Oral History of Dick Sites

Interviewed by:
Dag Spicer

Recorded December 3, 2018
Mountain View, CA

CHM Reference number: X8868.2019

Spicer: So today is December 3rd 2018. We're here with Dr. Dick Sites, who's going to chat with us about his life and his career, which has been a very distinguished career. Dick, thank you for joining us today. It's...

Sites: Oh, you're quite welcome.

Spicer: ...wonderful to have you here at the museum in Mountain View. So let's begin at the beginning when you were a kid in school, let's say 10 or 12. That's usually when we first see the spark of whether you're interested in science or other things. Can you tell us a bit about your early school years?

Sites: Sure. First, I was born and grew up in an odd city, Oak Ridge, Tennessee, that had been established during the War [World War Two]. So it was a city that had very few southern people, and even though it's in the South, it was full of bright physicists and other scientists, and the school system was well funded and one of the best in Tennessee. So, let's see, 10-year-old.

Spicer: Sorry. Were your parents working at the labs or ...

Sites: Yeah. My dad worked at the labs, and helped separate Uranium and [he] ran a mass spectrometry lab. So, in my alternate biography, I start with, "My ~~my~~ dad was a bomb builder."

<laughter>

Spicer: Right. So he's a physicist, I guess, by trade?

Sites: Yeah.

Spicer: Okay. And your mom was a homemaker or...?

Sites: Homemaker, and she also did some teaching and tutoring.

Spicer: Oh, great.

Sites: Mostly in mathematics, and I was surrounded by playmates whose fathers, and [in] some cases mothers, were famous scientists. So, in the summer of 1959, one of my friend's dads, Bob Coveyou, took three of us [kids] sort of as an experiment and taught us how to write programs. We just went over to his house on Saturdays for most of the summer and he ... walked us through how to write some programs for this one-of-a-kind machine, and we scratched stuff out on coding sheets and he would take it into work and ...

Spicer: And sorry, what was this machine that did ...

Sites: It was called the ORACLE, Oak Ridge Logical ... Oak Ridge Compute-something and Logical Engine. [Oak Ridge Automatic Computer and Logical Engine].

Spicer: Right. And that was when you ...

Sites: And I only learned 20 years later that he was one of the people who helped build it. <laughs>

Spicer: Wow. And sorry, what was his name again?

Sites: Bob Coveyou C-O-V-E-Y-O-U.

Spicer: So ORACLE was one of the 17 von Neumann machines [modeled after the Princeton IAS architecture]; was it not?

Sites: Yes.

Spicer: The original ...

Sites: Exactly.

Spicer: ...Institute for Advanced Study ...

Sites: Vacuum tubes.

Spicer: ...Architecture.

Sites: Had a CRT display that you could turn on some points.

Spicer: So did you get to actually use the computer?

Sites: Well, the computer, of course, was inside a lab building that was inside a security fence.

Spicer: Oh.

Sites: They did have open houses about once a year, so I got to see the machine through a glass wall, but never actually touched it.

Spicer: So when you were doing programming, would the professor bring you programs over to the machine himself and run them for you or...?

Sites: Yeah. Bob would take them into work and they would get whatever was done to get them into the machine and they would run and he would bring back the results, which initially were pretty crummy. But

eventually [it] ended up [that] by the end of the summer, the program would print out Pascal's Triangle, and I got interested in computers. <laughs>

Spicer: What age are you at this point?

Sites: I was 10 years old.

Spicer: Ten? <laughs> Oh, my gosh.

Sites: You suggested starting around 10.

Spicer: Yes. Wow. <laughs>

Sites: So I wrote my first computer program in 1959.

Spicer: Oh, okay.

Sites: And the Museum actually has the output from that. No, I take that back. I have the output but the Museum will have it.

Spicer: Oh, thank you. Yes, we'd love that.

Sites: <laughs>

Spicer: I certainly know about the ORACLE machine. That's wonderful. Did you ... how long did your family stay in Oak Ridge?

Sites: Oh. They stayed for 40 or 50 years.

Spicer: Oh, okay.

Sites: They were there during the war. When the war was over my dad went back to grad school for a year or so and then they returned to Oak Ridge in the, I don't know, 1947 or 8. I was born in '49, and then they lived there up until sometime after 2000. They moved to Colorado to a retirement home near my brother and sister, and they died two years ago at age 95 and 96.

Spicer: Oh, my goodness. That's wonderful. Yeah.

Sites: So good genes.

Spicer: Yes, yes, exactly. So 1949, you're 10 years old.

Sites: 1959.

Spicer: I'm sorry, 1959.

Sites: Yeah.

Spicer: Yeah, you're born in '49. 1959 you're 10 years old. What was school like for you? Were you a pretty good student?

Sites: Yeah. I actually skipped second grade. I was born in February, and so I just missed the December cutoff. So most of my playmates before kindergarten went to school a year before I did, and then I had to wait until I was well into five and a half before I went to school, and so skipping second grade I sort of caught up with other people I knew.

Spicer: Right.

Sites: And then in fifth and sixth grades, I had a teacher who took six of us under his wing and taught us math separately from the rest of the class. We had little tables on the side during math period and he would take us off, and brought a bunch of supplementary material. So we got quite ahead on mathematics, the six of us, and it turned out he got changed from fifth grade to sixth grade the same year I moved from fifth grade to sixth grade, because I got assigned to him again, and he eventually got in trouble with the school board for teaching us separately, <laughs> and we had to abandon that partway through sixth grade.

Spicer: Oh.

Sites: But the net effect was by seventh grade the six of us took Algebra I ~~1~~, and then as the school people looked ahead they realized they were going to sort of run out of mathematics for us, and in the end, they arranged for [three of] us to skip eighth grade. I had to do summer work of eighth grade English from a self-teaching book and write a theme on the ... was on the Civil War.

Spicer: Amazing.

Sites: It was a crummy theme and it was handwritten and it was messy, but they put up with it.

Spicer: <laughs>

Sites: So I was 16 when I graduated from high school.

Spicer: Wow. Did you have any dreams at that point about what you'd like to do with your life?

Sites: I knew I was very interested in computers. I spent high school time writing some programs for computers at the labs through my father, who would take them in and get them run and bring them back. I think partly to keep me off the streets, and so I ended up writing some programs for the CDC 1604, one of Seymour Cray's early machines, 48-bit, and then the summer after my senior year, summer of '65, I

worked at the University of Tennessee Computer Center in Knoxville, so I lived on campus over the summer, and they had an IBM 7040, and a 1401, so one of my favorite machines is the 1401. I love the [1401] lab here [at CHM].

Spicer: Oh. Good. Yeah.

Sites: Once when I was interviewing people at Google much later, we had an interviewee whose resume said that he knew 1401 assembly language, and I said, "Oh, great." So he sits down. I say, "Oh, I see you know 1401 assembly language. So if you were punching a program on a card, what's the character you'd put in Column 1?" Right? Because on the 1401 you can just type in a program and have it boot the ... read the first card and branch to location one and execute whatever those 80 characters are, and the 1401 has this interesting property that it's a variable length machine with word marks to delimit fields, and word marks also delimit instructions, and so when you boot a card, it clears the memory, including the word marks, loads locations 1 through 80 from the card branches for location one, and the first thing you have to do is you have to set the word mark at the end of the first instruction, otherwise it never finishes. <laughs> And the character code for word mark is comma.

Spicer: Okay.

Sites: So I asked this interviewee, so what'd he type in Column 1? And he sort of blanches and back pedals and says, "Oh, I didn't actually program the 1401, I studied it," and he had the bad luck to hit one of the only people in town who could really grill him on 1401s. <laughs>

Spicer: Yes. Yes. Well, sadly, I've heard from other recruiters that, let's be generous and just call it skills inflation, is somewhat common.

Sites: Yeah. We can get to that when we talk about ...

Spicer: People say they know ...

Sites: ...interviewing people at Google later in this interview.

Spicer: Yeah. He probably thought he was safe picking the 1401. <laughs>

Sites: Yeah. But I really enjoyed the 1401, because I could. I had lots of time. The computer center was open to me, and I could type up programs and run them and I did.

Spicer: Well, you ought to come down here and bring some old programs. <laughs>

Sites: I don't have any left with me.

Spicer: Okay. Yeah.

Sites: I do remember a lot of the utility programs for that machine. You'd boot a card deck in front of the cards you wanted to actually list or whatever or process, and for frequently used card decks they were punched on cards that had Mylar edges, and they would last much, much longer than just the cardboard edge ones.

Spicer: I've never heard that.

Sites: Oh. It was wonderful.

Spicer: Interesting.

Sites: So...

Spicer: Yeah. Okay. So 1401, 1965.

Sites: Oh. I left out a couple things.

Spicer: Oh, sure.

Sites: A couple [of] summers during high school, my parents sent me off to National Science Foundation math camps, so I spent a summer at the University of North ... I'm sorry, University of Florida, and it was a summer math camp with math and geology and computer programming, and so we wrote programs in FORTRAN for [the IBM] 709, vacuum tube machine, and then the following summer at Purdue University, which had an [IBM] 7090. So by the time I got out of high school I had actually been actively programming since I was 10.

Spicer: <laughs> And on some of the leading machines of the day as well, leading scientific ...

Sites: Some of the existing machines of the day. <laughs>

Spicer: …machines. But a 7090 was a very powerful machine at the time.

Sites: Well, at the time, of course, IBM was giving 90 percent discounts to universities.

Spicer: Oh.

Sites: So lots of universities had 7090s.

Spicer: I see. Ninety percent discounts?

Sites: Yes. Yeah.

Spicer: Wow.

Sites: On the rental, of course.

Spicer: Right.

Sites: <laughs> But the universities had to pay for the electricity.

Spicer: Right. Okay. So we're in summer of '65 here roughly. What, you're a senior? No, a junior?

Sites: Summer of '65 I had just finished my senior year.

Spicer: Okay. Just finished senior year.

Sites: And I had had my driver's license for about four months. <laughs>

Spicer: That's very nice. Nice milestone, and I keep remembering, yes, thank you for bringing that up. Keep remembering, you're very young here. Much younger than your cohort, than that year's ... than a normal cohort would be.

Sites: Yeah, yeah. Sophomore year in high school I flunked phys ed, because I was two years smaller than everyone else, and it was those two years where the 17 and 18-year-olds all had muscles and things. <laughs>

Spicer: Oh, that's nice.

Sites: My parents had to sort of go off in a different room in order to laugh about it.

<laughter>

Spicer: So at this stage, you're working ... you're at MIT; is that right?

Sites: I went to MIT as an undergraduate. I met someone at a Purdue Summer Math camp who lived in Boston and was going to apply to MIT and that sort of just put the idea in my head, so I applied, which appalled my parents. Partly because it was going to be expensive if I got accepted, and partly because they weren't at all sure I'd get in. So they sort of encouraged me to have some backup plans. The last person from Oak Ridge who had applied and gone to MIT flunked out, and so they were a little worried about that, but in the end, I applied to Duke and Harvard and MIT, and Harvard, of course, turned me down, and Duke accepted me and MIT accepted me.

Spicer: And you say "of course." Why would that be at Harvard?

Sites: It would've been a bad match all around.

Spicer: Oh. Oh, okay.

Sites: And they figured that out. <laughs>

Spicer: Oh. Okay. Just, and I'm curious, how not ... they don't like younger scholars or...?

Sites: No. I think I would've done badly in humanities.

Spicer: Oh, okay. So you would've changed discipline then if you went to Harvard? No?

Sites: I don't know. I just ...

Spicer: You don't know? Yeah.

Sites: I'm not sure why I applied to Harvard, but it seemed like it would be good to be able to say I got accepted there, but yeah.

Spicer: Oh, yeah. Yeah.

Sites: <laughs>

Spicer: Okay.

Sites: But I did get accepted to MIT.

Spicer: Yeah. Well, MIT is pretty darn good, and so you're doing a bachelor's in mathematics at that stage?

Sites: Well, I ended up graduating in mathematics. I didn't start in mathematics.

Spicer: Tell us about your ...

Sites: But I have to back up about applying. I had a lot of help in my application. Not writing the essays and stuff, but I had a lot of people in the background writing recommendation letters. Because I was so young, I think. So there were a dozen teachers and scientists who sent recommendation letters for me. I only learned this later. My high school math teacher Mary Laycock arranged lots of people to pitch in and say a few good words about me, and I found when I sat down 10 years ago and wrote for our kids a list of all the jobs I had and how I got them, it turned out to be seven pages, and I realized when I finished, every single job, there was a specific named person or set of people who helped me get that job. Every time, all the way down the line, and Mary Laycock was one of the first people who really went to bat for me and helped get me ushered in to MIT, where I was a mediocre student. <laughs>

Spicer: Aw. Well, I'm sure that you're being modest.

Sites: Oh, no. I was on academic probation my senior year.

<laughter>

Sites: Second semester of my senior year.

Spicer: Oh.

Sites: But I ...

Spicer: Maybe ...

Sites: ...went there knowing I was interested in math and science.

Spicer: Right.

Sites: I got an A in freshman physics, which was a great pleasure to my father, who was a physicist. It was all downhill from there, but was a good start.

Spicer: Wow.

Sites: And they had computer courses, which I took, but it was a year ... I graduated one year before they had a computer science major. So I took what were essentially the trial versions of many of the courses, but I couldn't graduate in computer science, so I was, my sophomore year I was in electrical engineering, until I had to take the course with Dirac's equations and working out the charges and spin states of hydrogen atoms [Schrödinger equation], which I was uninterested in and did badly on. So I ended up transferring into math and did okay there until my senior year, when I took a point set topology course. I work in pictures, in heads. I do lots of drawings. I do lots of graphical output. I do nothing good with theory, <laughs> and so I'm in this math class and it's all theoretical sets of points and the tests are true-false, right-minus-wrong, and so I flunk the class. It was after the drop date before I realized, "Here's a math class that's over my head," and so I was on academic probation, having flunked a required math course my senior year. So I'm off skiing in Colorado with my friends from IBM, where I had worked over the summers and also, as it turned out, my fiancée, and I get back and there's this pile of many, many days of frantic phone calls from my parents, messages waiting, handwritten messages. This is before phone answering machines, and so I call them and they're panicked thinking about a fifth year of tuition.

<laughter>

Spicer: Right.

Sites: And I didn't realize until I got back from skiing that I was on academic probation, so...

Spicer: Ah.

Sites: Fortunately, I knew before I called them, so I could understand why they were calling so much.

Spicer: Right.

Sites: So I talked to my advisor in the Math Department. He said, "Look. Take my freshman probability class. You'll get an A and you'll graduate." So I graduated with a B- average. I was, in fact, a mediocre student.

Spicer: I'm not going to argue with you.

Sites: <laughs>

Spicer: Well, that's a great story. Now, one of the things I noticed in your resume was that you, and you just brought it up now, was that you worked at IBM.

Sites: Yeah.

Spicer: It seems like [you worked at] almost <laughs> all of their labs, there's something, you know, at all of these different places. Yorktown, San Jose… Can you tell us a bit about how you got started [at IBM] as a summer job?

