to MIT.

Burniece: So he's an MIT mafia guy but …

Lary: No.

Burniece: … from your high school buddies.

Lary: He was half a generation removed from the MIT mafia but he still came to Digital.

Then, maybe shortly after it started, a guy named Dave Rodgers joined, who was also a hardware guy and very bright. Later he became, I think, the VP of Engineering for Sequent and wound up at Compaq. He has had a long history in the computer industry. He became basically the technical lead for the first VAX implementation. In fact, I think that's why he was put on the committee - he was going to have to build the damn thing. Ok?

Gordon came to the first few meetings, set the tone, layed out the strategic imperatives but he was usually not involved in the day to day stuff. He'd come by.

Burniece: So he acted as a leader.

Lary: Mostly, we would work. We would just find a room somewhere and go sit in a room and have bull sessions. What should be instruction set that look like?

Interestingly enough early on, Steve Rothman proposed an instruction set, which was radically different than anything that DEC had done previously. We kind of looked at it and went, well, it's an interesting instruction set but it wastes a lot of memory.

So we rejected it and, in fact, what Steve had - I don't think there's any record of this - but my memory of it is he was proposing a RISC machine.

Burniece: Is that right? That's fascinating.

Lary: The world wasn't ready for a RISC machine yet, because memory wasn't cheap enough for a RISC machine. The instructions were wasteful in terms of space and memory was still precious.

Burniece: Just on a historical note …

Lary: He proposed a RISC machine and we rejected it [but] it was the right thing to do at the time.[1]

Burniece: … had anybody done RISC yet?

Lary: No.

Burniece: That would have been unique.

Lary: The VAX committee was inaugurated on April Fools Day in 1975. That's my memory.

Burniece: April Fool's Day, that's great.

Lary: April Fool's Day 1975 is when we had our first kick off meeting and by the summer of '75, Steve had proposed this as a potential computer. Being a hardware guy, he proposed it, because he could see how to implement it in a small number of gates and make it fast. We rejected it just because it was wasteful of memory and that was the right decision in 1975 but by 1985, it was the wrong decision. You know? Just the way it happens.

[Once the eventual architecture was set, after we had kicked around a whole bunch of things, we started calling outside people in, outside experts. One of the guys we called in was Dennis Ritchie, because of C. We regarded him as one of the guys on operating systems, languages and machines who would be good to [have] compile languages into [VAX], because we wanted to make this machine as software friendly as we could. Unfortunately, it didn't go well and it wasn't Dennis Ritchie's fault. It was because we couldn't tell him what we were doing.

What we could tell him was we were considering extensions to the PDP-11, so he took a small view of the word extensions and came up with, "you know, it'd be nice if you added this instruction, it would be nice if you did this, it would be nice if you did that."

[Since] he didn't know that we were looking at a radical rethink of what a PDP-11 was, he thought we were looking at a natural extension, so it was kind of an awkward meeting with him, not because he's

[1] [Interviewee's note] Memory was a large percentage of our system cost, so doubling the amount of memory needed to hold programs would have put us at a cost disadvantage that the increased performance wouldn't overcome.

dumb. [It was] because we couldn't open up to him about what we really wanted. At which point, if we could, he probably [would] have had some much better suggestions. But this was the early days and it was a complete secret.

They organized the [larger] VAX group, I forget when, but there was something called VAX A, which was the core [architecture] committee, which is what I was on.

Burniece: There were like 6 guys on that, right?

Lary: I was responsible for creating the new architecture document. Then there was VAX B, which were guys who were allowed to read the architecture document and submit comments. They were people with a much broader amount of expertise in the company but they still got to see this thing on a need-to-know basis, because this was hush-hush. Then there was VAX C, who were guys who knew that something was coming, knew the general shape of it and had to think about how to plan to organize the company around this new machine, but weren't privy to the details.

That's how Gordon had set it up I believe. So we'd occasionally bring VAX B guys into it and talk to them and things like that but, at least in my recollection, the guy who came up with the key idea for the VAX architecture, which was how the instruction set (the ISP) should be organized, was Strecker.

Burniece: Ok.

Lary: He built an architecture that was superb at conserving memory, because instructions were variable lengths, so short instructions would be only 16 bits long, 2 bytes long. Then, because there were addressing modes, the same opcode would work on a 2-byte instruction or a 3-byte instruction plus 4-, 5-, or 6-byte instructions, as things got more and more complicated.

As it turned out, it was a very difficult machine to make run fast, because of the complicated nature. In order to fetch the next instruction, you had to figure out where it started and, because this instruction was variable length, as you got more bytes of the instruction, you could finally figure out how long it was.

It introduced a certain sequentuality in there but, given the constraints, which was we knew that the first VAX had to be a full commercial system with all kinds of bells and whistles, was going to ship with [only] 256K of memory. It was also a great architecture for compilers to compile code for, because operations and addressing modes [were] completely orthogonal.

Pretty much every operation you had an add operation and you could add register to register. You could [also] add register to memory and could add memory to memory. You could add memory with indexing, et cetera. There weren't separate op codes for that, so the compilers were a lot easier to write and we got through the instruction set.

Burniece: So you're saying that most of that architecture ultimately came down through Strecker.

Lary: The core of it, which was the overall structure of it, came from Strecker. When it came down to picking opcodes, a lot of us who had more programming experience than Strecker got to get our [ideas] in - oh yeah, this is an addressing mode that really cuts the size of the code down or cuts the number of instructions down, blah, blah, blah.

But the framework for the architecture, in my memory anyway, came out of Strecker's head. He just kind of got up on the board one day and drew it. We said - that's it.

Burniece: Now, just for historical reference, you say you started on April 1, 1975? When did that happen? Was it within a year, a couple months, or what?

Lary: Well, I can't give you a precise timeline. We started in April of '75 and I believe that Ken Olson announced the VAX to the public with a working model in October of '77. I'd have to go check that.

Burniece: I think that's about right but when did Strecker's revelation come ?

Lary: So there was a certain amount of overlap. I'd say Strecker's revelation came sometime probably in the late summer or early autumn of '75, after we'd been collecting a lot of data and starting to explore alternatives.

Burniece: So about six months or thereabouts - something like that.

Lary: and then we started to flesh in around that.

Burniece: Ok.

Lary: and then we also had to develop an interrupt structure for the machine, memory management structure for the machine, process structure for the machine, etc. Because this machine was going to be a micro-coded, we could do a lot of complex things.

We had instructions for decimal arithmetic, we had instructions to do locked access to memory, and I think we may even had instructions for due process context switching. Because it was microcode, you make your instructions as complicated as you wanted. The microcode will execute it. This was the anti-RISC philosophy - concentrate on getting the code crunched down and making it easy for the compilers.

The instruction set was well-received within the company and well-received within the industry. The hot thing in memory management at the time was something called “capabilities”. There aren't any machines right now, except maybe that 64-bit Intel processor that never made it (Itanium) with capability-based memory, but it was an extension of a virtual memory where pages could be marked as to what they would be used for.

There was a little bit of object stuff in it and we built this very interesting virtual memory system that was kind of recursive. The tables were self-describing and there was one entry at the bottom of the table that let you bootstrap into the rest of the memory structure. I probably still have the documentation for that somewhere.

Burniece: If you do, by the way, I'm sure the museum would love to see it.

Lary: Yup.

Burniece: So you are basically saying you had kind of an object-based virtual memory system?

Lary: Object is not the right word. But it had a little flavor of objects in it.

Burniece: But a self-describing system.

Lary: Which is that when you describe the page, you describe more than just here is it. You describe what they can do with it in certain ways and other things. Dave Cutler, who at the time was a member of VAX B, was charged with building the operating system, VMS, along with a guy named Dick Hustvedt, who came from Xerox.

The two of them were complete opposites. Cutler had a real-time background. Hustvedt had a time sharing background. Cutler was a bull, hard charging and stuff. Dick was a very laid-back guy who relied on intellectual convincing. He had wonderful ideas. Not that Cutler didn't, but [Dick] didn't rely on the force of his personality so much to project what you should do as to kind of just gently convince you.

They were a really good match for each other - this kind of team of opposites - to build VMS. But after we had circulated the memory management scheme, we were having a meeting and Cutler comes in with a couple of his guys. He says, “this memory management sucks. It's way too complicated. I ain't going to build a system with this, as the memory management. It's going to fall on its face.”

He gives us a half a dozen reasons. He says, “me and Dick came up with something simpler” and sketches it out on the board. That's the memory management system in the VAX. That's the virtual memory system in the VAX.

Burniece: Now what was different about it?

Lary: What was different about it was all the capability stuff went out the window - all the fancy bootstrapping went out the window. It was a very, very simple system where page tables for user space wound up in kernel space, which is itself virtualized, and it had a page table, so that you could page user page tables in and out. It was just a very simple, elegant system.

My guess is nobody really missed the features that [were] cut out of what we did. So it was the right trade off to do and that was Cutler doing what he does best, which is getting to the heart of the matter and concentrating on performance. We had gotten a little overblown in our ambitions and he called us on it.

Burniece: So from an architectural historical perspective, two of the key decisions that were made were made by Bill Strecker …

Lary: and Dave Cutler …

Burniece: … and Dave Cutler on how to structure this thing.

Lary: … on how to structure.

Burniece: Dave with the memory management and I guess Bill basically with the overall code structure.

Lary: … with the architectural framework - the meta architecture.

Burniece: At that point Dave was on VAX B, so he wasn't really a member of the core committee, but he was close.

Lary: After that, we started including him more in VAX A. It's like, oh, you know, if he's going to come up with ideas like that, hey, Dave, come to the meetings.

Burniece: Before I forget it, because he's such an important person, talk a little bit about Hustvedt and what happened ultimately to Dick.

Lary: So Dick Hustvedt came to us from Xerox and wound up in VMS. I don't want to try to separate out what parts of the VMS belonged to Dave and what parts …

Burniece: I think he kind of drove the program though afterwards.

Lary: Huh?

Burniece: When I joined the company in 1981, I think he was basically in charge of VMS.

Lary: Could be. I think it was more of a shared responsibility.

Burniece: Dave was already in Seattle by then.