Sites: Well, when I was at the University of Tennessee the summer after my senior year in high school, I decided that I would like ... well, my parents made it clear I needed to work part-time while I was on campus. By then I'd been accepted, so we knew where I was going. So I decided I would interview at IBM and see if they had any part-time jobs in the Boston area. So I talked to the customer engineer who was servicing the data center at UT and also the head of the computation lab, and they wrote off to someone at IBM Poughkeepsie, and I gave them the dates we would be there driving up to campus, and so we drove up to Poughkeepsie and I showed up at the front entrance of the IBM Galactic Headquarters and got some receptionist who had never heard of me and had no idea what to do with me and spent some time on the phone, and managed to get someone in HR to get someone in the operating system [section] to chat with me for half an hour. So we went over to that building and I ... this guy explained to me about building operating system images on disks, which was completely foreign to me. I had no idea what he was talking about or even that you had to build at that time a separate version of the operating system for each installation. This is before stuff would be done at boot time, to find out which tape drives were there or whatever. This was all pre-built, partly because the disks weren't very big and there wasn't ... it would have taken minutes and minutes and minutes to boot it. It had to do all this computation to decide what drivers to load.

Spicer: I see.

Sites: This is in an era where drivers were limited to be ... to fit into a thousand bytes, <laughs> and some of the 360s had eight kilobytes of memory.

Spicer: Sure. So rather than the OS going around and polling every ...

Sites: Yeah, that all came much later.

Spicer: Every peripheral. You designed a fixed system.

Sites: We preconstructed a disk image that would roll into memory and everything was there.

Spicer: Right.

Sites: For that particular machine.

Spicer: Right. So consequently, when you added or removed a peripheral.

Sites: Yeah. Then you had to rebuild the operating system.

Spicer: Right.

Sites: So I didn't know anything about this, the guy explained to me. So went off to drop me off at college and I didn't hear back or anything, and then I got a telegram about a week later that said, "Call Len Page," and a phone number, at IBM," and IBM Boston Programming Center at the time was in a building right at the edge of the MIT campus. It was actually in Cambridge, and they actually wanted to talk to me about a job, but they had no way of figuring out how to reach me, because I had a ... I didn't know my address when I was in Poughkeepsie. So they sent a telegram to the dorm and it got to me, and so I called them and then I walked over and I had written a letter, I guess, saying that I was interested in working [at] IBM, what I've done and that I was particularly interested in this new programming language, PL/I, and a few other things that were incredibly naïve, <laughs> and they were fascinated apparently by this. So I went over and they chatted with me and they decided to hire me for 10 hours a week for $2.50 an hour. So I, you know, reported to Elaine Bond and they were doing this software called FORMAC, FORmula MAnipulation Compiler, which they thought in September of '65 would become part of the PL/I language. You could do symbolic algebra inside that language, and that -- as IBM actually tried to get to the point where they could ship System/360s and ship matching software -- that got pared out and turned into a prepass that would turn into a whole lot of subroutine calls within the PL/I language, a bit like the C preprocessor. I ended up working on the automatic simplification software in System/360 assembly language with lots of linked list data structures, so I got a whole bunch of little magnets and put labels on them and put them on the blackboard, my whiteboard, with all these diagrams and moved pointers around as I wrote the code. It took forever. I was very slow. It's something that one of them could've, one of the full-time people could've knocked off in two weeks. But I learned a lot, and in the end, it worked. So I ended up working part-time during the school year and summer of '66 and '67, and then summer of '68 I ended up working at IBM in Boulder, Colorado. The group who worked on the PL/I compiler in Hursley, England, had moved back to the 'States and moved to Boulder as a group.

Spicer: Oh, wow.

Sites: And started a programming lab in Boulder.

Spicer: I didn't know that.

Sites: And Jim Cox was the head of that group.

Spicer: That's a big move for that team.

Sites: Well, they were all expats in England for two years at Hursley, and so they were all moving back.

Spicer: Oh. So they're already Americans then?

Sites: Yeah. Yeah.

Spicer: Okay. Got it. Right.

Sites: And so Jim led this group to move to Boulder because they liked Boulder, and at [the] programming lab they were going to build an optimizing PL/I compiler that would do much better than the first effort at Hursley. Meanwhile, in IBM's usual way, they had a second optimizing compiler effort in Hursley and funded them both for a couple years. Partly by fluke, and partly in this long list of jobs and friends, Jim Cox, who was the head of that lab, had worked in Oak Ridge and in fact had been a manager of my brother, my older brother, for a summer job that my older brother had in the early '60s. Maybe late '50s. So Jim knew the family. <laughs>

Spicer: Right.

Sites: And when I got a recommendation from Elaine Bond in the Boston Programming Center, [Jim] said, basically, "You're hired," and they gave me a big raise to 2.75 an hour.

Spicer: Oh. Great.

<laughter>

Sites: So I went out and had the summer there and had a great time and did the same thing the following summer.

Spicer: That's amazing, and I'd love to ask you a couple of things. One is, okay, you mention a brother, which is ... it's on me. I should've asked you if you had any siblings.

Sites: Ah.

Spicer: And then the second question is I'd love your general impressions about the System/360 as an architecture.

Sites: Okay. First, I have a brother who's five years older, a sister who's two years older and a sister who's eight years younger.

Spicer: Oh. Great. Thank you.

Sites: Let's see, the 360. It made a big impression on me. It was announced in April of '64, and I paid attention to that announcement and paid attention the following year as they got closer to actually shipping some, and then when I got to Boston and got hired by IBM, I immediately got one of the green cards and programming manuals and stuff and learned the [System/360] assembly language. Also wrote a little bit of FORTRAN, but it was most ... and PL/I, but it was mostly assembly language, and so I learned the user mode part of the architecture pretty heavily. I read all the [IBM] System Journal articles by Fred Brooks and Amdahl, and really liked the distinction they made for the first time in the industry between the architecture and the implementations, and so I had a lot of respect for that. I liked the instruction set since the fields are regular, they're multiples of eight, multiples of four bits. The base registers gave a good way of addressing lots of memory without big addresses, which, of course, had always been the problem in the 7090 and the GE 645 that Multics was using at MIT at the time I was there. They [GE] were able to increase the address to 18 bits instead of 15, but... So I really liked the 360. They moved the industry over, as they intended, over this huge range of processing power.

Spicer: Mm-hm. What about on the software side? They struggled a bit with software; did they not?

Sites: Well, OS/360 was late, and they split off DOS/360 partway through that, and this ... '63, '64, '65 internally at IBM, as you well know from having gathered history of all of this, was a time of great turmoil and great financial strain. But they pulled it all off with, for this industry, not much in the way of schedule slippage, and they got six different machines shipped to customers and they mostly worked on arrival. So let's see. OS/360 was slow, but miraculous. You know, it used disks instead of tapes for storage and got better every several months. I didn't have, as a young student, I didn't have any insight into much of the issues and problems. I remember when the printer spooling software arrived and actually worked so you could keep the printers running while some other job was really doing work.

Spicer: Is it ...

Sites: Before that you'd print a line and everything would stop until the printer had finished ~~stopped~~ or ...

Spicer: Was that HASP?

Sites: No. It was the year before HASP, I think. It was just a printer spooling.

Spicer: A spooler, yeah.

Sites: Yeah. And I was pleased to read that it worked quickly by lining up 60 channel command words, one for each line, and pointing the channel at that list and it would go through the list and print an entire page without using the CPU and then it set an interrupt at the end of the page.

Spicer: Right.

Sites: And there was a fairly tricky piece of code to deal with what happens if there's a paper jam in the middle, where do you restart?

Spicer: Mm, yes.

<laughter>

Sites: And all this is done with little kilobyte overlays. So...

Spicer: Yeah.

Sites: From an engineering point of view it was quite incredible, and hugely ambitious.

Spicer: Did you use the system 67 [IBM System/360, Model 67] at all, the timesharing [version]?

Sites: I did not use the 67. I was in the building. The people doing the translation buffer hardware were actually in the same building as the Boston Programming Center, but they were one floor up. They were on the fourth floor, and that was the Cambridge Programming Center. <laughs>

Spicer: Right.

Sites: And they built a model 47. They took a model 40 and added ... this was a one-of-a-kind machine

Spicer: <laughs> I've never heard of it.

Sites: ...and added to translation [hardware and] software.

Spicer: Yeah, wow.

Sites: The DLAT, Dynamic Look-aside Address Translation, and got it up and working, and that was the prototype for the model 67. They were under tremendous pressure at that point because MIT had decided to use the GE 645 for Multics and had pushed IBM out, and IBM, see, determined to get back in the game.

Spicer: I know from reading a few biographies of the Watsons [IBM's first two leaders] that when they lose a major account it ripples through the whole company and they take it extremely seriously.

Sites: Yeah.

Spicer: I think when Prudential went to UNIVAC, that's what got IBM, "Old Man Watson," started to pay ...

Sites: Watson, Sr?

Spicer: Watson ... yeah, sorry.

Sites: Yes.

Spicer: I shouldn't call him old man Watson. Watson, Sr., yes. That's kind of what kicked it off was a competitive poke. <laughs>

Sites: When I was at MIT, they also had on campus, an [IBM] 7090, that had been modified to have twice as much memory.

Spicer: Oh.

Sites: So it had the 32K words, 36-bits words of memory, and another complete box with, these are boxes, with another 32K and a couple of instructions that would flip a bit of bank A or bank B and then they built the Compatible Time Sharing System [CTSS] to run out of the second box on that machine at MIT to give lots of interactive terminals -- timesharing.

One of the things -- and IBM had time on that machine a few hours a week by arrangement. And they did some of the 360 operating system bringup from tape kinds of processing on that machine in order to get a 360 bit-image that they could use. It also, well ...

But one of the things I found using the CTSS timesharing system, [was that] I read that the [during the] IBM 7090 instruction fetch, you could fetch an instruction that had an indirect bit that pointed to a[nother] instruction and you could have it point to an indirect bit. But it also had an execute instruction that you ~~You~~ could by mistake or maliciously point to another execute instruction that could point to another execute instruction and if you had an execute instruction that pointed to itself, it would go into an infinite loop in instruction fetch and the STOP button on the operator's console was only operative between instructions. So ~~so~~ I had read this in the manual. So I immediately type it in at one of the CTSS terminals and I hit carriage return, the entire room goes silent, these are terminals with Selectric typewriters, this is before screens and everything stopped typing, so I sort of wandered out and went over to the glass window looking into the machine room and there's the operators trying to figure out what to do. They eventually had to power off.

Spicer: By pulling the big power off switch?

Sites: I think there was a power off button.

Spicer: Oh no.

Sites: So I wandered off and didn't go back for a couple of days.

Spicer: Yes, so they didn't trace it back to you.

Sites: But the manual was correct.

Spicer: Wow.

Sites: It's one of the great blessings when I was a student working at IBM [that] they made available to me the entire IBM catalog of manuals, they said, "Just order whatever you want." And so I ordered dozens and dozens of manuals, probably 150 manuals over the course of four years and I just read them all cover to cover and a lot of them didn't make sense and then a few years later they'd make more sense. And it was a huge education and I'm so grateful to the cost center because they were actually paying three bucks and ten bucks and stuff for every manual I ordered, internal charging and it was I think one of the best investments in my education.

Spicer: Yes, they're excellent at IBM with documentation and the Bell system was also very good about publishing research results in the BLR, the Bell Laboratories Record, and [the Bell System Technical Journal]. IBM had two journals too I think.

Sites: Yeah, the IBM Systems Journal and the IBM Journal of Research and Development.

Spicer: Right. Yeah, so that whole journal set too is a wonderful education, reading articles in those.

Sites: Yeah, I read those. And with the manuals, at the time I was there it was a time when IBM was the second largest publisher in the world behind the U.S. Government Printing Office.

Spicer: Oh, my goodness.

Sites: They went through trees and trees and trees. They would ship pallets of manuals with an IBM 360 system.

Spicer: Oh yes, right, and I remember you used to get that big metal rack, about as wide as your arms can stretch out, where all the manuals were lined up near the [operator's] console for reference.

Sites: Yeah, and shelves full of manuals. And then you'd get replacement pages every few months with vertical change bars.

Spicer: Yes, that's right, yeah. Wow. Okay, well thanks for clarifying your time at IBM there as a summer student. What was your job title at the time, intern or…

Sites: Summer student something or intern.

Spicer: Summer student? They didn't have titles probably.

Sites: They didn't actually have any students working at the Boston Programming Center when I showed up so I don't know if I had a real job title. There were a couple of years when I had to fill out the complement of my time on my timecard, I had to put down how many hours less than 40 I did not work.

Spicer: Oh that's funny.

<laughter>

Spicer: That's funny.

Sites: Jean Sammet was there and she got tired of this at some point, filling out timecards so she put down her actual hours and complained when the field only had two digits and she had worked more than 99 hours that week and I think sometime soon after that they sort of relaxed the rules on these timecards for exempt employees.

Spicer: Right. Now since you brought her up, tell me about Jean Sammet and what she's…

Sites: I didn't interact with her very much, she was there at the Boston Programming Center, she was working on her book about languages. I didn't know anything that she would ask me about, so she was sort of off in the corner. I remember that she had at one point lots of galleys with lots of yellow sticky things sticking out so I guess the 3M Post-it Notes had been invented a couple of years before that and so she was using them heavily.

Spicer: I'm curious, do you remember what her function was at IBM, was it to create documentation or…

Sites: Well I think she had started out as the Programming Center Lab Manager but when I was there she was very much into the book writing and IBM had given her a year or two to do that.

Spicer: I see, okay.

Sites: And then I guess, I don't know the time ordering, at some point around there, probably a little later she was President of the ACM I think.

Spicer: Yes, that's right.

Sites: IBM at the time would support a lot of people just because they were good people, I heard stories like the story about somebody at the Time Life Building for IBM in New York City who just sat around and read the newspaper all day and at one point this friend who was telling me this story said he had been there and asked, "What is this guy doing? How come he's getting paid, he's not doing anything?" And they explained the seven million dollars that he had saved with one question in one review and he had covered his salary. This is at a time when a big salary was 20 grand a year.

Spicer: Right, right. Yes. Oh, well that's fun. I wonder what the question was. Okay, a master's degree, you ... let's see what's going on here, tell us what happened with your master's.

Sites: So when I got out of MIT, first I got out by the skin of my teeth, second, I was getting married in June.

Spicer: Sorry, what year was this?

Sites: Nineteen-sixty-nine.

Spicer: Okay.

Sites: And I was too tired of school to apply for a PhD program, a lot of my peers were going off to MIT PhD or to Cal Tech PhD or whatever but I was just, it was too much. So I decided to try to go to the University of North Carolina Master's Program for a year explicitly temporizing to say if I was still tired of school, I'd finish the master's and then go find a job and if I got my energy back I would look at transferring into a PhD program. And I picked the University of North Carolina because by then Fred Brooks had left IBM and started the department there in 1964 and I wanted to take Fred Brooks' computer architecture course. And again, through friends, Nat Rochester at IBM Boston Programming Center suggested to Fred that he should chat with me. And ~~and~~ so one winter day, I have no idea when, which month in Boston, he was in Boston and so he arranged to chat with me after dinner or something so we talked for an hour or so and my application to the master's program was accepted.

Spicer: That's great. Wow. Now you mentioned Nat Rochester, so he's a famous figure in IBM.

Sites: Yeah, he was the head of the lab at the time I was there.

Spicer: Oh, okay, at the Cambridge Lab?

Sites: The Boston Lab…

Spicer: Boston Lab.

Sites: …in Cambridge, on the third floor and the Cambridge Lab was on the fourth floor.