Lary: Yeah. Dick, unfortunately, was in a car accident, a very bad car accident. Somebody t-boned him on the driver's side.

Burniece: That was like 1982 or so.

Lary: Yeah.

Burniece: Because I was already at DEC.

Lary: And I was already in Colorado. It left him brain damaged - there was damage to the frontal lobe and the parietal lobe. Digital, to their credit, had an engineering hierarchy that paralleled the management hierarchy.

You could be a supervisor, a manager, a senior manager, a vice president, whatever. And as engineers, you could be an engineer, a principal engineer, a senior principal engineer, a consulting engineer, senior consulting engineer and corporate consulting engineer. Dick had gotten up to being a corporate consulting engineer, which was pretty much the highest level [and equivalent to Vice President]. Later when we got bigger, DEC invented senior corporate consulting engineer.

But [Dick had] gotten up to corporate consulting engineer and Digital, to its credit listed those in the annual report. They never took him off the rolls.

Burniece: Right.

Lary: His residence became a nursing home and he was in the annual report every year as a corporate consultant.

Burniece: That's a real tribute. Now, has he now passed away? Or is he still alive?

Lary: I believe he has passed away.

Burniece: But I know they did this.

Lary: I would go visit [him] occasionally.

Burniece: Yeah.

Lary: You know, but I lived in Colorado. It wasn't that easy. There were people who visited him regularly and he had moments of lucidity and moments, where he was hallucinating.

But he'd recognized people when they came in. I mean, it was a real tragedy. It was a major loss to the company.

Burniece: Oh, it really was. Absolutely. He is probably one of the most revered guys in DEC history, when you get right down to it. So talk a little bit more about Dave’s and Dick's working relationship. Did they work together on the original coding of the VMS? Or did the split it? Or how did they do it?

Lary: They kind of split it up. Dave was responsible for …

Burniece: He was the Kernel guy.

Lary: … the Kernel, right? Handling devices, scheduling jobs, things like that, right? Dick was responsible for the OS services layer. That's my understanding.

Burniece: That would make sense.

Lary: Right? And the things that reached into the file system and other things like that, right? So the closer you got to the user experience, the more Dick was involved. And the closer you got to the hardware, the more Dave was involved.

Burniece: Right.

Lary: Right. As I said, I was at a distance from there, because after I finished on VAX, I had still never done any hardware but before being exposed, I had never done a computer architecture [either].

Again, this is a learning by doing. After being on the VAX architecture committee, I kind of got a taste for hardware. I wasn't ready to do hardware. I didn't know anything about hardware.

So I did two things. One is there was a PDP-11 coming out called the 11/60 that was micro-coded. So I decided to help them out. I kind of volunteered and wrote some microcode for them, which is getting right down, controlling the gates of the hardware.

Burniece: Was that your first time actually doing microcode?

Lary: That was the first time I did any microcode.

Burniece: Now, differentiate for the audience between microcode and software.

Lary: So the software is what the machine presents to the world. Microcode is just an internal technique for designing a computer that's going to …

Burniece: That intimately manages the hardware, right?

Lary: … do complex things. Usually, every bit or bit field in the microcode has some direct control of switching some mux or turning on some gate or providing controls to an ALU or something like that.

Burniece: Where software is compiled [correct?].

Lary: Microcode words tend to be very wide and, although later people have actually succeeded in doing this, [it tends] to be very difficult to do any kind of machine translation to write a high-level language that converts into microcode. So it is all hand coding, which of course I was great at, and the other thing was that microcode used to fit in a read-only memory, or at least a very hard to modify memory. You could do updates to it, [however]..

Which meant you only had so much [memory] and this is where my skills at turning N words into N minus 1 came in handy. So I became a microcoder on the PDP-11/60, helped write some of that code for floating point and some other things.

And then as a hack, I wrote a PDP-8 emulator in microcode for the PDP-11/60. A friend of mine, Mark Bramhall, who was doing RSTS, built a shell in RSTS that we could run PDP-8 code native on this PDP-11 and it would run on PDP-8 speeds.

Because the 11/60 was a pretty fast machine and the PDP-8 guys who were left who were still doing PDP-8 development, they used that as their development platform. They had a virtualized emulated PDP-8 that was time shared that ran at PDP-8 speeds, and that they could access from their offices as a time sharing system and run multiple jobs on and stuff like that. And they used it for the next 10 years to develop PDP-8 software.

Burniece: Just as a quick footnote again for the historical record, when I joined the company in 1981 to run the Colorado Engineering Group of which you are a member, Gordon Bell, who was my boss's boss (I worked for Grant Saviers, who worked for Gordon), made it very clear that you were on to be call at any time on any product, where they needed somebody to fix the microcode. Thus, you were considered the microcode fireman of the company.

Lary: Yup.

Burniece: You did get called and you would leave Colorado to get into somebody else's spaghetti code and fix it, then come home.

Lary: Yup. In fact …

Burniece: And you did it all the time.

Lary: … I wound up working on the microcode for one of Tom Stockebrand's terminals, where they were running out of space. When the original VAX was done, the VAX-11/780 had only 4k of ROM for microcode and they had a small writeable control store in case that overflowed.

So they started to run into a space trouble on there and I joined the project temporarily and kind of crunch, crunch, crunch, crunch, wrote negative code, right? I took 100 words of code and turned it into 80 and kept doing that in other places. Then I wound up implementing some of the weirder decimal instructions and other things and it helped us. The more room we had, the more stuff we could pack in. So yeah, I was a code bummer, right?

I don't claim to be a great theoretician or abstract thinker. But I was an abstract “shrinker”. I could bum code like nobody's business.

Burniece: There's no question about that. Your legacy is definitely there among other things. Let's talk a little bit more - we're going to take a break in a little bit - about the VAX team, or let's say the implementation of the VAX, how it happened, who the key players were in actually implementing it. And then we're going to kind of move into the VAX cluster after a break. But tell me about how that whole VAX architecture turn into a product. Because you're one of the key guys that made it happen.

Lary: OK, there are a lot of names that I'm going to have to refresh my memory on. But the guy who pulled it all together was named Peter Conklin, who unfortunately died young of pancreatic cancer. But he was effectively the program manager I think. He kept all the balls in the air.

Burniece: So he was the overall program manager of VAX.

Lary: This was an immensely complicated program. We had new hardware, new software. We had a field to train. We had manufacturing to get going.

So he was where it all kind of came together. The hardware effort was run by Dave Rodgers, coming out of the VAX A committee, as the hardware development manager. He had a lot of help.

Some of the guys I remember on that project were a guy named Jud Leonard, who actually came out of the real-time PDP-12 laboratory stuff, and became one of the firmware developers. There was definitely a very, very competent hardware engineer named Bob [Stewart]. I don't remember how much of a role he played on the 780 but he played a role in one of the follow-ons.

At the same time, we were doing the 780, we realized we needed a family, so Steve Rothman came off the committee and became the lead for what became the 750, the smaller brother of the 780, which was about a year delayed in time. But he started putting a team together around that. The software effort --

Burniece: On the hardware side, Bill Demmer was in there someplace, too.

Lary: Bill Demmer was in management. He was a senior manager. But I'm talking about guys who …

Burniece: did the implementation.

Lary: … lived it, right? I mean, Bill certainly lived it. But you know what I mean. He was a people and budget manager. Not to denigrate people and budget management, it's incredibly important. But in terms of the technical project, getting all the pieces to come together at the right time, et cetera, it was guys like Peter Conklin.

The software had some strong personalities in it, like David Cutler, et cetera. It was led by a guy named Roger Gourd. Roger has had a career since then [Note: he wound up as the VP of Engineering at the Open Software Foundation]. But he was the kind of guy who could keep strong personalities on target and not working at cross-purposes. He was very, very effective at that.

I still have a little note the day he wrote.

There were a bunch of us, names I mentioned before, myself, Stan Rabinowitz, and George Berry. This is a little side thing.

Every computer that Digital had ever produced had a code card, which was a single piece of cardboard that an engineer could stick in the ubiquitous shirt pocket, and you could pull it out and it would have the architecture of the thing, what the instructions were and how to …

Burniece: Something like a 3 by 5 card of the architecture.

Lary: It was a cheat sheet. It was more like a 3 by 8 card.

Burniece: Ok.

Lary: Well, VAX was so complicated that the idea of a code card was ridiculous. George Barry, who I originally worked for at Digital in PDP-8, had gone on to become the head of the Typeset-11 group. Digital had gotten into the computer typesetting business, as a little side thing, setting type for newspapers and classified ads and stuff.

This was the first kind of non-WYSIWYG (what you see is what you get), but real editors that would do real kerning of fonts and all that kind of stuff. [Note: Typeset-11 wasn't like Word, where you see immediately what the final output will look like, because in 1976 we didn't have graphics on workstations. It was more like HTML, a "markup language" that you had to run through the typeset software to produce the final result]

I don't know who had the idea but, between Stan Rabinowitz, George Berry and myself, it was like we could do a code card for the VAX that would be like an accordion and fold down to the same size as the regular code cards. So we just came in over a weekend and by Monday morning we had it done. George provided the technical support for Typeset-11.

Stan and I did the labor of laying everything out and we handed it to Roger Gourd on a Monday morning. He almost kissed us and that become his managing document from then on.

Lary: Because it wasn't in anybody's job (nobody had signed up to do this), it was like, hey, Roger, here is a present for you. You know, boom, he went and had a gazillion copies printed and all these guys now had something they could stick in their shirt pockets that was the cheat sheet on how to write code for the VAX, because VMS was written in assembly language.

Burniece: Now, when you talk about Roger being in charge of software, are you talking specifically about VMS or actually beyond that in some of the other pieces?

Lary: Everything. He was the software management …

Burniece: So all the software.

Lary: … yup, that is my memory.

Burniece: So somewhere down in there we got Hustvedt and Cutler managing it.

Lary: Yeah. They were the technical leads, the project leads for it. But he was the guy who pulled all the software guys together and made sure that the operating system and the compilers and all that other stuff …

Burniece: We had all of that stuff.