Spicer: And he's an electrical engineer, is he not originally?

Sites: Yeah, he worked on the 701 and the 704. I was there when the patents for the floating point 704 hardware… anyway the floating patents issued and this was 1965 or '66, IBM had successfully delayed the issuing by ten years and at that time patents started running at the time they were issued and ran for another 18 or 20 years. So at that time delaying the issuing was a big win and there was a great celebration around the lab when they issued it because they were Nat's patents.

Spicer: And now delaying a patent is a good idea because you give away to your competitors what you're doing, is that what you're getting at, that it was good that it was delayed because…

Sites: No, no, that you got that many more years of coverage.

Spicer: Oh I see, okay.

Sites: Now the clock starts when you file and so delaying the issuing doesn't buy you anything.

Spicer: I see, and from filing to issuing could be five years or something, right? Which is feeding into…

Sites: And if you have good lawyers who are incented, it can be ten years.

Spicer: I see, yeah, because patents of course are sometimes…

Sites: Yeah. So Nat was really a nice guy.

Spicer: Well, yeah. Who's the other guy, John, he's from the Watson Senior era, John McPherson... Anyway so you started your master's at UNC with Fred Brooks?

Sites: Yeah, so I took Fred's Computer Architecture course, it was a semester and took other courses and I really enjoyed it there but I got my energy back and so I started talking to Fred partway through fall semester about transferring to a PhD program. I also was his teaching assistant in the architecture class which was an additional education and I graded all the homework programs and write-ups on different machines, I graded them all on this odd scale of checkmark, check minus and X. Considering it was a graduate course, I didn't think there was a lot of distinction to be made and I don't like getting judged and I don't like judging other people so checkmark was this is good and X was you missed the boat. And it wasn't until the end of the semester that Fred asked me, "So how would these map into letter grades?" I said, "Well, a checkmark is an A, check minus is a B and an X is an F." And he did something much, much more gracious than that.

Spicer: Put it on a bit of a curve maybe. Yeah. Oh, okay, so that's good so you were a TA for his course as well.

Sites: Yeah.

Spicer: Wow.

Sites: He also partway through the semester asked me to stop answering all the questions and give other people [in class] a chance, which was good.

Spicer: Were you a bit older than the…

Sites: No, I was a bit younger than everyone else.

Spicer: Oh a bit younger even, still…

Sites: I was still two years younger.

Spicer: Right.

Sites: Fred was good at reining me in a bit because I'm pretty ... I was and still am pretty bullheaded and just sort of go charging in so he sat me down and said, "You need to give the other people a chance." I said, "Oh, okay."

Spicer: Oh that's nice.

Sites: But in the end, we had this conversation where I said, "I don't have very many people to talk to in the department at UNC," meaning that I wasn't getting much feedback from other bright students who were challenging ideas or making me think about something. And he agreed and behind the scenes talked to Don Knuth and said, "You should take this guy." And in the end there was a one page letter that he wrote of recommendation but the real work was done ~~doing~~ in the phone calls and so I applied to Stanford and they accepted me.

Spicer: Okay, so now that must have been a very happy day for you I think when you got accepted and to work with Don Knuth who is the, if there's one person in computer science, he's the guy.

Sites: I had met him because I spent the summer between MIT and North Carolina, so the summer of '69, I worked at the IBM Lab in Palo Alto which at the time was on Sand Hill Road, it was the tail end of the ACS Project.

Spicer: Right, right.

Sites: I had applied again through friends to work for Fran Allen who was head of the compilers [group] for the ACS machine and it had transformed during '65 I think from this wonderfully fast John Cocke designed RISC machine to ACS-360 at [Gene] Amdhal's insistence. And then it got slower and slower and slower as they tried to implement the 360 instruction set with five levels of gate delay or something. And so they cancelled the hardware after I had accepted the summer job working for Fran. So I called her up and I said, "Am I out of a job?" and she said, "Oh, no, no, this is IBM, come work for the summer, it'll take them at least until September to cancel the software," which turned out to be true. Sometime in late September they cancelled the software. So I worked for her for a summer on a three-headed optimizing compiler for Fortran, COBOL and PL/1.

Spicer: Wow.

Sites: In the end we didn't build anything of significance but it was a wonderful education summer because we went through the class notes from Cocke and Schwartz teaching at NYU and so I learned all about, from the master, I learned all about optimizing compilers. And then I ended up working for John the following summer in Los Gatos. But the summer before gave me a start on spending time with the ~~master's~~ masters but during that summer, at the behest of John Walters who was my nominal manager on Sand Hill Road working for Fran, he signed us up for a summer seminar that Knuth was running of looking at actual practice in Fortran programs and eventually he [Don] wrote a paper on the Empirical Study of Fortran. And so I was in that group attending every week, I didn't actually do much of significance but there were people there who brought in lots of code from Moffett Field and went through counting variables and how big expressions were and things which turned into the paper. So I had interacted with Don once a week over the summer which John Walters had set up and brought me along on specifically to mentor me, to help me build a network. And so when I applied, Don already knew who I was and then had feedback from Brooks. John Walters actually knew several hours before I did that I'd been accepted because they posted the list at Stanford on the wall or something and John made a point of going over that day, he knew when it would be and calling me and say, "Congratulations." And I got the letter the next day.

Spicer: Wow, that's so incredible. So I want to dig into that in a minute. The name of the person I was thinking of is John McPherson, I don't know if that name rings a bell.

Sites: Yeah, he was a VP at IBM, but I never met him.

Spicer: Right, and he went way back even in the early punch card era in the '30s, '40s. And one of the things that ... we have his papers and one of the things that I saw, it actually fell out, it was written on a tiny slip of paper and he had written, "Compatibility is an absolute brake on innovation." And when you were talking about ACS-360, it reminded me of that because his view was that you need to break with the architecture to create something new. And do you have any thoughts on that, compatibility versus…

Sites: Well I have some experience in this of much later working on the DEC Alpha architecture which was very much a break from the VAX. Should we get to that in a little bit?

Spicer: Yeah, let's get to that. Okay, so Don Knuth, he's now your…

Sites: My advisor.

Spicer: Your advisor and describe to us what you were trying to prove with your dissertation and/or knowledge you were trying to add.

Sites: Oh my dissertation was much later, you know, I got there, I was just this random new grad student and I took Don's course on Fundamental Algorithms and was in fact his TA for that and the book had just come out a year or so before so he was just over 30 years old and I remember he walked into the first class and sat down and opened the book and said, "Everything I know is in this book."

Spicer: That was his "Art of Computer Programming?"

Sites: Yeah, the first book. And he wasn't a great lecturer but it was true, everything he knew was in the book, we had to read it. And of course the graded exercises, the graded by difficulty exercises were quite a revelation and the 50s, the M50s or whatever were the research topics… so there were lots of thesis topics in the exercises in the book that's an unknown area. And while I was there the sorting and searching came out. And so I had all these peers who had become famous, I shared an office for instance with Bob Sedgewick who did the analysis of Quicksort. I shared an office with Ron Rivest who then went off to MIT and did public-key encryption. I shared an office with Lenny Shustek who went off and did the Computer History Museum. Ralph Gorin who started the low overhead timesharing at Stanford after I had left. Also Bob Tarjan entered and exited two years later with his degree much to the discouragement of the rest of us.

Spicer: He got a PhD in two years?

Sites: He got a PhD in two years.

Spicer: <laughs> My goodness.

Sites: It's so wonderful to be in a group of incredibly smart people, much, much smarter than I am because you just learn more.

Spicer: Wow, that's amazing. Well how was your PhD and…

Sites: Oh, it was terrible.

Spicer: Oh, was it difficult?

Sites: Oh, my PhD, I was not a great student at Stanford either and I got very discouraged partway through. Knuth and his wife went off to Norway for a year and a half on sabbatical and so I was trying to do a thesis with an advisor who was oversees before the Internet, before electronic mail and so we would write letters back and forth and they would take a week or so. And I also was in the middle of getting divorced and we were living in Knuth's house, we were the house sitters, they wanted somebody in the house. So I got pretty down about whether to finish and what to do and so I went and sat on the hills behind Stanford for a while one afternoon and looked out over the Bay and just thought about, "Why is it that I'm getting a PhD, what does it really mean to me, what does it buy me, why am I doing this?" And it came down to just a couple of things, I wanted to teach when I got done and having a PhD was sort of a union card for being able to teach in a university. And my brother had finished his PhD in physics a couple of years before and I'd be damned if he was going to finish and I wasn't. So those were the two reasons and it had nothing to do with setting the world on fire or doing a tremendous thesis or something. So I realized, "I just need to finish." This is a great realization. So I wrote a mediocre thesis but its only purpose was to finish but it got accepted, I went home and cried for a while when the last signature arrived, it was just such an emotional relief.

Spicer: Oh yeah, wow, incredible.

Sites: And again, I was pretty much worn down about school at that point, this is spring of '74, so I didn't know what I wanted to do next but I know I was ... I'd been in school my entire life from age five to age 25. So I went hitchhiking for seven months, I did not do the graduating go interview for assistant professorships at places, I said, "I'm not ready for this. This is the one time in my life that I have time and I don't have a house and I don't thanks to the divorce, have a family or wife and I can do what I ... I can travel and I won't be able to do that later." So I hitchhiked up the coast from Stanford to Vancouver and across Canada and back into Maine and down to Boston and eventually back across the states and when I got to Yorktown [NY] on that trip, my friends there at IBM said, "Oh," first they hired me to do a bit of consulting and second they said, "Oh, you should go to Hursley and consult on this compiler they're building for this new internal programming language BLS, Basic something system programming language to try to get out of writing all of OS/360 Assembly Language. So they sent me off on this boondoggle to Hursley in the middle of my hitchhiking trip. And in turn, I arranged a trip from Hursley to Vienna to the IBM research lab there to give a talk on my thesis on proving that programs terminate cleanly meaning trying to prove they don't have infinite loops and that they don't get errors.

Spicer: This is a somewhat ironic topic given what you did to that computing center earlier by shutting down all the machines. That never terminated correctly, did it?

Sites: No.

Spicer: <laughs>

Sites: No, it's a real shock to a lot of computer science students to realize that the world is still in fact analog and that if you vary the voltage levels on chips, they start doing bad things and it's not all ones and zeroes, it's all voltages and currents and sometimes they miss an edge. But anyway, so I got this boondoggle off to Vienna and back and eventually back to New York. So Don was a great thesis advisor…

Spicer: Oh yes, yes.

Sites: …very gentle, quiet, positive. I was concerned about whether I would ever finish my thesis and whether I'd be ready and he said, "Don't worry, I'll tell you when you're ready and you'll graduate," and that was really comforting and that's what happened, I eventually got enough along, he said, "Okay, this is a systems kind of thesis and there's enough there."

Spicer: Now do you think your relative youth compared to everyone else, like how old students were supposed to be, gave you kind of an edge .. people clearly saw, "Wow, this kid," if I may use that word…

Sites: Yeah.

Spicer: …"Is years ahead of everyone else, clearly there's something special there?"

Sites: No, I don't, not at all. Not when you enter with people like Tarjan and Vaughn Pratt, I wasn't special at all.

Spicer: Doesn't matter? Okay.

Sites: I was in the lower half of the class. Let's see, I flunked one of my qualifying exams in formal logic. Fortunately a whole bunch of us flunked that exam and so they decided to do a makeup oral exam for each of us. I didn't know much about formal logic and probably made a bad choice to take the formal logic exam, this is one of like five qualifying exams of which you have to pass three or some such. So I go into this room for the oral makeup and there's a bunch of faculty and I think a few other students and me sort of at the front and I think [Professors John] McCarthy and [Robert] Floyd were both in the room asking questions and they did not get along with each other. It was unusual that they were in the same room at all and they asked me a few questions that I sort of stumbled through and then they got to arguing with each other and I just stood there biting my tongue and just hoping that the clock would run out and eventually it mostly did and they asked another couple of questions. They passed a bunch of us and they passed me on the condition that I would never claim to know anything about formal logic, which I said, "Sure."

<laughter>

Spicer: Oh that's funny.

Sites: And then later when I was teaching in San Diego, out of the blue I got this conference invite from some faculty member in Poland saying, "Would you like to come and talk about your thesis?" And it was a formal logic conference and I delightfully wrote back and said, "I'm very sorry, I have to decline your kind request. As a condition of my graduation, I must never claim to know anything about formal logic." And I'm sure he had no idea what to do with this response, but it was absolutely true.

Spicer: Well I got to say it must have been fairly intimidating to have McCarthy and that level of person grilling you. Was it at all?

Sites: Oh sure.

Spicer: Yeah.

Sites: I was a nervous wreck. I was so thankful when they got to arguing.

Spicer: So you made it.

Sites: I made it.

Spicer: And what year?

<overlapping conversation>

Sites: I'm thinking somewhat by the skin of my teeth, it was not a landmark thesis in any sense, but it was done and that was the goal.

Spicer: Yes, you are “Dr. Sites,” nobody cares after that. <laughs> Well tell us, so what year was that when you graduated?

Sites: I graduated in '74, spring of '74.

Spicer: Seventy-four with the PhD.

Sites: And I got a nice note back from Jim King who had worked in formal logic programs at IBM and he sent me a postcard that said, "This is your first postal doctorate," it was addressed to Dr. Richard Sites.

Spicer: Oh nice, nice. Oh, that's very nice.

Sites: I later worked with him at Adobe.

Spicer: Okay. So what did you do once you had the PhD in hand, did you work, did you travel?

Sites: Well I went hitchhiking for seven months.

Spicer: So you did the hitchhiking thing.

Sites: Yeah. And somewhere around October of that year, I started thinking about jobs and the end of the hitchhiking was February of '75. So around October, I started thinking about what I would like to do when I got back to California and so I wrote to several computer architects that I had read their stuff or studied their machines and said, "I'll be back in February, I'd like to come talk with you." And one of those people was Bobby Creech at Burroughs in Pasadena. So he wrote back and said he'd like to take a little time and chat with me. So I got back and drove down to Pasadena, this is February of '75, went and talked to Bobby for about an hour and he said there weren't any jobs at Burroughs in Pasadena related to computer architecture that ... he had done the B5000 and 5500 and 6700.

Spicer: Is this Robert Barton?

Sites: Creech.

Spicer: Creech.

Sites: C-R-E-E-C-H.

Spicer: Didn't Bob Barton…

Sites: I'm getting to Barton.

Spicer: Oh, okay, sorry.

Sites: So Bobby said that I should go talk to Bob Barton who was working at Burroughs in La Jolla, San Diego. And so he called Bob or something and said, "You should talk to this guy tomorrow morning." And so I drove down to San Diego that evening and went to talk to Bob the next morning. As we were leaving the Burroughs plant in Pasadena, somebody from HR caught on that I was in the building and they were upset that they hadn't been notified first. <laughs> Which both Bobby and I took pleasure in, their being upset. So I went down to La Jolla and drove up to this little beachfront lab with a dozen people or so that Barton was running the next morning.

Spicer: Sounds nice.

Sites: And told him I was interested in learning more about computer architecture and I admired his work on the B5000. And this is around nine thirty in the morning and while I'm talking with Bob, this other guy, Chuck Seitz, walks in, S-E-I-T-Z, who had been a TA in a course I took at MIT, electrical engineering course, and then was consulting at Burroughs. And I had no idea that he was related to Bob at all, but he, in fact, was living in Palo Alto and commuting down to San Diego a few days a week to consult with Bob. So Chuck walks in, looks at me, looks at Bob, looks at me, looks at Bob again, and says, "If this guy's looking for a job. You should hire him." And Bob says, "Okay. You're hired."

<laughter>

Sites: And he gets on the phone to Burroughs headquarters to talk to some HR person about hiring me and he rattles off my name and address. And in the midst of this, they're having some discussion about salary, that I'm only ... I'm sitting there. Right? I'm hearing one side of it, but I don't know what the person on the other end is saying. And then, I hear Bob say, "Yes, he has a Ph.D., but he's also useful."

<laughter>

Spicer: That's funny.

Sites: So they offered me $18,000 a year, and I took it.

Spicer: So that was your first job as a doctoral ... as a Ph.D.?

Sites: Yeah.

Spicer: For Burroughs in ...

Sites: In La Jolla. A suburb of San Diego, right on the beach, overlooking the La Jolla Cove. Bob had chosen this building, which was mostly a restaurant, because it had a couple of big rooms looking out over the beach.

Spicer: Right. Sounds like a lovely place to work.

Sites: It was about 12 people. It turned out ... the setting was lovely. It turned out Bob and I are both very opinionated, or he was. And in the end, we didn't get along. So I only lasted a year. I made the mistake at one point of asking a little too loudly why someone else who was working there was there at all and didn't seem to be doing anything very useful. And it turned out that she had a relationship with Bob. <laughs> And that was unexpected for me, and I was on the black list after that. So I ...

Spicer: I see that. What kind of technical work ...

Sites: So I ended up working ... going after that year to teach at the University of California in San Diego. But what kind of work ~~worked~~ at Burroughs?

Spicer: Yeah. What kind of technical work ...

Sites: It was a little standalone lab trying to build computing hardware without the use of microprocessors. And the focus was on serial processors. The goal was very, very low-cost. It turned out Chuck was there working on a video monitor that used a standard TV screen, complete with interlacing and stuff, showing text. But showing text with four levels of gray instead of just green and black or whatever. And by using the two bits per pixel, you could get pretty readable text with terrible resolution. So you could get the O and the cut-out in the middle of the "e" dark enough that people could see that it's an "e".

Spicer: I see.

Sites: And so, Burroughs actually shipped a bunch of these for a while as very cheap terminals, just using standard TV screens and some electronics.

Spicer: And the technology was TTL?

Sites: Yeah. It was all TTL.

Spicer: Yeah.

Sites: And we built some rotating loop storage stuff and the processing serially, bit by bit on the fly.

Spicer: You mean like a delay line or ...

Sites: Small amounts of electronics. This was sort of on the verge of DRAMs.

Spicer: Oh.

Sites: And I built at one point an interface to a Memorex eight-inch floppy drive, from scratch, finding the index point bit going past and trying to read the patterns and write the patterns. And it was bit serial off the disk. And then, you could hear it seek. Clunk, clunk, clunk. Stepping motor. So there was a fair amount of feedback of whether the electronics were doing what I intended them to do. But it was all hard-wired boards, so weren't any programs. And in the end, I don't think much came of it, and I moved on.

Spicer: It reminds me a little bit about the integrated Woz machine, as Wozniak calls his disk controller that replaced a lot of ... like, 60 TTL chips with a couple of PALs and ...

Sites: Yeah.

Spicer: ...and three TTL chips.

Sites: Yeah.

Spicer: And he did it by clever software, I think. It's a very rudimentary software.

Sites: And he was one of the real wizards at balancing hardware and software and getting big efficiencies out of both.

Spicer: Mm-hmm.

Sites: Yeah.

Spicer: Yeah. I mean, to make a disk controller with five chips is pretty good.

Sites: Yeah. So ... but I learned some from Bob. He explained to me walking over to lunch one day ... five of us or so walking over. He started talking about [unknown], the inventor of the four-valued bit, which puzzled me. And he went on to explain that in the 50 ... in the B5000, there was a 48-bit word and then a bit ... a tag big that was used to say this is an address, [the] beginning of capabilities. But it was also used to mark the end of a list. And so, it turned out you couldn't have a list of addresses. You needed four values to say end of list and an address in the single bit. So Bob was sort of down on that pseudo invention, in his sarcastic way. Yes. But sometimes, you learn little tidbits by examples of things. [I] remember, John Cocke once described somebody as like the two-by-two matrix, one-one-one-zero. He said that's not useful for anything, but it's a great source of counter-examples.

<laughter>

Spicer: Could you tell us a bit about John Cocke and what he meant to you as a designer and what it was like to work with him?

Sites: Sure. So I worked for John the summer of '72 ... '71, [actually 1970] I think, at the IBM lab in Los Gatos, which no longer exists. It was a weird little out of the way advanced system and division lab. John had gone there after ... I think he'd gone back to Yorktown after ACS was canceled, or after it changed to ACS 360. And then, from his point of view, it was effectively canceled. And then, he went to the Los Gatos lab and I got recommended by a friend to go work with John the summer after I worked for Fran. And he wanted me to write some programs to split nodes in flow graphs as part of an optimization to take pieces of flow graph that were badly structured loops that had more than one entry and duplicate some of the nodes of code logically so that the loop became a single-entry. And there were pieces of the first iteration from other ways of getting in that then all went to one place. And then, you could optimize the remaining loop, which was written up in his class notes with Jack Schwartz. But it took me a while to learn that John had a whole lot of ideas, some of which were not very good, some of which were wonderful, some of which were quite speculative and he had no idea whether they were good or bad. And so, he just bounced out lots of ideas and part of my job was to actually sorting ... trying to implement a few and seeing if they made sense. And I went in assuming that this was all perfectly thought out. <laughs> And I realized partway through the summer, it wasn't all perfectly thought out.

But I had a great time working for John. He's a unique individual, a really kind gentleman, Southern tradition gentleman. At the time, he was still a chain-smoker. He had lots of ideas. I never heard him angry. I'm sure other people have, but I didn't. And I put together this FORTRAN code to manipulate essentially bit masks of nodes and split them and stuff. And it took me all summer or something that any decent programmer could do in about a day now. <laughs> And then, he had me go back and give a talk on it at Yorktown and people asked questions about it. But ... and I got introduced to more people at Yorktown. So it was again a mentorship. The project in Los Gatos was ... the actual project going on was an unpublicized RISC machine that was a predecessor to the IBM 801. It was called Service-Free System and it was John's design, and the stuff I was doing with flow graphs was sort of a side amusement for him, something he was very much interested in for optimizing compilers, but it wasn't the thing he was working on full-time. And the service-free system, I don't know much about it because I didn't work with any of the hardware. But I think it was an ECL machine, 16-bit wide, and a whole lot like the 801 was a couple years later. And it was called service-free system because the way it had been sold to IBM was this would be a one-board computer that could run native code that emulated an IBM System/360 Model 75 at speed.

Spicer: Wow.

Sites: And it would be service-free in the sense that if anything broke, you just did a board swap. This is when the Model 75 would fill a room and had thousands and thousands and thousands of little boards. And it was going to be fast enough. It had ... I'm trying to remember right. A 50-nanosecond cycle time. And so, you could lots of essentially microcode operations and emulate all of the pieces of the System/360 architecture at speed, at the speed of a Model 75. It also had, at the time, innovative and, I assume, since it was IBM, highly patented I/O connection that was a serial line, 32-bit words, with the line-carried data and clock on just one line. And the clock pulse or edge was missing every thirty-second thing so that peripheral devices could get in sync with the word frame. And I think this was a very early form of the kind of thing we see all over the place now in serial ATA and USB and things. And again, it

was, at the time, very high-speed serial ... bit serial, 50-nanosecond or something per bit. But John was thinking ahead about how to get rid of these big connectors and stuff if you have only a one-board machine. You don't want a connector that's twice the size of a three-and-a-half-inch hard drive.

Spicer: Right. IBM is certainly known for its big cables and connectors.

Sites: Yeah.

So after that summer, I just worked for him for three months, I became one of the people that John would have an idea and he'd call. And John was always having ideas, need to just sort of pick up the phone. So he'd pick up the phone. "Hey, Dick. I have this idea." And he'd walk me through it. Except it was two in the morning in California and he was in New York, <laughs> where it was five in the morning. <laughs> And as best I can recall, we would have a very rational conversation. Then, I would go back to sleep and in the morning, I would have no idea what it was about.

Spicer: Right.

Sites: Just that John had called.

<laughter>

Spicer: That's funny. Did the serial I/O system you were just describing result in a product or ...

Sites: No. None of this turned into products. Everything I ever worked on at IBM was canceled. It got to the point, about my senior year or so, I guess when I started working for Fran, I pointed out to her that everything I'd worked on was canceled, and maybe I was just the kiss of death.

Spicer: <laughs>

Sites: This was after the phone call where the ACS had been canceled, and she said, "Come anyway." <laughs>

Spicer: Right.

Sites: And sure enough, the three-headed compiler was canceled the following September.

Spicer: Now, is there anything you want to say about your teaching at UCSD? That's ... I think that's where we're at right now.

Sites: Yeah. So after hitchhiking for seven months, I then applied to teach at USCD ... I'm sorry. And then, after working at Burroughs, when it became clear I had a limited future there, I applied to teach at UC San Diego, and they accepted me as an assistant professor. I had actually taught some seminar or something there, I forget the details, on my own time, the semester before. So they knew who I was. And

so, they hired me at $13,000 a year, at a time when the bus drivers in San Diego had just received a new contract at $21,000 a year. <laughs> So it was financially a step down from Burroughs, but I wanted to teach about computer architecture to a bunch of good grad students. So I went there and spent four years. I did not write many papers. And so, it became clear I wasn't really on the standard tenure track and was unlikely to get promoted to associate professor, tenured. But I had a bunch of great grad students. It was the time of UCSD Pascal, and [Kenneth] Bowles was running that group. And so, I had many of his students in my classes. I never worked personally on UCSD Pascal, but I used it in the programming classes we were teaching, first on Apple IIs and then on IBM PCs, which were brand new. Because, of course, it was mostly an interpretive system and they just ported the eight kilobytes of interpreter over to the PC and it was up and running.

Spicer: So since you teach programming. Do you have a view on what is your favorite programming language to use ... as a teaching tool, what would you say that is?

Sites: Yeah. Let me come back to that in just a second. I made a mistake.

Spicer: Okay.

Sites: The programming class with UCSD Pascal initially was on these little machines called Teraks built out of LSI-11s. The Apple II port came much later. I got the times wrong. So it was Teraks and then the IBM PC. So programming languages, I'm not a fan of a lot of fancy programming languages. I ... my programming ... I'm a mediocre programmer, like I'm a mediocre student. My grounding in programming is really from Knuth's Fundamental Algorithms course, where the focus is on doing things quickly and efficiently. And so, I'm really happy with C as a programming language in order to write down, "here's what I want the machine to be doing and I don't want it to be doing anything else." I use C++ for the syntax for comments, but I don't use it for lots of indirect method ... virtual method calls and things because the picture in my head is always there's this extra code going on. And there's these constructors running in and destructors running, and I don't want to spend the time on that if it's not necessary.

Spicer: Right.

Sites: As far as teaching people programming, I don't have a strong opinion. Pascal was a good language for teaching people programming because it was consistent and coherent and relatively small. And so, people could pick it up easily. My only opinion today on programming is I think every undergraduate should be taught enough to build their own webpage, to program in HTML, and if they feel like being fancy, a little bit of JavaScript. But to just ... the nice thing about hand typing HTML and having a browser open that file is you get to see immediately what it is. And for someone who's going to med school or whatever next, just knowing what's available and how easy it is to get some text and to get things to reflow and to ... maybe you want to add a picture. You spend a little while learning about image tags, whatever, and then, you can put pictures in. That's a piece of cultural underpinning, which I think nearly everyone in the world is going to need going forward. And having a little bit of familiarity with how the insides work is enough. And building your own webpage is the right kind of motivation. You get to see yourself.

Spicer: Yeah. I'm just guessing that you're not a big fan of ... well, let me get your opinion on visual programming, for example, where you drag icons ...

Sites: Drag and drop lots of icons.

Spicer: The icons embed, like, 10,000 lines of code, for example, even ... you don't get to see inside, really.

Sites: Yeah. I think techniques like that where you can piece together prepackaged code, they certainly have their place. IBM used to long ago have this product called Report Writer [Report Program Generator, RPG] that you would ... you had a very constrained structure, but you get to say pretty simply what you wanted the report to look like and how the calculations were done. And there was the ... a built-in loop and stuff. And I think many of today's graphic programming things are long-later derivatives of that kind of thing that we're ... there's a whole lot of structure and there's some stuff you can do. And within that structure, if that's what you want to do, it can be almost error-free. Bill McKeeman at one point pointed out to me, when he was working on building software for a bank ATM machine, what they built was a little language of describing what you could do in a transaction, and that was all you could do. And then, they built an interpreter for that language. And in the language, things like moving money from one account to another was atomic. The implementation made sure that that was the case. And then, you could write whatever you wanted to in this little language and it would be error-free.

Spicer: Right.

Sites: It would not lose any pennies.

Spicer: Perfect.

Sites: And that was a strong lesson I took, and just inventing little languages to handle exactly what needs to be handled with no further complication, I think still has a big place.

Spicer: That's really interesting.

Sites: When I write output graphics for some of the work I do in studying CPU performance, I tend ... I used to write in PostScript, in handwritten PostScript, but I would create little routines. I would create a little language and then, I would write all of the other stuff or have a program write all of the other stuff in this little language of PostScript function calls, essentially. And once you get the underpinnings done, then slapping together stuff that you can see on screen or on paper gets real straight-forward.

Spicer: Right. Well, that's really cool. So these little ... what do you call these little ... like, pre-processors?

Sites: McKeeman called them Little Languages.

Spicer: Little languages. <laughs>

Sites: Bill McKeeman. I think he has a couple of papers on little languages.

Spicer: Oh, right.

Sites: From the '70s.

Spicer: I have to look that up. That sounds good. It sounds like a brilliant way to simplify a programming project and to eliminate errors as much as you can, anyway.

Sites: Yeah. Well, that's the big advantage of being privileged to work with a lot of bright people. You get these little nuggets tossed out and you end up paying attention to some of them that are good for 40, 50 years.

Spicer: Okay. So you've just spent four years at UCSD.

Sites: Right. So I had ...

Spicer: Then, what happens?

Sites: Several master students graduated, and one PhD student graduated.

Spicer: Great.

Sites: Dan Perkins and another one, Jeff Finger transferred to Stanford and then graduated from there.

Spicer: Wow.

Sites: But after four years of teaching students how to design computers, I came to the realization that I needed to do it myself.

Spicer: Okay.

Sites: Teaching people how to design stuff that I'd never done is a common thing for teachers. Those who can't do, teach, but I figured it was time to actually go do it. So I applied to, I don't know, half a dozen different places, including Bell Labs, and DEC, Digital Equipment Corporation, and took the offer from DEC in Hudson, Massachusetts. Again, through friends. The guy who hired me, Craig Mudge, shared an office with me at University of North Carolina. I ended up transferring to Stanford. He stayed in North Carolina and got his PhD there and then went off to Caltech for a bit, and then ended up working at DEC. And this was the era of [Carver] Mead and [Lynn] Conway and students just starting to be able to design their own IC layouts and have them fabbed.