Lary: … was all going to get there at same time and all work well together and the interfaces were going to be long-lived and all that. So he had the architecture guys and a lot of implementers working for him. He was kind of the Peter Conklin of the software side.

Burniece: Yeah. He was almost the equivalent of the Fred Brooks on the System/360--

Lary: Yeah. Exactly. Exactly.

Burniece: … who wrote The Mythical Man-Month about how hard that was.

Lary: Right.

Burniece: Did you have similar problems? Or did Roger keep the size of the team down make it efficient.

Lary: We went in 2 and 1/2 years from Gordon calling a meeting to announcing the product. That's pretty damn good.

Burniece: How many guys were there?

Lary: [A lot]. You didn't have a lot of slip-ups in that kind of thing. Very early on, the hardware team built a breadboard, which was I think all the VAX breadboard. It was like a 1/10 speed version with wire wrapped cards and a lot of discrete components and it was slow as hell.

And the software guys developed the software on that until the real hardware was ready. We didn't use, I think, any custom chips on the VAX. But we were using chips that were going to be available when we needed them, not were available when we started the project.

So we were laying boards out, and then waiting for certain chips to arrive that were necessary, that were the fastest SRAMs we could buy at the time and things like that.

Burniece: This was the middle '70s.

Lary: This is the middle to late '70s.

Burniece: Was this now TTL or ECL or something unique?

Lary: It was all TTL. The 11/780 and the 11/750 were all TTL Schottky S series, S series TTL logic. Later, we built something called the 8600, which was ECL. And in fact, Judd Leonard went on from playing a supporting role in the 11/780, is my memory, to being basically the lead firmware guy on the 8600.

Burniece: OK.

Lary: But when the architecture part was through, I kind of went back to PDP-8s and PDP-11s. You know, VAX stood for Virtual Address Extension. Meaning, address more than 16 bits.

I was so impressed by that that I went back to PDP-8 land. I said, you know, memory's getting cheap. Instead of having a 3 bit extension to the PDP-8's 12 bit address that got you up 15, maybe we ought to add another 2 bits and get you up to 17. And we actually made a VAX for the 8.

Burniece: Is that right?

Lary: All it did was extend the memory. And in fact, Tom Hastings kind of went back to the PDP-10 guys and said, we only address 18 bits of memory. We need some kind of address extension for the 10 and got the hardware guys involved. And they figured out a way for a program to address more than 18 bits on the PDP-10.

So we went and did the VAX. Then we took that religion back to our old groups and we came out with a 128K word, so about 192K byte, PDP-8.

Burniece: What was the model number of that?

Lary: Huh?

Burniece: You remember the model number of that?

Lary: No. I just remember the poster for it, which was a picture of a PDP-8 with an elephant standing next to it with it's paw on top of it and something about memory only an elephant could learn.

Burniece: I love it.

Lary: Or whatever. You know, so Gordon's idea behind the VAX infected parts of the rest of the company. It was like, oh, cheap memory is coming. We got to do something about this. It's interesting how a concept spreads.

Burniece: It absolutely is.

Lary: And at same time I got involved in an AD project working for a guy named [Jega Arulpragasam]. And that was when I got to, instead of watching hardware get built, I got to get intimately involved in the building of some hardware, which helped me out a lot when I wound up joining the storage group, and we had to invent a bunch of hardware and bunch of software that all worked well together.

So one other story, and this will probably take us to the break. Again, Stan Rabinowitz, who was one of the Poly mafia, was another mathematician, much more talented than me at mathematics, and a very inventive guy. He and I had this idea before the VAX existed that we thought we could build a program software for computer that would let it play Scrabble.

OK. There were lots of chess programs and checker programs. But Scrabble was a different program about word making and things like that. It did have some strategy aspects.

But mostly it was about figuring out a way to do these kind of searches that weren't really searches for words, but were searches for words with wild cards and figuring out how you might be able to make words from this crappy bunch of letters you have and this crappy bunch of letters on the board. We worked out some algorithms and we wrote a Fortran program, which ran on the PDP-11.

It was way bigger than the PDP-11's memory, so it had to do a lot of paging in from disk, and this was all in our spare time as a hack.

One of the questions you would put on your little sheet was what are you most proud of that you did at Digital? There was a lot of things that I did at Digital that I was proud of but one of the things I'm the most proud of is the Scrabble playing program. I'll tell you right now - because it was good. We published it and we got a lot of flack.

We got flack in two ways. One, we got flack from the managers of the tech writers, because the tech writers were wasting all their time playing Scrabble. They were English majors and this program was a great word maker. It had some strategy stuff to it, too. The second place we got flack was we wanted to release this program to DECUS, again, free to the DEC user community We talked to some people and they said that was a good idea. Then somebody said, maybe you should talk to the lawyers.

I have a Scrabble board here and it has this copyright statement on it, Selchow and Righter copyright. So we went to Digital's chief lawyer and said, this is what we want to do. He said, well, let me see what I can do. I'm going to write a letter to Selchow and Righter. He writes them a letter and somebody at Selchow and Righter must've been forward thinking because they wrote us a blistering letter back - don't you dare. When I say forward thinking is I mean, this was 1976 maybe, '76, '77. Computers were for grownups. Computers were for big companies.

Who cares if somebody wrote a program to play - these guys were looking ahead to the point where maybe there'd be a handheld game that would play Scrabble and they could lose their copyright to a computer version. So they shot that down.

It became an inside the company program only. Then, of course, stuff leaked out to the user community. Digital could never be formally associated with the release of the Scrabble playing program but when they decided to announce the VAX, October of '77, Ken Olsen held a big press conference.

Not just the technical press, but The Boston Globe, The New York Times, and all this stuff, announcing Digital's new computer family to the world. We ported Scrabble to the VAX where it did fit in memory and he showed it at the announcement.

He said, you know, people have programs to do chess and checkers and these other games on the computer. Ours plays Scrabble and he let the press at it. Of course, they're English majors, too, so they loved it.

Burniece: That's fabulous.

Lary: They're sitting there oohing and aahing over the moves the computer's making.

Burniece: I never heard that story. That's sounds like a good place for the break.

Burniece: Ok, we're back and we left it with …

Lary: the announcement of the VAX.

Burniece: … you're now back in PDP-8 land, taking some of those ideas into PDP-8 land.

Lary: Right.

Burniece: Now, I want to bridge you into storage. The rest of your career is mostly in storage, so how did you get involved in storage and how did that eventually evolve into the VAX cluster and all that? So, go for it.

Lary: So as usual, it was right time, right place. This time in a negative sense. The VAX got introduced in October of '77. At that point, they didn't need me to bum the microcode anymore.

I was doing this advanced development project, PDP-8 stuff, and some PDP-11 microcode. In February of '78, a blizzard hit the Northeast. The famous Northeast blizzard of 1978 basically closed Massachusetts down for four days and I was housebound for four days.

Shortly after that, maybe March of 1978, a guy named Barry Rubinson, who I had known, was in the storage group and said how'd you like to get out of New England? I said, boy that sounds good to me. He said, well, we're opening up a storage organization in Colorado.

We're looking for people and I'd like to steal you and get you involved in the storage group. So you've got to leave what you're doing with CPUs and operating systems and all that stuff and get into storage. I said, sounds good to me.

So I went out for a look-see trip in April, I think it was, and in August of '78, I moved out to Colorado. In the time in between, I was working with this advanced development group that Barry was part of under Mike Riggle, exploring different concepts in controlling storage.

At the time, storage was dumb, mainly disk drives. There were lots of disk drives. But generally what a disk drive presented to the outside was an interface almost directly into the heads and the motors of the drive.

You'd issue it a low level command by writing a register, saying seek to this spot or turn on the read gate or turn on the write gate, and it would do that. The controllers weren't much smarter that did that. They would just issue those commands and then act as a conduit for the data to run into the computer's memory to and from the disk drive.

At the time, microprocessors were coming on board. Not so much the 8080 kind of microprocessors, which were wimpy. The microprocessor that we were interested in was a microprocessor called an AMD 2901, which is a bit slice microprocessor, and it was fast.

It could run in the 5 to 10 MIPS range, 5 to 10 Million Operations a Second, and it was kind of semi-microcodable. It was somewhere between microcode and assembly code in terms of its programmability.

So Mike Riggle's group had looked at the idea of using microprocessors in that class to add intelligence to the storage controller itself That intelligence would be scheduling operations to the disk, being able to handle lots of operations ahead from the computer, organizing them for best performance, sending them to the disk drive, getting the results back, handling the errors automatically, providing the idea of a perfectly good disk, relocating bad blocks as necessary, all the things you had to do to make disk technology really work. Because, whenever disk technology threatened to become too reliable, they just cranked the density up until it was on the hairy age of unreliability to get the cost per bit down.

All that was none of my doing. That was Mike Riggle, Barry Rubinson, and those guys. But I came in kind of as an implementer to try to make it all work. The other idea that we had at the time was the idea of using caching, which was a technique that had been developed for CPUs in the early '70s.

Turns out the first CPU that had cache on it was actually a big IBM 360 that shipped in December of '69. So really the 1970s were the era of caching and we started shipped computers with caching, around '73, '74.

We designed the VAX with caching and even had some systems with what was called write-back caching. The idea of using caching to speedup storage access by using DRAM as a cache for storage systems was very attractive, especially caching writes, which was where the real action was.

Operating systems, even then, would cache reads but nobody cached writes, because if your power failed, you'd lose your data. Well, we figured out we could put batteries in there and do other things.

So the goal was to build a disk system that was intelligent, could handle a bunch of drives, do the right thing, and eventually do shadowing - the word RAID didn't exist at the time but shadowing is now called RAID1. Nobody had RAID5 at the time.

Burniece: It's actually called mirroring today.

Lary: It is called mirroring today but we called it shadowing then. Shadowing has a different meaning today. So you know, technically it was intriguing and Colorado was real intriguing, so I said yeah.

So I moved out to Colorado in August, I think it was, of '78. At that point, Digital was in the process of building a huge plant in Colorado Springs, Colorado. They bought a little valley.