Spicer: Right.

Sites: And so, Craig had gone through that at Caltech when Carver was also at Caltech and got enthusiastic about the upcoming huge shrinks to microprocessors compared to minicomputers, which was the business DEC had been in. So he went to DEC to start a project to turn the VAX computer of all of its multiple refrigerators into a single chip, or a small number of chips. And he hired me to actually be head of the microprogramming group, microcode group, and then he left DEC a few months after that. <laughs>

Spicer: Sorry, what is the chip that you were working on?

Sites: The chip became the VAX 8200.

Spicer: Okay. Oh, is that the ...

Sites: It was a chip set.

Spicer: Or the chip.

Sites: It was a chip set. I'll get to a single chip in a moment. It was a chip that did instruction fetch and execution, a chip that did floating point, a chip that did memory management. And then, there was a gate array that interfaced to a bus and then, the ... I guess it was the Unibus. And then, the main memory was out on the Unibus. And that chip set and microcode, before it shipped as the VAX 8200, became the basis for the MicroVAX single chip. Bob Supnik, basically, went through the instruction execute chip we had, [and] threw away most of the microcode. This is when the MicroVAX subset was defined to get rid of all the decimal arithmetic and make it small enough so that they could treat the microcode into a much smaller amount of ROM and moved onto that chip, a four-entry translation buffer. Maybe eight entries; I forget. I guess it was eight. And some bus interface. And turned into the MicroVAX II chip. So the microcode that's in there is lifted directly from the VAX 8200, which then shipped a year after the MicroVAX.

Spicer: I see.

Sites: But was a multiprocessor. You could plug in up to three processor boards. And when VMS was able to, it would do asymmetric multiprocessing. One of them would be the boot machine and the others could run user code, essentially.

Spicer: One of the interesting things about Bob Supnik, who you bring up his ... he's got a BA in political science. That's his degree.

Sites: I thought it was in history, but it might be political science.

Spicer: How does a guy go from a political science degree to a microprocessor designer? <laughs> I've got to ask him that…

Sites: Bob has lots of interests... he was in the same dorm at MIT when I was. He graduated a year earlier.

Spicer: Oh, really?

Sites: But we had water fights with each other. And so, he was one of the people who influenced me to go work at DEC.

Spicer: Yeah. How interesting.

Sites: My path in this is ... has very much been mediated by lots of friends.

Spicer: Yes. Yeah. You were saying at every step of the way, you had support.

Sites: Yeah.

Spicer: That's great. So tell me more about ...

Sites: So Bob was the one ... who was instrumental in having me become one of the co-architects for the DEC Alpha. So after we did the 8200, I ended up working on some CMOS processor chip designs, none ~~one~~ of which came to market. In particular, there was a project to do chilled CMOS, meaning dunking it in liquid nitrogen, because it would be about twice as fast at low temperatures. And you use the liquid nitrogen to, first, make the circuits fast, second, to pull all the heat away.

Spicer: Like the ETA systems eventually did?

Sites: I saw one of the ETA systems, but I saw the one with the air impingement aimed at each heat sink on top of each gate array chip. So this chilled CMOS was a box about this size of an original ... about twice the size of a MicroVAX II, and half of it, three-quarters of it was refrigerator and the rest was a dewar with a few quarts of liquid nitrogen. And it had to be in a room with oxygen sensors, because if the power failed, the liquid nitrogen would boil out and fill the room with nitrogen and the oxygen level would go down. <laughs> Oops. And it went ...

Spicer: <laughs>

Sites: ...<makes noise>. It was the refrigerator.

Spicer: Oh, the pump. Yeah.

Sites: This never came to market, but it never came to market for an important reason. Dan Dobberpuhl sat down and looked at the next 10 years of trends in CMOS transistor behavior and realized that the very low temperature speed gain was going to diminish, was going to get less and less of a gain in speed. And so, it was only a short-term strategy.

Spicer: Yes. Right. Yeah. And I think Seymour Cray learned that lesson too. I'd love to get your thoughts on that. You know, switching from gallium arsenide to ... or ECL to CMOS, and maybe he waited a bit too long. I don't know on that score.

Sites: Yeah. So when I was teaching in San Diego, I, of course, had summer jobs, because at $13,000 ... they assumed that they would be paying only for nine months and that every faculty member would cover the other three months. So I spent one of those summers working at Los Alamos labs for Forest Baskett, who had been teaching at Stanford when I was there. And so, he knew me. And I forget how ... I guess I'd heard that he had gone to Los Alamos. He'd taken a year off from Stanford or something. And so, I wrote to him and said do you mind? I'd be interested to learn more about the Cray-1. And so, he wrote back and said ... I think at that point, we were up to e-mail. And he wrote back ... e-mailed back and said, "Immediately apply to get a security clearance because it will take months." And so, I did. I eventually got a security clearance, but it was at the end of that summer. I spent the entire summer in Los Alamos sitting with some of other summer students who were not cleared in a room outside the fence. And they would take us in once or twice to see the machine room and to touch the Cray-1. But when they did that, all of the tape drives were covered with black velvet covers and the flashing red lights in the entire machine room were all on saying there's an uncleared visitor here.

Spicer: Right.

Sites: <laughs>

Spicer: Yeah. I've seen that set up at NASA here.

Sites: Yeah.

Spicer: Same thing with the uncleared visitor <laughs> warning.

Sites: So ...

Spicer: Wow.

Sites: Forest had ... Forest was leading a group that was building an almost-Unix operating system for the Cray-1. And I went there to work on optimizing the machine code.

Spicer: Was that UNICOS? No.

Sites: DEMOS. D-E-M-O-S.

Spicer: Oh, DEMOS, okay.

Sites: Which eventually, I understand, was not well liked at Los Alamos because it wasn't really Unix, and it died, I think, somewhat after Forrest left. But I went there to specifically work on an optimizer for

machine language code for the Cray-1, so this involved reading the whole Cray-1 manual, which had wonderful, good timing charts of how many cycles each instruction took before it wrote an answer into the registers and the conflict matrix for never writing two answers into the registers in the same cycle. And so, sometimes it's very much like the FAA for flights today. Sometimes an instruction would not issue because the slot it needed to write its result was not available because of some shorter or longer instruction that it already issued.

So, I wrote a timing simulator from the manual, laying out essentially a big bit vector of which units were busy in which cycle, and then re-arranged the instructions in an assembly language listing to issue more instructions sooner, and drew little output diagrams of capital I's for instruction issue, and x's for execution, and dots for wasted cycles, and lined them all up, wrapping them 20 characters across and then wrapping around so you could see the timing diagram that was being used, and also so I can show "Here's the original code, and here's the timing diagram that takes 27 cycles, and here's the re-arranged code in a basic block. And here's the timing diagram that takes 21 cycles, and that turned out to be a back-end optimizer that they used for a little bit. It made occasional mistakes in the vector code of breaking a chaining slot, where you had a 64-element vector instruction whose result was getting on-the-fly moved into a second vector instruction. If you started the second one at just the right time, they would completely overlap except for the first few cycles, and occasionally my optimizer would end up moving the second vector instruction one cycle too late, and so the Fortran programmers didn't use it after they discovered that I sometimes made their program slower instead of faster <laughs>. And then, of course, the Cray Fortran compiler got better at instruction scheduling, and then the stuff I wrote was no longer needed.

Spicer: So, I'm just curious, because you were outside the walls, and obviously the machine is inside the walls, so how did ...

Sites: We had terminals.

Spicer: Oh, okay. So, you were doing your simulator by terminal?

Sites: Yeah. I was writing in ...

Spicer: Or, sorry, your optimizer?

Sites: I was writing in Pascal, running on the Cray-1. There must have been an intermediate machine. I think we were using something like a DEC PDP-11/45 running Unix as the base machine, and then we could ship a program to run on the Cray-1, and then ship it back.

Spicer: Now, I've never used a Cray-1. Is it a time ... Can you use timesharing? Or is it batch mode only?

Sites: It's really a batch machine, and a RISC machine. Wonderful, small instructions. But at least it had interrupts, which the [Control Data CDC] 6600 did not <laughs>, initially.

Spicer: Huh. Interesting.

Sites: Well, Seymour didn't know a lot about operating systems. He, like me with C programs and with Knuth's Fundamental Algorithms, he was really focused on "I want real work done every cycle, and this operating system stuff is not real work. It gets in the way." But at least the Cray-1 had an operating system that had interrupts, so ...

Spicer: Well, he wrote <laughs> ... Didn't he, himself, write primitive OS then?

Sites: I think so.

Spicer: COS, maybe -- the Cray Operating System?

Sites: Yeah, the Cray Operating System.

Spicer: I think so. Wow, that's neat. So, your Los Alamos experience was a good one, do you think?

Sites: Oh, yeah. Yeah, that was a really good summer, and again, picking up little tidbits from some of the masters of the world. I got to see one of the real IBM Photostores while I was there. [IBM 1360 Photostore].

Spicer: Oh, now that's quite rare, to see that in operation.

Sites: And I have ... I think I still have at home one of the photo store cartridges, which the museum will eventually get. You already have some.

Spicer: Yeah, we do.

Sites: I did meet here the guy who designed the photostore.

Spicer: Really?

Sites: Yeah.

Spicer: Who was that? Do you know?

Sites: I forget his name, but he was in this building when it was open display. He was wandering around, probably at some Fellows' meeting and I ...

Spicer: Wow. Because I know the guy in charge of it at Livermore [National Laboratory] was a guy called Jim Dimmick, and he lived with the machine. He kept it running, kept it full of fluids and developer and all that other stuff... It's really a funky multi-modal technology machine, isn't it?

Sites: Yeah. This is a machine that had little ... maybe two-inch by one-inch strips of film, and it would write a horizontal line with little blips above and below for ones and zeros, and it could store some huge amount of data in a small physical volume for the time, and these little pieces of film were in cartridges that were smaller than a cellphone, and then there was all this mechanical stuff that would move cartridges around and pull things out, and the developing was done inside the machine.

Spicer: Right, and it also had a high vacuum system that used x-rays, I believe, for the writing, I think?

Sites: Yeah, I don't know what the writing mechanism was, but ...

Spicer: I'm not 100 percent sure, but ...

Sites: With a magnifying glass you could see the bits.

Spicer: And it looks like a pulse train, doesn't it?

Sites: Yeah. Yeah.

Spicer: It's literally ...

Sites: And ... I don't know. I picked up a cartridge while I was at Los Alamos, and I have no idea whether it has classified information on it.

Spicer: Oh, my gosh <laughs>.

Sites: But I assume that because it was available as a souvenir that somebody had said it was okay.

Spicer: Oh, it's fine. Yeah. Yeah, that was a trillion-bit store when it came out. That's unbelievable. Yeah. I think they developed it for the CIA, actually.

Sites: A trillion bits?

Spicer: Yeah.

Sites: So, that's like ... a hundred gigabyte drive, which today is 50 bucks.

Spicer: You can't still buy them that small. Yeah.

Sites: Next year it'll be a compact flash disc.

Spicer: Exactly, yes. Yeah, the prices are unreal. Hard drives still are cheaper, but anyway ... Well, what did you do after Los Alamos? I think you went to Adobe, is that right?

Sites: Let's see. I taught for four years at UC San Diego, and then I went to DEC and worked on the VAX 8200 microcode and the microVAX was derived from that, and then the CMOS stuff. And then, DEC was starting to get concerned about the VAX architecture falling behind all these RISC machines. MIPS was out in the market then. Sun's SPARC was out in the market, and VAX implementations were getting to be slow, and so there was a lot of effort and task forces and things within DEC to figure out what to do to be competitive. The VAX 8600 was this huge ECL machine that barely worked, and eventually the clock slowed down from the designed 67 nanoseconds to 80, and these little RISC processors are starting to run circles around those.

So, there was discussion ... I was part of the groups discussing it ... of trying to do a further subset of the VAX architecture in order to be able to implement it more quickly, to do a totally different RISC machine. On the side, DEC had contracted with MIPS to build some work stations with MIPS chips and discovered in the process that at that time the MIPS compilers were student written and not actually robust for existing DEC customer code <laughs>, so there was a lesson going on over there. But we were looking at what to do about the VAX, and we looked at building hybrid machines that had a board full of VAX execution and a board full of RISC execution, and eventually decided to bite the bullet and say that it was time for a new architecture. So, it's coming back to the comment of compatibility.

But we were hesitant to do that and abandon what at that time was a large customer base, and it was clear that the customers could recompile for a new architecture, and it was clear from the MIPS experience that DEC would put enough money into compilers that they would be robust enough. This was looking forward about four years. But it was also clear from the MIPS 1 experience, where there were no pipeline interlocks ... that was the reason for the acronym ... that if you had a later implementation that no longer had a one-cycle branch delay slot but a two-cycle one you had to recompile, or it wouldn't execute correctly. The lesson from that was people didn't really want to have to recompile for every machine and didn't want to have to have different binaries if they had five different machines; they wanted to have one binary that ran on them all. So, in looking at moving to a RISC architecture, we had to at the same time look at moving a customer base's software, and there were some handwaves about translating VAX binary code into what became Alpha. At the time, it was called R-VAX for RISCy VAX ... Risky in both senses.

<laughter>

Sites: And partly as a cover name with the VAX in it, that if anyone picked up the cover name it wouldn't be obvious that it was not a VAX at all. It later became E-VAX for "extended VAX." Anyway, so, there was this handwave about rewriting binaries to be in a totally different architecture, and I thought that was bogus, so I took a month -- I gave myself a month, and I said, "I'm going to go write a proof of concept translator in order to find out why this won't work." So, I wrote this Pascal program that would read in a VAX binary and go find the basic blocks, and separate the instructions from the data, and then rewrite the instructions in some other form. I think the other form was the same instructions with some minor change. And at the end of the month, I thought, "This could actually work." It was ... Most of the time, you could find all the instructions and you could distinguish instructions from data and rewrite just the instructions and get an executable that ran.

So, I became head of the binary translation group to write such a translator for real VAX code to the eventual target architecture, and at the same time, largely I think through Bob Supnik's influence, Rich Witek and I became the co-architects for whatever this new RISC architecture would be. And the two of us were wonderfully blessed with a company that left us alone. The two of us were chartered to go design the architecture and ignore everyone else, and due to our personalities we were particularly good at that <laughs>. And the company let us make big mistakes and didn't have some committee change things, so the strength of that architecture I think was ... It was designed by exactly two people, and the weakness was it was designed by exactly two people and there was other inputs that it would have been useful to have, or to not ignore. But it was just us. And we argued every day for a year and a half. Rich has ... Have you interviewed Rich?

Spicer: No.

Sites: Oh, you should. Rich had this deep background of building chips, and I had more of an academic background.

Spicer: How do you spell his last name?

Sites: Witek? W-I-T-E-K.

Spicer: W-I-T ...

Sites: E-K.

Spicer: E-K. Wit-tek. Okay, thank you.