They had gotten some concessions, tax concessions and other things from the city of Colorado Springs in return for some concessions about not being too noisy, not being too high, not emitting too many noxious byproducts of disk manufacture. But we were in temporary quarters at the south end of town. I was not in the first wave of …

Burniece: You were down at Fountain.

Lary: … people to come to Colorado Springs. By then, the original plant in Fountain had been superseded by having two or three floors of a building on the south end of Colorado Springs itself, an area called Garden Valley. Barry had been busy recruiting a team of good engineers, some of them out of Digital in the east, some of them out of California.

One of the reasons Digital established its storage group in Colorado Springs was [there was already a strong base of experienced disk drive and storage system talent in Colorado, Minnesota, Oklahoma and California that could be attracted to Colorado Springs]

The other reason was some of the incentives that Colorado was offering and Colorado Springs happened to have one of the first number five ESS phone systems in the nation. This was great for communications with the rest of the country, so we also established a support group in Colorado Springs

Another reason was that Digital at that time, late '70s, had gotten so big, we had exhausted the talent pool in Massachusetts. We had pulled all the people from MIT that were going to come from MIT. We had pulled all the people from New York that were going to come from New York. And it's real hard to get engineers to move from California to Massachusetts, because of climate and other things.

But you could get engineers to relocate from California to Colorado. Because it was Colorado - Rocky Mountain High, low taxes, outdoor life, and no crowded freeways.

Burniece: Let me add a footnote to this. Because I do know some of the back story here. One of the key reasons that they end up located in Colorado was because Fred Hertrich was in Boulder. Fred was an ex-IBM engineer, who Grant Saviers contracted to design the RL01 removable media disk drive, which was going to be the first disk drive manufactured by DEC. Grant and Bob Puffer had made an agreement to manufacture the RL01 and other future disk drives in Colorado, so they wanted the storage engineering team near manufacturing but did not want to be in Boulder, because Storage Tech was in Boulder. So they ended up going to Colorado Springs.

Lary: Boulder's expensive. Colorado Springs was cheap.

Burniece: Colorado Springs was just far enough away from Boulder that people were not likely to commute it, so it would be more difficult for the Boulder companies to steal DEC's people, although DEC could probably get a few Boulder people to move down to Colorado Springs …

Lary: Exactly

Burniece: … which they did.

Lary: And we recruited a lot from …

Burniece: California.

Lary: … Cal Poly. Not so much from Caltech, but a lot from Cal Poly.

Burniece: Yeah. I can specifically say that as the guy who ran that group for a number of years in the '80s, a large number of our very talented engineers came from Cal Poly in San Luis Obispo and the reason was they were very practical lab-trained engineers.

Lary: Yup.

Burniece: They weren't just theoretical. They were very good.

Lary: So, Barry had put this on this team together and I started working with some people doing simulation. In fact, as a side effect I wound up using a language called SIMULA, which was one of the first object-oriented languages. I got kind of an early introduction to object-oriented software, which didn't take. Still, I don't consider myself an object-oriented software guy.

But it was a good learning experience. We moved out to Colorado and we started working on a couple of things. First, we started working on some backplane storage controllers that fit into Digital's storage buses at the time.

Digital had a bunch of buses out there but there were two peripheral buses that were in common use and were kind of universal across DEC systems. One was called the Unibus and it came out with the original PDP-11. It was an asynchronous parallel bus. The other was called the Q-bus and that came out with low-cost PDP-11s.

VAX's supported the Unibus and I think there were adapters that could support PDP-10s through the Unibus but I'm not so sure. PDP-10 tended to be more supported on something called the MASSbus.

So we started out building a controller called the UDA50, which [stood for] Unibus Disk Adapter. It was numbered the 50, because it was going to be the middle of some range - maybe. As opposed to an ad hoc disk bus, where every disk wound up with its own unique bus that controlled that drive, we architected an interface to [a new family of] disk drives [called the RA family].

It was called SDI, the Standard Disk Interface, and it started out being a parallel cable, with 16 bits of transfer and a whole bunch of control wires

We later had to fix that, because one of the early prototypes - you know, disks were big then - had 14-inch platters. The whole drive was a full rack wide. Usually, you could only fit about four in the height of a [full] rack and they weighed some ungodly amount, you know, 150, 200 pounds or something like that. So if you wanted any number of them, you had put them in multiple cabinets that were separate from the cabinets that you put your CPU in. Then you had to run cables out to those cabinets and they pulled a lot of power, as did the CPUs. The first time we actually hooked up our parallel SDI bus to a disk drive that was in a separate cabinet and plugged it into a different outlet on the wall, the cable caught fire.

Burniece: I hadn't heard that story.

Lary: Because …

Burniece: That's the parallel SDI

Lary: … the ground difference between the two cabinets, the ground wires and the cable, wasn't good enough. Then it was like, oh, my god, we've got to beef up the grounds and we got into differential signaling. All of a sudden, this cable's becoming another MASSbus cable, which was about this thick.

So one of the hardware engineers got the idea, why don't we go serial? This knocked the many, many wires in a parallel cable down to just four serial wires.

Since we needed a lot of parallel control, we used generic control signals, like start reading, start writing. The drives were sending back things like I saw a sector pulse, an index pulse, things like that. Two wires ran in each direction. One of them held data and the other one held a rotating 6-bit status field, so we'd run these wires at the incredible speed of 25 megabits per second.

Burniece: Which was very fast in those day.

Lary: Which was faster than the drives could transfer, way faster than the drives could transfer at the time in 1979.

Burniece: In fact, the CDC SMD interface, which was [an ANSI] standard of the time, was at 10 megabits a second.

Lary: Right

Burniece: And that was a parallel bus.

Lary: We thought we had plenty of margin but, by the late '80s, we had exhausted that margin and that was as fast as we could get the stuff to go reliably, because we had to go 25 meters, since these cabinets could be arbitrarily far apart. We didn't want to really set any hard limits on it.

So the bus got redesigned to this [serial] interface. There was an LSI chip group in Shrewsbury under, I think, Norm Field, with guys like Bruce Buch and Mike Leis, plus a whole bunch of chip designers and analog guys, who developed the Serial SDI.

I wish I could remembered the name of our ECC expert, because he was something. He wasn't great at communicating but, boy, did he know Galois fields

Burniece: Well, we had one who was Chinese.

Lary: Yes. That's the one I'm talking about.

Burniece: Lih Weng.

Lary: Lih Weng. There you go, Lih Weng.

Burniece: He was a world-class expert, as was Mike Riggle.

Lary: In ECC.

Burniece: Yes, but there were a couple other guys that were pretty good at ECC …

Lary: Yup.

Burniece: … including Mike Leis

Lary: So they designed a couple of chips to support this serial version of the bus. One [the SERDES] would take up parallel signals and turn them into serial signals and do syncing and all that stuff on the serial bus. The other was a dedicated chip to do the first part of the ECC calculations.

ECC calculations are basically solving simultaneous equations in Galois fields. But first, you have to compute something called residues, which is very compute intensive, so this chip would compute the residues. Then the software could go read the residues, and go solve the--

Burniece: And this was …

Lary: … simultaneous equations.

Burniece: … actually a real-time Reed-Solomon code …

Lary: Yes.

Burniece: … which was very much ahead of everybody.

Lary: It was real time, very sophisticated at the time. Most disks had like a 32-bit check code on an entire sector for 1,000 bits of information in a 512-byte sector, and it would detect one or two bits in error.

This was 170-bit [Reed Solomon] code, very sophisticated for its time. It consisted of seventeen 10-bit symbols. and we could correct any eight symbols being wrong somewhere in the sector.

The idea was that this was going to give us a competitive advantage, because we could crank our density even closer to the hairy edge than other people and get more information out. That would pay for the roughly 4.5% overhead that the ECC put on top of each sector, so we could crank our density by 20%.

Burniece: Just again to get a little historical perspective, you moved out there in the summer of '78.

Lary: Summer of '78.

Burniece: The UDA50 first shipped on my shift in early '82, but when did the decisions get made to go from parallel to serial and use a Reed-Solomon error correcting code? Did that happen right after you got there, or closer to the ship date?

Lary: The Reed-Solomon error correcting code was an independent decision. That came out of the disk electronics guys saying we really needed a [more] sophisticated code and that also worked on the parallel data that came out of [the SERDES]

So when we made the move from a parallel bus to a serial bus, that chip basically hardly changed at all. But [that] was kind of part of an oops - parallel ain't the way to go. We [have] got to go serial, so let's get this chip out in a hurry, so that was one of gating items there. [Note: data from the disk head is always serial, so basically we moved the SERDES from the disk side of the parallel bus to the controller side of the serial bus]

Burniece: One of the reasons I'm asking is because that's a key piece of what became known as the Digital Storage Architecture, which you'll elaborate on later.

Lary: Which was that [serial] bus.

Burniece: And it was pervasive. It was used in every DEC disk drive for the next 10 years plus, as well as tape drives.

Lary: Right. We had controllers that could mix and match …

Burniece: That system.

Lary: … disk drives across generations, across form factors, because they all corresponded to this standard bus. Before this time, there were some standard buses but they were things like ST-506 and a couple of others.

Burniece: The SMD had just become a standard.

Lary: They were pretty low end buses. They didn't really work for high performance disk drives very well.

Burniece: SCSI was not yet a standard.

Lary: Exactly.

Burniece: They were working on it. It wasn't a standard.

Lary: Intelligence had not gotten …

Burniece: IDE had not happened yet.

Lary: ... cheap enough to push into the disk drive. We did have, in the disk drives that were on this SDI, a processor and memory but that processor was much more like an 8080 and would handle things like sending commands over the data lines to the drive and you'd use the status lines to say, hey, this is a command. The command would be like a [direct address] seek and it would consist of multiple bytes. The driver would put it together and go, ok, I got a seek and the microprocessor and driver would control that.

When the data started flowing, the microprocessor just kind of ducked down and let the data fly right over its head right onto the wires, because it was nowhere near fast enough to process that data. It was nowhere near fast enough even to control the timing of what's going on in the sector with headers and data and turning things on and off at the right time. That was done over the wires by the controller.

Burniece: Now, just for historical perspective, because I know Barry [Rubinson] and Mike [Riggle], plus of course you, when did [you three] begin to come to the conclusion that you needed a comprehensive extendable architecture that could be repeated for multiple generations? Did that happen kind of simultaneously or had that been pre-planned?

Lary: The idea of a standard disk bus was there before I arrived. So I can't really tell you who came up with it. It was just a given that we were going to have standard disk bus. Digital was very big as a company into backwards compatibility.

Meaning, you don't just build something. You build an architecture and then you build members that comply to the architecture. Because you do that, the stuff that you build today will interoperate with the stuff that you build five years down the road. We were an architecture-driven company.

Burniece: That's true.

Lary: That's why we built a PDP-11 architecture, and then we built like six or seven implementations over a span of 10 to 1 or more of compute power. In fact, Gordon Bell's famous dictum on April 1, 1975, when he started the VAX architecture group, along with the memory is getting cheaper speech, he said, I want an architecture that will cover a 1,000 to 1 range (1,000 to 1 in performance). Sometime around the late '80s, I went and looked at what you could get in a VAXcluster, versus what our cheapest VAX was selling for and I sent out a memo that said, we made it.

Burniece: That's fascinating.

Lary: The biggest VAXs you could buy, I think at the time, was maybe VAX 8600s or VAX 9000s or something like that and could put like 16 of them out there as a VAXcluster. Then you [could compare] the MicroVAX performance and it was 1,000 to 1. Gordon got his wish.

Burniece: That is a fascinating thing. I'd forgotten that piece and Gordon was long gone by then.

Lary: Gordon was not really involved in the storage stuff. In fact, somewhere in the middle of it, around the mid '80s, he left the company. But he was involved at the beginning and influenced some decisions, [although] possibly not for the better I can mention now.

Burniece: So go back again. You're now in the late '70s. The company's working on the RL01, which was a completely separate product, which is not part of that architecture.

Lary: That was [started before we had the SDI-based Digital Storage Architecture (DSA)]

Burniece: You were working on the first product set for the DSA architecture, which is the UDA50 and a new Winchester drive called RA80, [when I joined DEC in August,1981].

Lary: We were turning [the MASSbus-based RM80] into--

Burniece: the RA80.

Lary: -- We started with an existing drive and just modified the front end. Because it minimized the amount of new stuff that all had to work on -.

Burniece: You actually put the MASSbus RM80 out first, which was [DEC's first Winchester disk drive]

Lary: Exactly.

Burniece: And then we put out the RA80 on my shift, which was the SDI version.

Lary: We used the same HDA (Head Disk Assembly) servo and all that. Then we put a different control card in front of it and it became the RA80. If memory serves, it was like a 120-megabyte monster drive.

Burniece: 120 megabytes right. I started at DEC in August '81 and we shipped the RA80 on the UDA50 in the spring of '82. So I know that happened in that time frame. When I got there, the HSC was well under way.

Lary: Yes.

Burniece: When did the concept of VAXclustering with a storage controller that could control storage to multiple VAXs, called the HSC (Hierarchical Storage Controller), start?

Lary: I can't put nearly as precise a date on that, because I was a little peripheral to that. I was out in Colorado but I do know that the genesis of VAXclusters came, as usual, from Gordon Bell.

Burniece: Ok, so it was his idea.

Lary: And in this case, it wasn't a technology trend that was worrying Gordon. It was the rise of Tandem [Computer].

Burniece: OK, makes sense.

Lary: Tandem was building highly reliable systems.

Burniece: Fully redundant kind of systems is what they were doing, right?

Lary: They got their reliability through redundancy of separate computers. Tandem had, I don't want to put words in Tandem's mouth, but my recollection was that Tandem had a dual, because everything was dual with Tandem (that's what Tandem means) - bus. They called it, I think, the X and Y bus.

Burniece: Might be right, yeah.[2]

Lary: They connected their multiple processors together and basically, Gordon said, we need something to compete with Tandem's X and Y bus but, because I'm Gordon and because we're Digital, I don't want to just copy what they did. I want a bus that not only will give us a reliable redundant connection between computers that we can then build a reliable system out of the two computers, I want a reliable bus that will connect N computers together, for N to be arbitrarily big. Not quite, he didn't say that but I want N to be big and I want to be able to scale the compute power with the number of computers you put on this thing, as well as allow, enable full tolerance between all the computers in this collection.

Burniece: So that's what became known as a cluster?

Lary: That concept became the VAXcluster. We, at the time in Colorado, were building these single board and two board backplane controllers, the UDA and the QDA, and they used the 2901. The 2901 was a 4-bit slice microprocessor.

We put four of them together, so we'd have a 16 bit data path, put some EPROM on it and wrote code to drive that bus. Also on the host side, we figured we need two things - one, we needed an interface to pass commands over to the controller and second, we needed a command language for the controller.

So, again, because we were Digital and architecture comes first, we build an architecture to do both. We built what's called port architecture to figure out how you pass information using shared memory from a process running on a computer to a process running on a peripheral.

I don't remember if that ever had a formal name. We called it the UDA port but it was basically a ring buffer in memory. One guy would be the producer and the other guy would be the consumer and it was very efficient.

There was a little doorbell you rang, after you put something into the ring, and the other guy had a little doorbell that he rung when he freed something up. If the things got hot and heavy, you didn't have to ring the doorbell right away.

You could put a bunch of guys in there, so the harder you hit it, the more efficiently it ran. It was a nice little bus and later, some version of it kind of became an industry standard way to do communication. But we actually were there first and got a patent on it.

Then the other thing was you needed a language to specify what you wanted the controller to do and also to handle all the interesting bookkeeping about what drives do you own - what's their status? Could you tell this drive to do this, et cetera? That was called MSCP, the Mass Storage Control Protocol.

We didn't want to call it a disk control protocol, because we wanted to handle tapes, as well, with the same architecture and the same bus.

Burniece: And ultimately, it also took optical [devices]

Lary: SDI also became the STI [Standard Tape Interface]. We just redefined what some of the status bits meant, to be tape status bits, and used the same physical bus, the same encoding schemes and all the rest of it.

Burniece: Again, was MSCP developed in your time in Colorado?

[2] [Editor’s note] it was called the Dynabus.

Lary: That was in my time frame.

Burniece: So this happened in in the late '70s.

Lary: I was a major contributor to MSCP but I didn't own the spec. The key spec owner for MSCP was a guy named Ed Gardner, who was very good at this kind of architectural thinking and architectural writing.

He was a prolific writer. The MSCP spec was a thick, thick, thick document just like the SCSI spec today. which is an even thicker document. In fact, he wound up representing us on the SCSI committees for a while.

Burniece: Yes, he did.

Lary: … because he had that kind of experience.

The UDA port was a cooperative effort and again, I was involved. Ed Gardner was also involved and I think our names are on the patents along, with Barry Rubinson and some other folks.

But MSCP we never patented. it was a trade secret, I believe. We had been developing that stuff for the UDA and we knew that we wanted to build a big brother to the UDA.

We knew that it wasn't going to be a backplane controller; it was going to be a controller that connected with some kind of bus [a lot of disks and tape to a lot of VAXs] that were going to sit at the end of the bus. It was going to be big enough that it was going to have to occupy a significant hunk of a cabinet all by itself.

That controller was going to incorporate some of our real cool ideas about what to put in a storage controller and we called that the HSC, which stands for Hierarchical Storage Controller. The H part was because we were going to implement caching in that controller, so it needed to be capable of having a lot of memory and was going to be an expensive box, for it to be able to handle a lot of disks.

Burniece: So when did that idea flourish into basically an architectural spec, but which you could start implementing it? Did you have that by, say, 1979 or '80?

Lary: Yes.

Burniece: I know it was well under way by '81.

Lary: We actually were working on that, when I started in '78.

Burniece: All right. So it was already underway.

Lary: It really was preliminary then, so I got to contribute a lot to it. There were lots of good hardware engineers in the group, as well as lots of good software engineers, but I was one of the few people in the group that had experience in both. I could do microcode, the kind of code that would run in those 2901 chips, so I knew what hardware could do, and I could write software, as well.

So I became in charge of what was called the [HSC data channels]. I had done a lot of coding for the UDA and the QDA, which was writing code for the 2901s, but the HSC was going to be big. There was really no CPU that could handle the kind of stuff that we wanted to do and could simultaneously keep 16 or 24 drives running at the low level of care that the drives needed.

It needed to turn on read gates and write gates of the drives, et cetera. So we built HSC as an asymmetric multi-processer, ok, and what happened was the HSC consisted of a central processing unit, which was a classic computer

Burniece: a little PDP-11.

Lary: Well, this is where Gordon got to stick his oar in and we didn't like it. We wanted to use a Motorola 68000, because it was a 32-bit machine and it allowed us a big address space.

We had no idea how complicated our code was going to become or if we could even write it. We could even have the luxury of doing some programming in a high level language, because we could waste a little memory at 32 bits, blah, blah, blah.

Gordon came by and said, we got to eat our own dog food - you have got to use a DEC processor - and the only DEC processor that could fit in the form factor was the LSI 11 class of processors.

Burniece: The T-11, right? Or was it the F-11?

Lary: I think it was the T-11.

Burniece: Yeah T-11.

Lary: So we used that. And it didn't hurt us at first. But five years down the road, we kind of regretted it. Because the code was getting more complex and we had to deal with the restrictions of the 16-bit architecture.

Burniece: So you basically had this little tiny CPU -- the T I think meant “tiny”.

Lary: We fought Gordon on that one and we lost but we should have won.

Burniece: You had the T-11 as a central processor running the machine and then you had 2901s feeding all the disks

Lary: The T-11 was slow. It was under a MIPS, maybe a little less than 1 MIP. Then we had these 6 and 2/3, 150-nanosecond cycle time [2901] processors on [data channel] cards. In fact, they were the guts of the UDA, because we already had the code written for it to handle that bus.

So we [designed it to] just plug-in [data channel] cards and we had a bus that the PDP-11 could send commands to these guys and receive status results back from these guys.

We also built a host controller, which we weren't sure what the host interface was going to be a time, but we knew we needed one and we figured whatever it was, we can handle it with this 2901 bit slice thing.