Sites: He works at Apple here, but he lives in Seattle. He and [Dan] Dobberpuhl did the StrongARM chip. He and Dobberpuhl also did several internal DEC RISC chips that never came to market, but they were very useful experiments that let the ... Much like the IBM Stretch helped the 360 architecture by the Stretch having lots of bells and whistles that they got to throw away, but a few good things they got to carry over. The internal DEC RISC chips that actually got fabbed in unit quantities trained a few people, including Dobberpuhl and Witek, about how to build a real RISC machine that would last. So, Rich and I had lots of divergent ideas about what should be in this thing, so we argued a lot, but in the end we agreed that we would leave out anything that wasn't supported by both of us, and that was a huge win because if you put in something that's actually not a good idea, you have to live with it forever. If you leave it out and you figure out two years later what you should have done, you can put it in later. You can add. But it's not an industry where you can subtract very easily unless you make one of these major changes of throwing everything away and starting over.

Spicer: Right. I think one of David Patterson's points of drivers behind Berkeley RISC was, in fact, they used the VAX as a canonical case of the CISC architecture. Is that true?

Sites: Yeah.

Spicer: Yeah. So, the instruction set of the VAX was considered very complicated, I think, with over 700 instructions. Is that right?

Sites: Well, there were a lot of instructions, and there were lots of addressing modes, and so there were lots of combinations. But there were some six-operand instructions that would do stuff on locked, linked lists and things.

Spicer: Oh, yeah, it's very sophisticated.

Sites: And a whole lot of decimal arithmetic instructions that were lifted from the 360 architecture, and lifted badly, in the sense that they failed to account for the 360 as a big-endian address machine, and the VAX is a little [-endian] one, and PDP-11 was a little-endian. And so, the bytes of the decimal fields were in big-endian order, and the entire rest of the VAX was in little-endian order <laughs>, and so finding the sign nibble was always a pain because it was at the wrong end, and you were pretty much forced to load decimal stuff a byte at a time because the four-byte load gave you the bytes in the wrong order. Anyways, there's a bunch of that.

Spicer: Right. Wow. That's interesting. Well, tell us more about Alpha and what it meant to the company, for example, technically and commercially.

Sites: Sure. So, I think we started on what turned out to be named Alpha design in 1988, and its basic outline we set quite quickly, just in the first month, that it would be a RISC machine. It would be a 64-bit RISC machine ... this was at a time when everything was 32 bits, except for something really large like the Cray-1 ... and that it would be load-store registers, and we decided on 32 registers because for VAX, those 16 registers were almost always enough but occasionally not, and we figured with faster programs and better compilers it might be useful to have a bit more, a few more registers. And we decided to have the floating-point registers and, in fact, instruction set somewhat separate on the off chance that there would be an implementation built in gallium arsenide ... this was at a time when Cray was working on the Cray-3 in gallium arsenide ... which would not have the chip density to put integer and floating point on a single chip. So, we designed the architecture so that the floating-point loads were separate: They didn't share registers, and they could be reasonably implemented in a second chip, somewhat like the Intel 287, and along with the 286 for the 287 doing floating point before the 486, putting them both on a single chip.

Then we went through a lot of details over things like, do a 64-bit floating point or 80-bit extended precision, which is what Intel had at the time. Rich, fortunately, argued me out of trying to do 80-bit stuff, because it would have been just a big wart on the loads and ~~in~~ the stores and everything was the wrong size. And we did an internal document of the entire architecture about a year later and had a big review, had like 30 people and two days and review every piece of it, and there were a number of things we struggled with.

Our goal was to design an architecture that gave implementers the freedom to make it really fast. We set ourselves a 25-year time horizon saying, "We want this architecture to be able to last 25 years," and then we worked backwards and said, "What are the consequences of that?" We said, "Well, over 25 years, it

needs to end up with implementations that are about a thousand times faster if we're going to stay on the industry curves leading up to 1988." We said, "So, how are we going to get a thousand times faster?" The initial implementation was aimed at a 200-megahertz, five-nanosecond cycle, and we didn't expect that clock's rates would go up by a factor of a thousand <laughs>. So, we decided that a reasonable approach would be for implementers to pick up a factor of ten in clock rate, going from 200 MHz to 2 GHz. It would be reasonable to pick up about a factor of 10 by having more and more processor chips. We weren't quite thinking yet of multiple cores on the same chip, but to have maybe as many as 10 processor chips. It looked like two to four would be perfectly doable. Ten might be a stretch, but over 25 years that might be okay, and the remaining factor of ten would need to come from issuing multiple instructions per cycle. We actually thought we could probably get up to a factor of eight, but not ten, that maybe we could get a factor of twelve on clock speed or something. So, order of magnitude, but it was just ... No one of those techniques was likely to be enough over 25 years, so we designed to all three of them.

We said, "So, if we want the clock rate to scale up by a factor of 10, we want to be sure and not put in some kind of instruction that necessarily has lots of gate delays." We wanted ... If there's going to be multiple issue, we wanted instructions that you could look at, hardware could look at a group of two or three or four or eight instructions and decide which ones could be issued and make that decision really quickly, because you have to do that every cycle, and to make all the decisions about register conflicts and bus conflicts and execution unit conflicts, to make all those really quickly. So, that dictated that almost all of the conflicts could be decided by the initial six-bit op code and the register numbers and nothing else. And that led us to trying to have almost no other state except for the registers. So, no funny little bits over here. No condition code bits, for example, no floating point global state bits, rounding modes or whatever we wanted. Everything buried in the instructions themselves and in the registers to make instruction decode fast, because we had seen that the [CDC] 6600 had this fairly massive problem with scoreboarding. It was, I think, a stretch for Seymour to get the scoreboarding on the ‘6600 to work at all, and he somewhat simplified [things] in the Cray-1 so that the scoreboarding would be easier.

We saw -- internally, there was a follow on to the DEC KL10 processor, Super Foonly, which died. It was also called Jupiter. It died because they got to the point where they could execute instructions a lot faster, but they couldn't issue them quickly enough to keep the execution unit busy. So, Gordon [Bell, DEC VP of Engineering] canceled it. So, we had seen this history a little bit and paid attention to it, about instruction issues. So, we designed the architecture to make multiple instruction issue as easy as we knew how for implementers. And then we had to deal with multiple processors, and how do you handle shared variables where you need the effective locking a variable and changing it back to the original IBM [360] test-and-set instruction where you get to hold the old value, change the value, and put it back before any other processor gets to look at the old value. Only then can you build real multiprocessor code that's safe or doesn't have race hazards.

So, we said, "How do you build locking in a load-store RISC architecture?" Holding a lock-bit on the bus between a load instruction and some subsequent ... who knows when ... store instruction won't work, and certainly won't scale to lots of processors. So, we simply lifted the MIPS load locked -- I forget the MIPS name, so the Alpha name as we ended up with a load-lock and store conditional to load something and within the processor remember its address, update it with whatever instructions you needed. And then

when you went to store it, to do the store conditionally, based on whether any other processor had changed that location whose address you held and did bus watching for changes, and if no one else had changed it, then do the store. Otherwise, don't do the store; return from the store instruction a bit that the code branches on and redoes the load and redoes the update. And after some small number of iterations, you're successful at doing the atomic update, and that was then the underpinning for all of the things that we now think of as atomic ... or test-and-set, compare, and swap. Those are all implemented in Alpha with little short sequences of load locked followed by store conditional. And then the last thing for the longevity is the 64-bit address. It was clear the 32 [-bit] addresses would run out, and we didn't want to get partway through a 25-year program and tell everybody to change from 32-bit compiled code to 64-bit compiled code.

Spicer: As a product for DEC, how did it do? What did DEC do with this new technology?

Sites: Well, the Alpha came to market at a time when DEC was dying. Microprocessors, Intel PCs, IBM PCs had done the traditional Innovator's Dilemma thing of starting at the bottom and working their way up, and they were working their way up into workstations. And DEC really didn't know how to build anything that cost less than $50,000, so they had some PCs, but they were expensive to build, and expensively priced, and they didn't have really any management bench strength that knew how to deal with lots of small, cheap things. So, Alpha was brought to market as a high-end machine, and the big splash was from getting the Oracle database code to run on an Alpha that had lots and lots of RAM because we had a big address space, and so, you could be 300 times faster doing database stuff in RAM. It didn't talk too much about the RAM price, but much, much faster than beating discs to death, and that got people interested. So, I think it sold well for a while.

At DEC, we were never allowed to see what the actual sales numbers or costs or anything were, which I always thought was a mistake. I thought engineers would be better engineers if they knew what they had done in a previous project that was too expensive. So, I don't have a good answer to how well it did. It did okay for a while, and then ... The year that Alpha came out, Ken Olsen retired, and Bob Palmer took over, and he took over a company with 126,000 employees and an income that could at best support half that. So he, on good advice, immediately upon taking office floated something like a $200 million bond issue on the assumption that six months down the road it would be impossible to bring a bond issue to market <laughs>, and then spent many months laying off half the company, because the company had just grown unreasonably large with lots of not-profit-making things. The VAX people always said that they made one-and-a-half times the company's income, and then the other products spent the rest <laughs> to get the income back down. So, let's see. When we came to market, there were two operating system developments for Alpha: There was the VAX/VMS development, and the Unix development. Ray Lanza ran the Unix group, and I forget who was nominally in charge of the VMS development, but ...

Spicer: Not Dave Cutler?

Sites: No. He was at Microsoft by then. It was about that time that there was the settlement between Microsoft and DEC.

Spicer: Hudson Alpha stuff, maybe?

Sites: There was some dispute about whether early Windows NT code bore too strong a resemblance to VMS code in terms of the data structures being identical but the names were different kinds of thing.

Spicer: Oh, wow.

Sites: And so, there was some settlement over that, which among other things included that Microsoft would port Windows to Alpha. So, in that sense, there were three operating system developments, but I don't know much about the Windows one. So, in order to do continuing VAX development and this new Alpha development, the VMS group split into two. One group of people whose job was VAX/VMS, and one group of people whose job was Alpha VMS, and that was the right thing to do because if you don't split the group, then you have people who are doing a little bit of this, a little bit of that for two different architectures, and you end up spinning your wheels and things take much too long, and people don't know how to get performance reviews if you're doing two different things.

So, the VAX/VMS group had a clear charter: Keep VMS running on VAXes in the marketplace as competitively as possible as long as possible. And the [VAX -- I misspoke] VMS Alpha group had a different charter of: Get this thing up and running and out in the marketplace, and drive VAXes out of the marketplace by being that much better, because otherwise somebody else will. So, very different goals and very different measurements of how successful they were. The thing that pleased me was, as I watched this split, the good people went to Alpha VMS. That's when I thought the whole thing might actually make it, and that turned out to be true. But they wrote Alpha VMS in a very odd way. They wrote it in VAX assembly language, and they wrote a translator from VAX assembly language to Alpha assembly language, not at the binary image level, at the assembly language level, and they put stylized comments in the assembly language to help the translator. And that let them take a whole lot of code they already had and move it over fairly quickly, but VMS for Alpha was a 32-bit operating system because VMS's entire design was defined in terms of 32-bit values and pointers, and those were embedded in the definition of the system calls and could not be changed.

Ray Lanza took, for Unix, the opposite tack. Ray said very early on, "Look. If we do a 32-bit port of Unix, this machine will never be used with 64-bit addresses. Both operating systems will be 32-bit only, and nobody will ever change. So, I, Ray, will do a complete 64-bit port of Unix. Otherwise, Alpha can't possibly succeed." So, he did, which meant having to go through all of the Unix sources that knew that int and pointer were the same size and fix them. We also had an 8K page size instead of 4K. ŀ Ray had to go through and fix all the places that knew the pages were 4K and turn them into defined constants and then define that to be 8K. So, it was a very difficult port, and it was industry leading. They had to define what's now an ILP integer-long-and-pointer amongst the choices.

One of the other choices was to make all the integers 64 bits, but that turned out ... It broke more code than making the pointer 64 bits. And they had to build the compilers that would do that, and they had to build the LINT programs that would detect likely struct-overloading unions where somebody unioned an

integer in a pointer and go fix those and make them bigger. Nonetheless, they finished before the VMS port.

But meanwhile, Ken Olsen had decreed that this was his VMS machine, this Alpha, and that it would not run Unix. And I sat down in Palmer's office -- This was before the transition, but Palmer was coming up as one of the likely replacement candidates -- and Ken came by Palmer's office by arrangement, and the three of us sat down, and Ken said, "This is only going to run VMS." I knew perfectly well that Ray was porting Unix, but none of us said so. So, Ray and his group did the entire Unix port under the sheets. It was not a funded project, and there were, I think, a dozen people, and they just ... They were off doing other things. And then when we got the first chip Unix was ready to run because we'd been doing lots of development on simulators, and so Ray came down from New Hampshire with a disc full of bootable Alpha bits, plugged it into the first chip we found that actually could fetch instructions correctly, and turned it on, and it booted. It booted Unix. This is with an in-memory RAM file system. Booted up to the command line. Ray took the disc and went home to New Hampshire because it was starting to snow. So, first boot, first try. VMS at that point was running six months behind, and so the Unix announcement and the product announcement had to wait until VMS was ready <laughs>. So, this must have been winter of 1991 ... November of '91, I guess, was about when the first boot happened with Unix, and then the announcement was in February of '92, and VMS machines were delivered late in the spring of '92.

Spicer: Okay, well, that's a very thorough look at the Alpha. Do you want to move on to ... Or, do you have anything else?

Sites: Yeah, I'd like to take a moment and look back at what Alpha has left in the industry, because it certainly came and went. For a little while, it was the fastest machine in the world. It was in the "Guinness Book of Records" for one year. But meanwhile, the [Intel] x86 architectures keep getting better and faster, and DEC didn't have really a big enough market to sustain. There were some discussions with Apple and IBM and MIPS about combining and everybody using the same architecture, meaning everybody using Alpha, but in the end nothing came of that, and it slowly ... The company closed up and got sold off to Compaq and then HP.

Spicer: Did any of the chip-making knowledge or IP ...

Sites: Well, Intel bought the Hudson [Massachusetts] Fab Line buildings, and those ... The equipment there was four-inch wafers and did not get upgraded to five- or six-inch wafers, and so Intel basically did old kinds of chips on those lines and then wound them down.
So, there's a few things that are stuck in the industry, I think, from Alpha. Among other things, until this past summer, there was about a three-year window when the fastest machine in the world was essentially an Alpha derivative, and that's the Chinese high-end supercomputer, the Sunlight [Sunway TaihuLight]. Their chips have two hundred fifty-six processors, cores, on one chip plus something like four supervisory processors, but the two hundred fifty-six main cores have a 96K-byte cache and 8-kilobyte pages, and if you look up on the web "96K-byte cache, 8-kilobyte page," you get two hits. You get a hit on the Chinese machine and you get a hit on the Alpha 21164, the second chip <laughs>. And if you do the scaling arithmetic, two hundred fifty-six, 21164s from its [original] five hundred nanometer design rule scaled down to twenty-eight nanometers. Only two hundred fifty-six of them fit on our chip. Before that, a couple

years ago, there was a stretch where Alpha implementations were the slowest machines in the world because there's a lot of academic simulation of Alpha to run cache design studies and things from long address traces because it's an easy architecture to simulate because it's so simple <laughs>, so we got both ends of the spectrum on that.