So we had a k.host (k stood for controller) and a k.disk and p.I/O, which was a programmable I/O element, the way we conceived it, that wouldn't do a lot of work. It was a traffic cop and it did some work but it was fundamentally a traffic cop between all these other guys.

One of the key things was we had to build what we called the hardware-software interface. Even though these k.disks and k.hosts were programmable themselves, the p.I/O looked at them as pieces of hardware that had intelligence in them.

We had to build that interface, so that the communication overhead was as low as possible, and that was the part that I tore off, doing the hardware-software interface on how the central processor gave work to the other guys.

Burniece: Now, Barry (Rubinson) always described that as a “data flow machine”. Can you talk about why he called it a data flow machine and what that meant?

Lary: Yeah. I can in a way. One of the things we did was (everything here is comparative, when I mention some of the feeds and speeds, any modern person's going to laugh at them) we would have data coming in off the disks at up to 3 megabytes per second (25 megabits per second). We would also have control information that we would need to send to these processors and the control information was in the part of these kind of structures that these processors would thread their way through to figure out what they were doing next and be able to introduce.

What we didn't want to do was say “do this, then do this, then do this”, because we're telling them what order to do things in and they have a better idea than we do about what order.

So what we did is give them a thing saying this is all the stuff you have to do, especially once you get on track, there might be multiple things to pick up on this track, so you figure out the best order to do them in. The key with creating these data structures was to give these guys as much autonomy as possible.

Burniece: So between the data streams, that were kind of asynchronous, you’ve got to manage it.

Lary: One of the things we did was, because memory bandwidth was hard to come by, we built a separate bus for data. So there was the p.I/O and its memory bus, which were just private. Then there was another memory bus that went out that was used to communicate commands to these k. things. Then there was a data bus, which was the highest bandwidth of all. It was a whole 13 megabytes per second that you could read and write memory at, so you could have four disks transferring at once.

It was completely separate from the other buses, so the way in some sense that it was a data flow machine is you set up these data structures, and these k. processors would operate off of them. If you set up your data structures right, you could have this guy go and read information from a disk and put something in a queue.

The queue that it was told to put something on was actually a queue that the host k. processor would pick up and do the transfer to the host. When it was done, the data structures told it to put something on a queue, which would wake up the central processor, and it would go, oh, well, that I/O's finished.

You don't you just send the guy [the p.I/O software] an end notification that it worked, so that's the sense in which it was a data flow machine. There was no central place that all the control went into and then went out of.

The (HSC) controller was distributed - all the disk processors and the host processors worked independently. They could communicate directly to each other and they could communicate to the central intelligence.

Given the amount of compute power that we had at the time available to us for the amount of dollars we were willing to spend, that was pretty much the only way we could do it and we got it to work.

Burniece: So from an architectural perspective, this had to be one of the first implementations of the separation of data and control planes, with asynchronous accumulation of data streams that could be managed independently going in and out.

Lary: We were probably at that time racing the network guys.

Burniece: Yes.

Lary: My network history isn't as good as my storage history but 1978 or so is right around the time of first generation Ethernet. You know before then, when you're communicating a 56 kilobit, or 1.4 megabits on a T1 line or whatever it is, you could have the data plane and the control plane be the same. Memory was fast compared to what you could put on a network.

Burniece: But this was a separation of this.

Lary: Probably 100 megabit Ethernet came out and maybe even 10 megabit if they had switches. But they didn't at the beginning.

Burniece: Well, 10 megabit Ethernet was not there yet.

Lary: Right? You started to need a separate control plane from your data plane. But we needed it for disks, so we implemented it then. That was one (but) I don't know if we can claim a first there.

Burniece: Well, it's got to be one of the early ones.

Lary: We were early.

Burniece: Now, let's move up to the host side. The host bus was called the CI, the Computer Interconnect. Talk about that and how it worked.

Lary: Yep. This was the low-level result of Gordon's imperative, which was we need something that will allow us to reliably communicate between computers that's high bandwidth and scalable, so we explored several things. Again, Ethernet was coming out, you know, in that frame.

So we understood about the idea of just transmit and detect collisions. It was called CSMA/CD, I think, is what Ethernet used. When we looked at that, we decided that it was not predictable enough. The latency was not good enough for us.

You could get into this kind of collision chain where you keep pounded heads for a while until one guy moves out of the window of collision, et cetera. There was too much wasted space. It was great for Ethernet, where there were hundreds of guys competing on a single bus.

Burniece: Well, of course you had to have guaranteed connections because it was storage rather than best effort, which Ethernet did.

Lary: Yeah. Well, I mean, you could put things back together at the transport layer but, yeah, the fewer errors you got, the few reconnects you got, so we wanted something that was a little more deterministic.

We were willing to accept some restrictions, so we decided to restrict the bus to 32, I believe, nodes I wish I could tell you which guys came up with the hardware ideas. I know there was a guy named Steve Pomfret from the large computer group, who was instrumental in doing a lot of the hardware for this.

Burniece: Again, we can backfill that later.

Lary: Somebody had to do the analog signal design, et cetera, and those names are lost in the foggy history in me but it was a very innovative thing. We wanted the advantages of point-to-point signaling, where a connection was kind of terminated on both ends.

Yet, we also wanted a lot of [connections] there but we didn't want to implement a switch, because switching was expensive. Nobody understood it that well back in the late '70s, so somebody and I'd love to know who, came up with the idea of a big doughnut[that we called a Star Coupler, but it was basically a big iron doughnut]

All of the signals were AC signals but they were all AC modulated signals and everybody's point-to-point connection would terminate at the doughnut. It was a big transformer.

Burniece: And it was called the star.

Lary: It was called a “star coupler”. You'd send energy down there and your energy would go into the star coupler and radiate out.

Burniece: To everybody, right?

Lary: The first star coupler was a 16-node star coupler. It was radiate out 15 different ways.

Burniece: So everybody got the same information.

Lary: Everybody got the same information. You even got a copy of your own information back.

Burniece: So that was unique.

Lary: I had never seen that anywhere. I wasn't an analog guy but that impressed the hell out of me,

Burniece: How did you manage getting the right message to the right person?

Lary: Ok, so messages had an address field on them - that part was easy. The question is how you prevented guys from talking over each other.

The way it worked was that we built an arbitration scheme and I think we even got a patent on this. When the bus was idle, it defaulted to Ethernet-like CSMA/CD.

Namely, you would send information out and then you would listen to what came back. If you saw your own information coming back, then you were the only guy transmitting. But if you saw garble coming back, then it meant that some other guy had stepped on you and that the message didn't get through to anybody.

The difference between CSMA/CD and what we did is that CSMA/CD, people would just kind of time out and try again. What we did was we said, ok, when that happens, we're going to enter a more orderly regime.

Burniece: [INAUDIBLE].

Lary: So when you got a collision, you took your node number, which was a small - this is a difference between us and Ethernet. Your node number was a small number between 0 and 15 for the original CI implementation and you would count that many time intervals. A time interval represented a round trip on the bus. so it was pretty quick. The bus was, I think, was 70 meters from point to point, 35 meters to the star coupler.

Burniece: And it was 70 megabits a second, right?

Lary: So you would you count. If you're node number three, you count one, two, three and if you haven't seen anybody transmit, then you transmit. You were clear, so you were guaranteed to get through the second time. Ok, you could get a collision and once somebody transmitted, everybody started counting.

If a transmission got through and somebody else had something to say and there were no number five - one, two, three, four, five, then they'd transmit if they haven't seen anything. So the busier the bus got, the more you got out of collision mode and into orderly transparent mode.

Now, the problem with that was that number one always has priority. If number one wants to say something, he'll always be able to lock-out number two, because number two will count one, two and in the middle of one, he'll see something on the bus. Then he'll go, oh, sh*t - right.

So the way we fixed that was we said, instead of 16 numbers in this count, we'll have 32 numbers in the count. The idea is every time you use the small number in the count, every time you counted the three, the next time you had to count to 19. You had to add 16 to your number.

Burniece: Oh, ok.

Lary: So guys would bounce back and forth between being in the high priority regime and the low priority regime. Everybody got a turn.

Burniece: It's like a fairness algorithm.

Lary: It's like fairness algorithm - everybody [node] got a turn and once that bus was defined, it became very obvious that we were going to use that bus to communicate with hosts.

Not only were we going to use it to communicate with hosts, but we were going to use it to communicate with multiple hosts over the same set of wires. So without knowing it, because the terminology didn't exist at the time, we were building the first network storage controller, only the network was a Digital proprietary network with some very nice properties in it.

It ran at what, at the time, was an astounding speed. It ran at 70 megabits per second and because it had to be as reliable as the [Tandem Dynabus], there were two 70 megabit per second buses in parallel but any one node couldn't be sent to on both at once.   Three could be sent into seven on one of the buses and six could be sent into four on the other bus, however.

So the total throughput, because you arbitrated for them independently, was 140 megabits per second, which was 14 times Ethernet. We felt real good about that and I think it was higher than the [Tandem Dynabus] too.

Burniece: Oh, it was the fastest thing in town at that point.

Lary: But that was the lowest level of VAXclusters. What made VAXclusters unique was not that bus, although the bus was instrumental in making it fast.

I mean. later, VAXclusters even worked on Ethernet. When 100 megabit Ethernet came out, it didn't have the reliability or the predictable latency [of the CI] but we got VAXclusters to work on a pair of Ethernets. What really made VAXclusters unique was something called the VMS lock manager.

Burniece: Right - talk about that.

Lary: That was Dick Hustvedt, ok? I believe that …

Burniece: That was Dick's contribution?

Lary: … yes, was all Dick Hustvedt.

Burniece: Now talk about the concept of file sharing and locking, what that's all about?

Lary: The idea was that, just like in a multi-processor, you lock to get exclusive access to something and then when you're done, you release the lock. You'd have these locks that were “mutex” locks, in that only one person [one active thread] could get through at a time. You could have locks that any number of readers could get through but the minute a writer got in, everybody had to back off and you can have different classes of locks.