Let's see. There's a few things that the industry picked up on and you now see commonly everywhere, but we were about a year ahead. I put two performance counters on the first Alpha chip to count things like instructions executed, cycles, cache misses, cache accesses. They were tiny. They were 12 and 16 bits, and I architected them so that you couldn't read or write them. All they could do is generate an interrupt on overflow. And I did that so that the actual counters could be dumped over on the edge of the chip somewhere near the pins where there was space, and only a single wire of the thing to count had to go over to the counter, and there were eight choices, and we ran them over ~~[ph?]~~. And so, when the space crunch came and the chip was too big to manufacture by 15 percent or so, lots of stuff got dumped, but the counter stayed, and you need two so you can do ratios, like instructions per cycle or whatever. Those started showing up in other vendors' chips in 1993 a year later, and now there's performance counters in nearly everything. So, there were a handful of things like that that we did, that showed up elsewhere. We had a conditional move instruction to avoid branches that you ... The usual, "If X is less than zero, X equals zero," kind of construct. We just did a conditional move comparing to zero, and if the condition was true it would store its result in the register. Otherwise, it would leave the register untouched. And a year later, there's conditional moves in x86 architecture, things like that.

Spicer: All right. Well, what did you do after Alpha? It looks like you went to Adobe, perhaps? After DEC?

Sites: Yeah. I spent a little time out here in Palo Alto at the DEC Systems Research Center working for Bob Taylor and working with Chuck Thacker and some of the crew, which was a wonderful experience. Bob is the best manager I've ever had. Very, very lowkey and incredibly supportive. He viewed his job as hiring the best people in the world, keeping them happy working with each other, and protecting them from the entire rest of the company, and he had done that at Xerox until they all moved to DEC one day, and he was very good at doing that at DEC. I really enjoyed working for Bob. I spent a little time there looking at tracing, as it turned out, Windows NT running on Alpha -- tracing instruction execution on the fly by having each branch ... by rewriting the binary so that each branch actually did a call to some tracing code that would then return using the PALcode or protected code that were the instructions that mediated all the transitions from user mode to operating system mode. And that was successful and ran at about 1/20th of normal speed, but would actually boot Windows NT, and run programs. It was not so slow that it made TCP/IP break and time out. It was close, but you can do 20 times too slow; you can't do 50 times too slow and have things actually run.

So, I did a bunch of address tracing of what was really going on in execution, and among other things we traced [was] some four-processor Windows NT database TCP benchmark kind of code to find out why with four processors it was only twice as fast instead of four times faster and found some locking bottlenecks that people were unaware of. And I presented on that at Microsoft, and they went and fixed SQL Server. Some of the fixes were that day and rethought about how to keep the locking overhead low, and that was one of many things I did that then show up again as I worked for Adobe and then Google.

So, I've always been interested in this boundary between hardware and software and how do you make things efficient, and I think that really all goes back to Knuth's Fundamental Algorithms course, where we had a specific fake computer and we wrote an assembly language for that and ran on a simulator, and the simulator counted cycles and we get to see how inefficient or efficient things are. So, as DEC was winding down, I went ahead and resigned in 1996 ... and applied, I think, is a little too strong, but talked to friends and looked for jobs at Intel and perhaps Apple and Adobe and ended up getting hired at Adobe. Actually, [I] had an offer from Intel and it was pretty reasonable. I actually got as far as taking the drug test before they would completely accept me, and I would start two weeks later.

Spicer: They have a drug test at Intel?

Sites: Oh, yeah. I found it very offensive, but I did. They lock you in a room and there's only ... There's blue dye in the water, so you can't bring your own sample in with you, and they check the temperature. I suppose that's what you have to do with drug tests, but I thought it was terrible. But I had asked right at the end of that if I could come in and hang out for a day with the people I'd be working with and get to know them a little bit before deciding whether to accept or not, and they said, "No. No, that would be a security breach." So, I asked the same question at Adobe, and Warnock said, "Oh, sure. No problem. Come hang around." So, I hung around at Adobe for a day and liked the people. I knew some of them, and so I accepted Adobe and turned down Intel.

So, I worked at Adobe for seven years. I got hired to work ... I offered to work in an optical character recognition group inside of Acrobat. The work was ... I was officially in the Advanced Technology Group, working with Tom Malloy, and I talked to Dennis [Nicholson], who was head of the OCR group, and I said, "I think I can improve the accuracy of the OCR." It turned out, six months later, no, I couldn't improve the accuracy [of the] OCR, and I was totally off base on how hard it is to make optical character recognition robust, and I didn't actually know anything about it. I just thought it would be a good, interesting hard problem <laughs>. But they hired me, and after six months I went to Tom and said, "I'm failing at this OCR stuff. It's not going to work. I need to stop," and I had no idea how he was going to react ... anywhere from "Oh, gee. We hired you on false pretenses. You're out," or whatever, and he said, "Why don't you work on it a while longer?" which I took as a very positive response, but I said, "No. I had that thought two months ago. I worked on it two more months. It's not working. I'm not adding anything to the robustness."

So, he said, "Okay. What else would you like to do?" which I also took as a positive response rather than "Oh, well. You're not going to do that, you're out." <laughs> So, I ended up working on adding digital signatures to Acrobat documents. This is at a time when digital signatures were just starting to happen, and they were in two forms: One was a bunch of encrypted-bit kinds of signatures, and the other was a picture of somebody's handwriting, essentially, with some crypto underpinning that you couldn't just paste in a picture. And so, Jim Pravetz and I built for Acrobat -- we defined the internal structures and built the code to implement -- both kinds of digital signatures, and I also built a PalmPilot application that you could sign, and it would download to Acrobat and put the signature in the document and had it take the rough digitization ... or rough handwriting as the case may be ... and make smooth curves fitting through all the points and also allow you on the small PalmPilot screen to sign your name across multiple lines, mostly

because our daughter was named Fiona [Louise] Sites-Bowen, and she couldn't get all that on one line. So, Adobe actually has a patent on that now for multiple-line signatures.

Spicer: I use that signature [feature] almost every day when I sign deeds of gift for the museum, actually.

Sites: Well, it's possible some of my code is still there, but I hope not <laughs>.

Spicer: ... the JPEG of my signature. Oh, I'm sure it is. Yeah.

Sites: Oh, no. It all needed to get ... I'm not good at real production code. I'm good at proof of concept, and so if my proof of concept code ships somewhere, my hope is that it'll get replaced by a professional programmer at some point.

Spicer: <laughs> Okay. So ...

Sites ~~Spicer~~: I did that at Adobe. I did a little bit on e-books, encrypted e-books that if you had bought the proper mechanism they would decrypt on the fly and you could read them, and ...

Spicer: Did Adobe have its own format for e-books?

Sites: Yeah, yeah.

Spicer: What was it?

Sites: Oh, it was embedded in Acrobat.

Spicer: Oh, okay.

Sites: And so, we got some vendors to sell books that were PDF files, except you needed to buy the way of opening it. And Jim Pravetz and I built the code for that also, to make sure that you were running on a licensed machine. So, we looked around for the machine MAC address or something, Ethernet address, which turned out to be a very bad idea because at the time you could buy ethernet plug-in cards for ... What was it, Card Bus or something? You could plug cards into laptops that could do things like interface to the ethernet, but each card had a different 48-bit ethernet media ID number, so if you changed ethernet cards all your e-books stopped working. I also made the mistake of checking the machine type but also checking its speed, and this was just as Intel was introducing SpeedStep, and so sometimes you could go read the speed of the processor and it would be a lot lower than the number I had read when I authorized the machine. And then all the e-books would stop working if you were on an airplane and not doing much and it was running slowly, and if you had it plugged into the wall at your desk, it would run at full-speed, and the e-books would work.

Spicer: Tell us what SpeedStep is.

Sites: Oh, it was an early form of slowing down the clock on an Intel processor to save power. It's now throughout the industry, of course.

Spicer: So, this is more for in the mobile space, then?

Sites: It was specifically for laptops to get longer battery life.

Spicer: Okay. So, it's really a throttling ... They say it's ... SpeedStep sounds like they're making it faster, but they're really throttling the processor, right?

Sites: No, they're making it slower.

Spicer: For battery life?

Sites: Yeah.

Spicer: Okay. Got it. So, they're not ...

Sites: This was in the era of the first Pentium.

Spicer: Anything else at Adobe that you want to talk about?

Sites: I really liked the people, and I really liked [Adobe co-founders] John [Warnock] and Chuck {Geschke]. Although it had become a fairly big company, it was very informal and very friendly.

Spicer: Yeah. We got to work with them on our exhibit, and they were the nicest people, really, right from the top on down; everybody is super nice.

Sites: So, I moved into the Photoshop group after a while, and worked on a bunch of the internal underpinnings for pulling metadata out of JPEGs and TIFFs and all sorts of stuff. I did the usual, thorough "Go find the 38 different metadata formats and how they're used and how they're defined." The JPEG standard is in a book form, which is out of print. So, I bought a copy, but the only one I could find ... and this was in the, what, the late '90s, early 2000s? The only one I could find was in Amsterdam, I think. And so, I bought it and had it shipped over.

Spicer: It's not published online?

Sites: It was published at the time, but out of print, so I had one of two copies at Adobe of the actual spec <laughs>. So, I did all the metadata internals for Photoshop and Photoshop Album, which was a low-end consumer product for a while.

Spicer: I remember, one of the things we worked ... One of the questions we asked was, "So, let's say a person comes to Adobe for a job and they say, 'Yeah, I know all about Photoshop.'" And they'll basically

say, "The interview is over," at that point, because Photoshop is such an incredibly protean and involved and complicated piece of software. There's no one in Adobe that knows all about everything about Photoshop, pretty much.

Sites: I think that's true.

Spicer: <laughs> So, that was an interesting thing, how it's become this ecosystem that is beyond any one person's understanding.

Sites: I think at the time I was there, there were still a handful of people who really understood the whole product.

Spicer: There are hundreds of functions, and then the order in which you sequence them is important as well.

Sites: Yeah. No, I think at the time, and Chris Cox knew the entire product, and Chris had written most of the inner-loop code and most of the filters.

Spicer: Okay. And they're on the splash screen, of course, I think, those two names.

Sites: Oh, the list of names. I have to tell you a story about the list of names.

Spicer: Oh, yeah. Sure.

Sites: So, I work on getting some of this metadata stuff extracted so when you say, "Give me the information about this picture," it does it. Also, when you manipulate a picture, I would rewrite the metadata to match. So, you cropped the picture; I had to go rewrite the widths. It turns out, there's 27 different places in these 38 different metadata standards that have the width, including in the embedded thumbnail image and in the TIFF information and in the Associated Press metadata information, which you can find in some images. So, I wrote I think the only code in the world that rewrites all of them correctly. So, this all got together, and we got a new version of Photoshop out. And I'm running it on my ~~the~~ machine and I'm going through the splash screen, and then I wait for all the credits. And the credits come rolling up, and I recognized all the people ... Mary, Queen of the 13th Floor ... and then down at the bottom it says, "And our favorite customer, Dick Sites." So, I go over to Mark Pawliger ~~Ploeger [sp?]~~ who is head of Photoshop at that point, and I said, "Mark, thank you for putting me at the end as your favorite customer," and Mark gets this really, really odd look on his face. He says, "Dick, everyone is our favorite customer. We pulled the name out of the operating system and pasted in."

<laughter>

Spicer: Did they leave that in the shipping version, "Our favorite customer"?

Sites: Oh, yeah. So, if you ran Photoshop with that version down to the end of all the credits, it would say, "Our favorite customer Dag Spicer."

Spicer: That's nice. Oh, you mean ... So, it uses the registered name?

Sites: It uses your registered name from the operating system, so I wasn't their favorite customer at all <laughs>.

Spicer: That's kind of clever, actually <laughs>.

Sites: Yeah. I got totally sucked in.

Spicer: Yeah. I probably would have, too.

<laughter>

Spicer: Although with less reason. Okay, so Adobe, let's wrap that up.

Sites: Yeah. So, I eventually got discouraged at Adobe and resigned. I had become one of the Adobe Fellows.

Spicer: Whoa.

Sites: Largely thanks to Greg Gilley on smoothing some more of my rough edges and getting me promoted. So, I was one of, at the time, six Adobe Fellows, and Adobe was going through a time of cutting back a bit, and also John and Chuck had retired three or four years earlier. And I felt like the company was not being very innovative, and I had no traction in trying to encourage them to have some advanced developments that were visible to consumers, much like at the time IBM's alphaWorks had trial software that people could play with, and then they get to see some things are good, some things aren't. And Adobe was ... The career path was things only go out in the big shipping products every 18 months, and if you have a great idea it needs to go into Acrobat or it needs to go into Photoshop or it needs to go into something else; otherwise, it doesn't go anywhere, and I thought that was stifling innovation. And then they ended up closing a lab of people that I'd worked very closely with on Photoshop Album that was in Minneapolis, and they gave all those people a week to find other jobs in the company, which was essentially impossible because there wasn't headcount, and then fired them and laid them off, and some of them had mortgages and stuff. And I thought that was a terrible way to treat valuable people rather than saying, "The thing you're working on isn't working out. Let's find something more productive." And I raised this viewpoint up through our management chain, up to Bruce Chizen, who was CEO at the time, and got no real response, so I thought about it over Christmas and resigned, said, "I don't want to work for a company that treats good people that way." And then I wrote a page-and-a-half letter to Chizen explaining basically what they had not done over the previous three years that made me feel I didn't want to work there; because I thought they deserved some explanation.

So, I ended up then going to Google. I had contacted friends who had gone from DEC to Google. One of the things I noticed while I was working at Adobe is people that I really respected working at DEC, as DEC declined, had gone off to other companies, and many of them, after a few years, had gone from those other companies to Google. And I started to notice that there was a real cluster of good people I knew who had gone elsewhere and then gone to Google, and when Mike Burrows left Microsoft and went to Google, I said, "I need to go find out more about this because I have a lot of respect for Mike." And so, I went in and interviewed, and they accepted me. Ever since then, when I've done interviews [of applicants] at Google, I've felt like I could never land a job there, but they accepted me, and people complained how the place was much too big with 4,000 employees. It was much better when there were 400 <laughs>. And of course, now, it's way over 40,000. But it was good. So, I worked on ... Well, Google hires people and then finds project matches over the course of a few years. You start with some project and go on. So, I built a bunch of code to trace procedure call and returns in order to see what some of the big software Google, like the web server front-end code, which is this huge, monstrous piece of code, to look at all of the calls and returns and timestamp them and draw pictures of where all the time's going, and in time sequence rather than just a histogram or a profile, and discovered some places that were not nearly as efficient as they could be. I had done a similar tool at Adobe where we found why Acrobat was taking ... Acrobat 4 or something was taking eight extra seconds to start up. It turned out it was a bunch of C++ constructors that were running before the main program, their global constructors. And there was no software tool that showed that execution time, and some of them were going to disc, and so when I traced, when I built a stand-alone tool that just traced calls and returns, all of those popped out immediately. And moving those constructors either to start later or off to a separate thread, we cut the start-up time by nearly eight seconds.

Spicer: Wow. That's amazing.

Sites: And so, I've done this sort of tracing and understanding performance at low levels ever since I took Don's course. So, when I went to Google, I did routine call and returns there and started partly learning a bit about the code base and partly finding some things that could be faster.

Spicer: What is Google like as a place for research and doing innovative things?

Sites: Yeah. I think there are two answers to that. I'm going to talk about my experience there. I think the company is somewhat different now, and so I can't talk about what it is now. But I think its culture has declined somewhat. I think to some extent, I was there during the golden years. So, when I got there, it was a very open culture and very deliberate culture. They deliberately thought out and articulated, "We want people to be innovative. We want to get rid of all the things that get in the way of being innovative." So, everybody, a day a week, 20 percent time, you could do whatever you wanted to a day a week if it had some vague relationship to something that Google might want to do. And at your performance review, you couldn't be dinged for working only 80 percent of the time on your day job, because some places have this only as lip service ... 20 percent time, but it's an extra 20 percent. You can't ...