The VMS lock manager implemented that as a protocol, if you will, as a presentation layer protocol, on top of the physical protocol of the CI bus and the logical transport layer protocol that I don't remember. I was [not] actually one of the people involved in it. It had some digital communication architecture - SCA, System Communication Architecture.

Burniece: Yes.

Lary: It was a way of establishing a reliable connection between two nodes where things were delivered in order and the usual goodies of a transport layer. But it used some of the good qualities of the bus underneath it to implement it at a very low cost compared to, for instance, TCP/IP.

Because the whole idea was low overhead, high bandwidth, low latency, the lock manager sat on top of SCA. It was quite complicated because what you didn't want to have is one processor holding the lock database.

Because A: that processor becomes a bottleneck and B: if that processor fails, what do you do? You're dead.

So it was a distributed lock manager, where there was a tree (address space for locks) and you can find out who the master was for a particular lock was by going down the tree. If nobody owned it, and you want to create a lock, well, you owned it, and everybody knew who owned every lock. If that guy failed, then everybody else would get together and figure out who was going to take over the mastership of those locks. It was a marvelous design but I can't speak to it in detail.

I didn't do it. I could just sit and admire it from a distance and, luckily, the controllers, the storage controllers, did not have to participate in that - we had our own system.

First, we made storage available to everybody. Every VAX on that cluster could do anything they wanted to any storage [device], because there wasn't a chance of two guys scribbling on each other's data.

The VMS lock manager kept them from doing that. In order to use a file, you had to take out a lock on the file and then you owned that file, so you could access it.

If you were an Oracle database and did things in parallel, the two of you could access the file in common But you'd use the lock manager to lock particular blocks of the file. It was a lovely design for what you might call a shared everything, lock-based multi-processor.

Burniece: So now just to put that in perspective, you have now created the overview of an architecture that became known as the VAXcluster architecture ….

Lary: It was VAXclusters.

Burniece: … in the VMS implementation. It later was implemented in other things besides VMS. But what you've now got is you've got this central bus, the CI bus, that's feathered out to up to 15 VAXs and HSCs?

Lary: 15 VAXs and one HSC or 14 and a pair of HSCs, etc. Actually, the HSCs would take up two nodes, because we were a fault-tolerant component of a VAXcluster.

We had two controllers. The drives had two SDI buses coming out of them. One went to one HSC. One went to …

Burniece: Everything was redundant?

Lary: … the other HSC. The HSCs each sat on both CIs.

Once I had to go up to the European Commission, because somebody had filed an antitrust case on us in the European Commission, and they wanted to know what was so special about these buses. As you know, these are bureaucrats and we had a plant in Ayr, Scotland.

I was giving a dull talk, since it was technical. These guys were like bored silly but I was telling them about what happened. In the middle of the talk, I reached behind a pillar, where I had a pair of bolt cutters, like 4-foot high bolt cutters, professional bolt cutters. I walked up to the system, grabbed the bolt cutters and snipped the cable.   I snipped another cable here and everything kept running. So I said, that's what we do.

Burniece: I love it. I can see it now. So again, getting back to perspective …

Lary: I wasn't even in any danger of getting electrocuted with those bolt cutters, because both the CI and the SDI were AC signaling. There's no current flowing in those wires. I just snapped them, right?

Burniece: So again, getting it back in perspective. You've got multiple VAXs attached across a CI bus and multiple HSCs, each of which has got lots of disks and tapes on them …

Lary: Yeah.

Burniece: … and each VAX is running its own copy of VMS.

Lary: Yes.

Burniece: … and then on top of that is this layer, this distributed lock down.

Lary: This distributed lock manager …

Burniece: …. that manages …

Lary: … and then other services that would use the distributed lock manager to provide cluster services, the clustered file systems, the clustered print service, the clustered batch? service.

Burniece: … so they even could have multiple VAXs running on the same file and this distributed lock manager--

Lary: Yes.

Burniece: … is keeping them from stepping on each other's feet.

Lary: At that point, it was up to the applications that both opened that file to use the distributed lock manager themselves. But, for instance, Oracle did that on VMS and our own database, RdB, did that under VMS, with multiple guys accessing the same file.

Burniece: … and that's completely unique.

Lary: At the time …

Burniece: No one had done anything like that …

Lary: I don't there was …

Burniece: … at that point in time.

Lary: …. anything like that. Not only did we have the full tolerance but you could make these machines cooperate to run a bigger application than could run on any given one of them.

Burniece: … and in fact, you could make it look like all the VAXs were all part of a multi-processing system running …

Lary: Yeah.

Burniece: … together on one app if you wanted to.

Lary: Exactly. Although we eventually got it working under Ethernet, Hustvedt and his team - I think he was still around then - had to stand on their heads to do it, because Ethernet didn't give us the guarantees that the CI bus gave us.

Burniece: I think he was gone [by then].

Lary: One of the problems when you have a big system like that is to figure out when somebody's gone, because you got to make decisions based on a quorum. The CI bus gave you a very reliable indication, when somebody had disappeared. Ethernet, not so much so.

Burniece: Talk about how that worked.

Lary: You had this membership problem and it wound up being a guy at SRC, Digital's System Research Center [in Palo Alto], who since has won the ACM Turing Award, named Leslie Lamport - just a first rate computer mind - to come up with the equivalent of the membership algorithm, which was at the heart of the distributed lock manager, that worked for an arbitrarily unreliable bus, like Ethernet.

Burniece: Are you saying Leslie did this as part of a DEC team? Or he did this later?

Lary: No, he did it as a researcher in parallel processing and distributed systems. He did it as a theoretician. It was not aimed at VAXclusters. It was aimed at the world in general. Since then, there has been a bunch of products out there in the world that have used his algorithm. It's called Paxos.

Burniece: And was the VAXcluster the first to do it or the VMS lock manager?

Lary: No, no. The VAXcluster didn't do it.

Burniece: Oh, ok.

Lary: The VAXcluster didn't do it, because we cheated. We relied on some of the features of the hardware …

Burniece: Got it. All right.

Lary: … and we didn't have to implement it in this unreliable general world environment. So we took a shortcut but it was a very effective short cut.

Burniece: All right. Now, we got about 15 minutes left and we're going to probably have to get back together at some other later time to talk about [what you did] beyond this. But [in the remaining time] I want to flush out the story of the HSC and its capability, because there's a lot of things we haven't talked about.

Lary: By the way, the Computer Museum has an HSC, because one of the last things I did before I left Compaq, when it acquired Digital, was ship an HSC to the Computer Museum.

Burniece: All right, good. Talk about some of the features like volume shadowing and the things that HSC did from a functionality standpoint and how that all evolved.

Lary: So even though it was called the HSC, the Hierarchical Storage Controller, the first version didn't support a storage hierarchy, mostly because we just could not put in enough memory. The memory that we had to use to get that 13 megabytes a second was hot, hot memory.

We couldn't afford to put enough of it in there to make a difference. Eventually, we figured out a way to take some DRAM and really stick it densely on a board and put it on those buses, the data bus and the control bus, plus put a little processor on it to move data in and out of that memory, and then we did implement cache.

So we did finally satisfy the H in HSC. But one of the things, obviously, when you do a reliable full tolerance system, is you have to have reliable full tolerant storage. So the first thing that we did in the controller was to implement volume shadowing in the HSC.

Now the word is mirroring but at the time we called it volume shadowing and you could designate two volumes. They'd become part of a shadow set and the HSC would maintain those two volumes in sync. Ok?.

Burniece: And that's actually what later became named as RAID0 ….

Lary: That was …

Burniece: … RAID1.

Lary: … RAID1.

Burniece: yes, RAID1.

Lary: RAID1.

Burniece: It became RAID1.

Lary: That was just giving it a name.

Burniece: Berkeley. Yeah, the Berkeley people did that.

Lary: Randy Katz and the Berkeley guys came up with a RAID taxonomy and then as part of that RAID taxonomy, they kind of introduced parity RAID (RAID5) to the world. So it wasn't just a simple act of definition; it was an act of invention as well. But in their taxonomy, the shadowing or mirroring became RAID1.

However, as the VAXcluster started to move to things like Ethernet and other interconnects, where we didn't have our own storage controllers (the HSC never had an Ethernet storage controller) we needed to implement shadowing [differently].

So, two very bright guys in the VAX VMS group, a guy named Dave Thiel and a guy named Scott Davis figured out a way to use [volume shadowing] on top of the distributed lock manager and everything. You don't want to, for instance, invoke the lock manager every time you do a write or you'll just be showing way too many calls. Disk drives, even back then, could do, oh, I don't know, a whole 30 to 50 I/Os per second, right?

Burniece: Yup.

Lary: Maybe more if you had a big queue, maybe 60 or 70, and that times a lot of disk drives would be a lot of calls to the lock manager. You don't want that.

So they figured out a way to do volume shadowing, do the asynchronous recovery, do all that stuff, without having the controller involved, distributed volume shadowing. However, when there was an HSC in the mix, when you were in a VAXcluster on CI, they were more than happy to delegate that to us, because it offloaded a lot of work from the CPU, a lot of data passing over the CI bus and passing between CPUs and stuff like that.

We just handled it. The other thing that we also extended to the HSC to do was to handle tapes. In fact, now that we handled both disks and tapes, you could tell the HSC to back up a disk onto a tape.

Because HSC was a block controller, it didn't know anything about what that disk contained, so it would just back up every single block of the disk to the tape. You could do it without interrupting your processing on the CPU, so it was good deal, right?

Once we had an intelligent storage controller, we kind of figured out things that you could do with it. That controller design lasted us - we shipped the first member of the Digital Storage Architecture in, as I remember it late '81 but it might have been '82 …

Burniece: Well, actually the HSC didn't ship until '83 …

Lary: … and the HSC right.

Burniece: … but yeah, the UDA shipped [in ‘82]

Lary: The first generation, which is the UDA. The QDA shipped shortly after that and then the HSC50 shipped in about '83.

Burniece: It didn't have shadowing yet though. That came a little later.

Lary: That came later and the hierarchical part, the caching, came [after that]. Then we implemented a couple of other [DSA controllers] because there were some new Digital backplane buses. There was a new Digital bus called the BI, which was done for mid-range VAXs.

There was also a Digital bus called the XMI, which was a higher speed bus, which was done for the higher end VAXs. So we were, at the time - and this changed, but I won't have time to get into it - a captive storage organization. We built storage for Digital systems period.

Burniece: And it was essentially, 100% proprietary, top to bottom.

Lary: It was 100% proprietary. We had a proprietary bus on the front end and we had a proprietary bus on the back end, plus we had proprietary protocols. They were either patented or trade secrets.

People tried to build knock off stuff and we would sue them. I would get involved in depositions and it was the classic proprietary '70s and early '80s model that worked so well then but stopped working in the late '80s and '90s.

Burniece: It definitely was the fuel of the '80s though.

Lary: We were slow to change and it was one of the things that caused us financial difficulties and ultimately ruin.

Burniece: Alright. We'll talk about that next time we get together.

Lary: We were so proud of what we did that we'd rather do it our own superior way than the standard way.

Burniece: Alright. We’ve got about five minutes left and there are about two things I'd like to have you accomplish in the next five, or so, minutes. Talk about some of the key guys involved in the development of the HSC 50 and what roles they played.

Lary: Right.

Burniece: People like Bob Bean, Barry Rubinson, those kind of guys.

Lary: Yup. and there are going to be names that I'm not going to remember now that I'll add later.

Burniece: We can fill them in later.

Lary: But Bob Bean was basically in charge of the software that went in the central processor, in the p.IO.

Burniece: You named him earlier, because he was part of the, what, the PDP-10 team?

Lary: He was the architect of RT-11 …

Burniece: RT-11. Ok.

Lary: … the operating system for the 11 and, of course this was a PDP-11, so he felt right at home. He did not use RT-11 as the Kernel for it. We built kind of a bare iron Kernel that it ran on, because the key when building these storage controllers is you have hundreds of balls in the air at once. You have hundreds of I/Os going on at once and, so you're constantly context switching as events occur between them. Thus, you want to implement a system with minimal context switching.

So you start using things like lightweight threads and weightless threads and things like that, implementation techniques that aren't necessarily compatible with any of the operating systems that we built at Digital. You do that, because you got limited CPU power, so Bob was the lead designer on the software in the PIO. I contributed a lot to the K.disk stuff.

Burniece: Well, you basically did all the coding on the 2901s, right?

Lary: At that point, I had taken more of a floating role. I wasn't responsible for any one part; I was a floater - a utility infielder. So the guy who was originally in charge of the code there, I'm pretty sure, his name was Michael Beckmann.

But he had something going on with his parents. So he married a Chinese woman named [Hsu-Shia Chow and changed his name to [Chren-Ling Chow] to kind of give his parents the finger. So he was the lead software guy on the disk interface.

The lead software guy on the host interface, and I might be wrong here, was a guy named Bill Grace.

Burniece: That may be true.

Lary: Ok.

Burniece: Bill definitely was involved with the UDA.

Lary: Yeah. He's a good software guy, marathon runner, interesting guy.

In terms of the hardware, I think the guy who pulled all hardware together was a guy named Bob [Blackledge]. He was the Dave Rodgers, - you know, he [Blackledge] didn't do any piece of the hardware but he had the overall responsibility for making sure that you could plug these boards in and out and they'd work. In fact, I believe that we supported the dynamic insertion and removal of those boards.

Burniece: Yes, we did.

Lary: So he had his work cut out for him in terms of being able to do a hot-swap of boards within an HSC and things like that. On the controller side, I have to think. I'm drawing a blank.

On the drive side, the electronics in the drive side, I think we had guys like Winston Sergeant working on that.

Then, of course, we had Pete McLean, who actually was a pretty good guy in terms of driving design in general, knew about servos, knew about read / write, but I think he also took a role in building the SDI electronics. I think his fingerprints are on that serial bus, his and Winston Sergeant's fingerprints are on that serial bus, and their names are probably on the patents.[3]

I'm trying to think who else comes to mind. Barry Rubinson was kind of the HSC overall architect. He ran the MSCP process, because at some point, it ceased to become our little way of getting a standard interface and became a corporate DEC standard. Various operating systems would want to add features, so there was an actual corporate-wide architectural process. He ran that, as well as some other processors.

Burniece: Talk about that a little bit. Because there were three major organizations involved, at least, in the VAXcluster. One was Colorado doing the HSC controller.

Lary: Right

Burniece: and everything that had to do with the controllers and the disks. The other was the VMS world ….

Lary: The VMS group.

Burniece: … with the lock manager and all that. The third was the CI group, which was part of the VAX group itself under [Bill] Demmer. So you had those three organizations.

Lary: Well, we had more than that, because remember, we had an architecture. There was a group in Shrewsbury that started up but this was a little later. This kind of overlapped with SCSI time. We decided we needed to build a low end …

Burniece: Yes.

Lary: … disk bus. We called it DSSI, Digital Small Storage Interconnect or something like that.

Burniece: Which was a SCSI takeoff.

Lary: It was based on the SCSI transceivers and the SCSI wiring but, of course, it ran the MSCP protocol.

Burniece: And it was also dual-ended.

Lary: We were Digital and we really didn't believe in industry standards at the time - ultimately, our downfall. So there was a group in Shrewsbury with people like Fred Zayas, Rich Wrenn and guys like that

They had a say in the MSCP standards and eventually, the Unix group wanted to deal with our controllers. Digital was late to the party with Unix but we eventually had several flavors, Ultrix and DEC Unix and all this stuff.

So it really was a corporate architecture process to the point where we had people whose full time job was to be the caretaker of particular pieces of the Digital Storage Architecture.

Burniece: In fact, [INAUDIBLE]--

[3] [Interviewee's note] Two other key guys on the drives were Mike Hammer (RA81) and Bert Miller (RA60), who had overall responsibility for those two early DSA drives.

Lary: Ed Gardner was to start and then it passed to a guy named Sid Snyder. He was the right MSCP architect. Anything that needed to be done to MSCP, if you had a question about it or if you had a change you wanted to propose, you'd propose it to him.

There'd be a process. Everybody would have to buy into the change, so it became a very cumbersome standards process, because Digital was such a big organization I wound up managing the group. Alan Kotok actually moved from CPUs, et cetera, to storage at some point and he wound up managing the architecture process …

Burniece: Yes, he did.

Lary: … and held the coordination meetings between the operating systems. In fact, there was a monthly kind of storage get together. Sometimes it was in Colorado. Most of the times, it was in Massachusetts, because most of the attendees were in Massachusetts.   Alan did those until the time that he left Digital to join, I think, the World Wide Web Consortium.

Burniece: Something like that, yeah.

Lary: He became a key architect in the W3C and I took [the storage architecture group] over from him and ran that process.

This is where the CPU guys would say what they were doing next and what they needed from storage. Then the storage guys would say what they were doing next and what they needed from the CPU guys. Spirited discussions would ensue and, hopefully, at some point consensus would be reached. Digital was very much a consensus company and then we'd move forward on it.

Burniece: Wrap it up with just a few comments on one of the guys who was probably at the peak of this whole thing with his concepts and views, and that was Mike Riggle. What was his role? Then tell us a little about Mike.

Lary: So Mike was - and this was before I joined the group - I think, the guy who saw the need for intelligent storage. I don't know if somebody whispered into his ear ….

Burniece: No, he definitely did.

Lary: …. or he came up for it by himself but he saw that if we didn't apply intelligence to the storage, somebody else would and leave us in the dust.

Burniece: In fact, they were [when he joined DEC in ~1975].

Lary: Being a density and read / write speed guy, as well as a general leader and thought leader, he also saw that you could translate intelligence into better performance, better density and things like that. When everything got moving on the Digital Storage Architecture part, Mike kind of detached from that and went more into an advanced development role. He would lead the creation of new processes. He also, I believe, got the storage group involved, for instance, in databases …

Burniece: Yes.

Lary: … which was kind of a neighboring technology that he thought we needed to work with as part or our ideas with storage, which gets into how I met my second wife, which will be a topic for another day.

In general, he drove the strategy - not so much how we should build things, but what should we build and what technology should we invest in

When the late '80s occurred, the disk industry went through this revolution going from iron oxide [media] to plated pure metal media and from wired heads to thin film heads. All of a sudden motors started to collapse, as we started to be able to do things in smaller form factors. Mike was really the guy who took the technology stew out there, the component technology stew out there, and figured out what we were going to build.

Burniece: I agree with you 100%. We're going to cut it at that point. Mike was probably the finest engineer I ever met in my life. I gave the eulogy at his funeral and he was just a remarkable guy.

Lary: He was also a gentleman engineer.

Burniece: Absolutely.

Lary: He personified the idea of you can reach a consensus and drive ideas through the system without bullying anybody.

Burniece: Who was the guy that did the RISC processor for DEC that used to work in the storage group?

Lary: The which?

Burniece: The RISC processor for DEC, the Alpha, who used to work in the storage group for Mike.

Lary: Dobberpuhl?

Burniece: No. That's not the right one. I'll get it later.[4]

Lary: Ok.

Burniece: But at [Mike Riggle's] funeral, I quoted him about Mike as the finest engineering manager he ever met and why.

Lary: Yup.

Burniece: And I don't remember the exact quote. But it was great.[5] Anyway, we're going to break it off. We're going to have to resume this …

Lary: Yes, indeed.

Burniece: … to go forward from …

Lary: Oh, boy.

Burniece: … we're now sitting there somewhere around 1986--

Lary: It turns out …

Burniece: … and we haven't got [to the rest if the DEC story, let alone to what you did next]…

[4] [Editor's note] The name was Mike Supnik.
[5] [Editor's note] The actual quote was: "Mike was my role model as an engineer and as a manager, and his example still stands unchallenged."

Lary: I'll be back in the Bay Area in two weeks or so, three weeks.

Burniece: Thanks, Richie. It's been great.[6]

END OF INTERVIEW[7]

[6] [Editor's note] The session of this oral history was recorded on May 3, 2018.
[7] [Editor note] Tom Burniece completed initial edit of the transcript on November 3, 2017. Richie Lary finished his editing on May 2, 2018. Tom did a final cleanup pass on July 24, 2018.