Spicer: <laughs> Yeah. You're working 120 percent, not 100 percent <laughs>.

Sites: No, this was an honest "Take a day a week," order of magnitude, "and work on something totally out of the blue that you think is interesting." So, they would hire bright people to work on things that the bright person wanted to work on, on the theory that I think is correct, that you're much more productive and much more innovative than if you're working on something that you're told to work on that you don't like. The other thing I really liked getting there is there was no hierarchy, that you just talked to everybody, and they deliberately put you in open offices with lots of other people working on different things, and you learn a little bit about the other people's projects and stuff, and then you move around every nine months or so.

And after a few years, you have friends that you know at some level scattered around the company. And then, when something comes up that you need some statistics help, you go email or call that person or go see them, and there's a huge strength in that, which was missing at Adobe at the time. I think Adobe has improved in that front, but at the time, if you were in the Acrobat group you really weren't encouraged to talk to any other group, and you certainly couldn't share any code because the code bases were all different styles, all different languages, all different conventions for naming variables and for declaring integers. So, you couldn't take a good piece of code out of Photoshop and put it in Acrobat. I tried.

Spicer: Right. It reminds me a bit about IBM. And I'm just guessing here; you worked there, so I'd love your idea on this. But IBM routinely moved people, I think, too ...

Sites: Yeah. I've Been Moved.

Spicer: I've been moved ... Is it every two years or something, or maybe a little more than that? What was their rationale for that? Was it to distribute the competence across the company, basically?

Sites: Yeah.

Spicer: Like Google is doing?

Sites: Yeah, I think so, yeah. And Adobe, at the time, was resistant to that idea. They really wanted people ... Oh, the people who shipped Acrobat 4 have the right expertise to build and ship Acrobat 5, and so the career paths were very limited, okay? At Google, you could work on anything, and if you had some success at it in your short time, you could attract some more people to work on it or even find a group that wanted to do that that had funding. But everyone was paid anyway, and so there wasn't a lot of headcount or budgeting issues that were at all visible.

Spicer: Well, it's interesting that Adobe has basically very mature products, a very mature product line, and that their changes are essentially incremental. They're not going to take any risks, I don't think, but maybe they should.

Sites: Yeah, I think they are more, now, but at the time I worked at Adobe, they were shipping boxes of software to Fry's or whatever.

Spicer: Yes. Right.

Sites: Okay? Google's business is very different. It's ... every week, there's a new version of the search code ...

Spicer: Yes.

Sites: ...running on a million servers.

Spicer: Right.

Sites: And there's little changes all the time.

Spicer: Adobe does that, too, with ...

Sites: Adobe is now doing ...

Spicer: They're a subscription model.

Sites: Yeah. I was there before their subscription model. I left before the subscription model.

Spicer: Yeah. So they're pushing the changes out all the time.

Sites: And so ... pushing things that only go into boxes, as a major change, every 18 months, so a very different business than changing every week and very different development culture ~~call~~.

Spicer: Yes.

Sites: But I really appreciated that Google invested heavily in the support that would make people innovative. So for example, there was one code base. There's probably now two with Android a bit cut off, but at the time, there was just one code base. So when Jeff and Sanjay built the indexing code to ...

Spicer: Just tell me their last names.

Sites: Jeff Dean and Sanjay [Ghemawat] … I always forget Sanjay's last name.

Spicer: Okay. Thank you. .

Sites: The two distinguished engineers who actually are responsible for half of Google's design.

Spicer: Really?

Sites: Yeah.

Spicer: Oh my gosh. It comes down to just a handful.

Sites: It's just those two.

Spicer: Wow.

Sites: And Sergey and Larry. But anyway. So Jeff and Sanjay built source code indexing. And immediately, the entire source pool was searchable, as opposed to building it for this database and this database and this database across 30 different projects. So Google has a strong culture of sharing code, and you can find the code to share. And without the finding ...

Spicer: Within the company?

Sites: Yeah, within the company.

Spicer: Yeah.

Sites: And without the finding, the please share more code would be harder to do. They built ... they invested in many ... probably over a year or two in a build system to compile all this code all the time on lots of machines in parallel. So that if you made a change to some big piece of, like, Google Search or Google Maps, you could send it off to be incrementally compiled, just the things that changed on a bank of machines somewhere in the world and it would finish 10 or 50 times faster than my experience at Adobe and the experience at many other places that haven't invested in build systems.

Spicer: You're waiting to compile something?

Sites: Yeah.

Spicer: Yeah.

Sites: And it's all software. It's all very, very, very bright people invested in the infrastructure instead of only the second or third stringers work on the build system. And that makes a huge difference. It's also the only place I've worked where they would actually spend the money on things that were cost-efficient. Like, every conference room there that had projectors had two projects in the ceiling. They were not cheap projectors, but there were two, so that if one of them broke, you could still have your meeting. Right? Because the people's time is actually worth more than the price of the projector. Right? But other places I've worked over the years, they could do that arithmetic and say, yeah, it would be cheaper if we had two projectors and then not do it. <laughs>

Spicer: Yes. Right. We have three projectors, which we move around as needed. <laughs>

Sites: So I really, really enjoyed working at Google.

Spicer: Oh, that's good.

Sites: Yeah. And the only reason I left is because I retired.

Spicer: Okay. How long were you there?

Sites: Twelve years.

Spicer: At Google?

Sites: Yeah.

Spicer: Oh, I didn't know that.

Sites: Yeah.

Spicer: Oh. Okay. Guess I ... from ...

Sites: 2004 to 2016.

Spicer: Okay.

Sites: And I originally was going to ... I started the first week of February in 2004 and around 2013, as I started to plan ahead of what to do next, I decided I would pick a date for retirement. And then, work backwards from that for what to do, and also, work forward for what to do afterwards so that I'm not sitting at home and my wife strangles me.

<laughter>

Sites: So I picked ...

Spicer: Yes.

Sites: ...the first week of February 2016 as exactly 12 years.

Spicer: That's good, but ...

Sites: As it turned out, the last project I was working on ran a bit late, so I retired the first week of May 2016.

Spicer: Now, this is just my own view, but you don't strike me as someone who is ... who retires. Am I right or do you still have some projects you're working on or ...

Sites: Yeah.

Spicer: <laughs>

Sites: The truth is I'm semi-retired.

Spicer: I thought so.

Sites: Yeah.

Spicer: Yeah.

Sites: So at Google, I also worked on this interface of hardware and software and trying to understand performance and improve it. So I worked on, as I mentioned, the early subroutine call return and I did with Ross Biro, a package of operating system kernel patches to trace every single transition between user mode and kernel mode, every system call, every system return, every interrupt ...

Spicer: Sorry. What OS is this?

Sites: Linux.

Spicer: Linux?

Sites: Yeah.

Spicer: Okay.

Sites: Linux x86.

Spicer: Okay.

Sites: Every fault return, every context switch, every switch into and out of idle. And to do that on all the processor cores with less than 1 percent overhead with 200,000 transitions between user and kernel mode every second, every core.

Spicer: Wow.

Sites: So we built that.

Spicer: Only 1 percent overhead?

Sites: Yeah. My current version is a quarter of a percent overhead.

Spicer: That's incredible.

Sites: Oh, no. It's just deciding to do it and doing the really, really careful engineering, again, based somewhat on my Fundamental Algorithms work or ...

Spicer: Boy, that one course seems to run like a thread through your whole career.

Sites: It focused me on performance.

Spicer: Yes. And did you work with Len Shustek at all? Because he was doing performance-related ... a dissertation [about the IBM System/360] Model 91.

Sites: Yeah. We didn't work directly, but we shared time together there. Yeah. So the tracing is basically just a time stamp and an event. Time stamp and event. Time stamp and event. And the time stamp and event, I engineered to fit into 32 bits. So there's 20 bits of time stamp and 12 bits of event. And the time stamp wraps around every 20 milliseconds or so and you have to occasionally have the high-order ~~harder~~ bits put in so you can reconstruct the whole works later. But you don't have to do anything very expensive during the run.

Spicer: Okay.

Sites: Okay. The 1 percent goal was real simple. If I had a tool that would tell us why things are slow in a data center and it had 10 percent overhead, for the people who run the data centers, you go waltzing in and you say I have this great tool to tell us why things are slow. It'll make everything 10 percent slower when you're running it. The answer is no.

<laughter>

Sites: Their job, properly, is keep the data centers running and don't spend a lot of money on hardware you don't need.

Spicer: Right.

Sites: Keep things efficient. If you go in and you say it's 1 percent, they say, oh, sure. Nobody cares. And if you go in with some other number in between 1 and 10 percent, it's a long discussion.

Spicer: Yes.

Sites: And in the end, the answer's no.

Spicer: Yeah. I would think so, given the cost of energy and so forth.

Sites: Well, I wanted to be able to trace live transactions during the busiest hour of the day.

Spicer: Right.

Sites: And so, 1 percent was the design goal. If I couldn't do a design that had less than 1 percent overhead, we couldn't use it.

Spicer: Right.

Sites: This was at a time when tcpdump had 7 percent overhead, and we could not use it in live data centers. Tcpdump has gotten faster now. Anyway. So I built this in 2006 and we used it and we started finding out some of these weird dynamics of these multithreaded search and map tile and other algorithms, Gmail. Sometimes, we'd go out to lunch for good fractions of a second. Nobody knew we got it ... we were able to observe and, therefore, show where the time is going and why. And lots of unexpected lock contention, as I had seen many years before with Microsoft SQL Server.

Spicer: And this little incremental analysis that you do multiplied by a billion users, or whatever, really adds up.

Sites: Well, mostly multiplied by a million machines.

Spicer: I should say machines, yeah.

Sites: We make those machines a few percent more efficient, it turns out that's worth a lot of money.

Spicer: That's a lot of money.

Sites: So I worked in that space at Google. I worked on tracing user/kernel stuff. I later worked on ... worked in the Gmail group, and, you know, that's a very complex, layered, software with a front end and a back end, which calls a database back end, which calls a replicator back end, which goes off to all of your Gmail stored in at least two different data centers that are geographically separated. So when the earthquake hits California, you don't lose everything. There's another copy in Amsterdam, or whatever. And that goes through a bunch of locking and a bunch of disk caching and buffering. There's dozens of layers. And you get some "give me my next e-mail" and it takes 12 seconds. And the question is why did it take so long? And as you go through the layers, everyone says, ah, the problem is this next layer down took too long. And so, I was left with ... everyone blamed the disk layer, which had no one else to blame.

Spicer: No one else left?

Sites: Right.

<laughter>

Sites: So I went ... I spent a couple of years working on disk server software and hardware to make the disk layer not slow and, in particular, not unexpectedly slow. We were interested in the long-tail latencies.

Everybody's average was good. Google and everybody else has lots of people who can make fast averages, but the 1 percent slow transactions are the things that ruin you when you're doing real-time responses to million or billions of people. And in particular, you size how much work to send to a data center based on sending more and more work until the 99th percentile latencies get to be too large, and you back off a little and you say, okay -- so this hardware can do this many transactions per second, just empirically. Okay? If you can understand and control the long-tail transactions and bring them in a lot, you can load up the same hardware with 10 percent more work, and there's a lot of money in that.

Spicer: Yeah. Ten percent is huge.

Sites: Yeah. So I looked at disk servers and eventually built a stand-alone disk server. I came to the conclusion that doing disk serving on the machines that were doing all the other work also inherently had uncontrolled long delays. The interference from other programs, other users of the disks, operating system going out to lunch when it was close to running out of memory, et cetera. All of those ... I'd gone through at Google 10 years of hand waves of we can control interference between programs, and none of them came to fruition. So I said, well, if you really want, honestly, to control the long tail, it's going to have to run on a separate machine that's dedicated to running the disk server software. So I went and bought a bunch of parts at Fry's and Amazon and built, from scratch, a little Intel i3 four-processor… [computer] Actually, two-processor, hyper-threaded machine. Brought up the Google internal Linux on that machine. Connected a box of a dozen disks to it. Got that running. Measured where all the time was going. And then, worked with the software group to get rid of a few delays in the software, but mostly to demonstrate that this little cheap processor, I built the whole thing for $250, could run a dozen disks and not have big, unexpected delays. And so, since then, they've switched to their disk deployment to that scheme.

Spicer: Yeah. Well, that's a massive change across their data centers, isn't it?

Sites: Yeah. It also had the great advantage that you could ... that they could now add more disk capacity without adding more expensive big processors. Or they could add more processing capacity without having to add more disks. Before, they were always coupled.

Spicer: I see.

Sites: And so, the decoupling allowed some incremental efficiencies in adding either one. So I worked on that for a bit, and then, I worked in a ... near the tail end on a network chip group to do ... to look at the next couple of increments in high-speed Ethernet. The time I started, 10-gigabit Ethernet was relatively new and expensive, but heavily used with the data centers.

Spicer: Okay. I was going to say this is for within the data center?

Sites: Yeah.

Spicer: Yeah. You're looking at ...

Sites: And so, we started looking at not 40-gigabit, but 100-gigabit. What would it really take to have a system that does 100 gigabits? First thing is is it completely saturates the fastest current PCI, the PCI 3.0. Just getting the bits all ...

Spicer: The network ... the standards?

Sites: Just getting the bits off the network, it takes 16 lanes of PCI 3.0.

Spicer: Oh my gosh. Wow.

Sites: And that's only if you build hardware that uses all of the bit ... all of the bandwidth and doesn't have big gaps in it. So I worked on how we ended up ... I now have several patents issued on things like routing Ethernet packets at line speed at 100 gigabits per second in hardware, independent of the protocol. So it's not dependent on, oh, this is only an IPv4 packet, and I know where the bytes are to route it.

Spicer: They never would have considered ...

Sites: Arbitrary bytes.

Spicer: They never would have considered something from the HPC world, right? The hyper ... like HIPPI or ... InfiniBand or ...

Sites: Yeah.

Spicer: Would they have ever considered that or is that overkill for this ...

Sites: I have a little bit of experience with InfiniBand experiments. The original design only scaled to 256 nodes. So it's not expected to run across a million machines.

Spicer: Right. That's true..

Sites: Later ones have expanded a bit. The main problem is it's guaranteed delivery, which means that unless you have infinite buffering inside your network somewhere ...

Spicer: Yes.

Sites: ...that you're essentially doing XON, XOFF. You're telling the sender not to send. All right? And you can buy ... well, this is a few years ago. You could buy chips that would do Infiniband and do all the hand shaking and they would work well for ... on the order of eight connections coming into a machine. And there is a bunch of dedicated hardware inside the network chip to do that.

Spicer: Yeah.

Sites: But if you had 20,000 connections, it all fell over.

Spicer: Okay. Right.

Sites: Because there was only eight sets of hardware.

Spicer: Thanks for explaining that.

Sites: <laughs>

Spicer: Yes.

Sites: And so, the marketing number said how wonderful it is and ... but if you asked what are the marketing numbers if I have nine connections? Hundred times slower.

Spicer: <laughs> Right.

Sites: Right? So scaling issues. Also, one of the problems of XON, XOFF is if you ever have any mistake that does an ~~doesn't~~ XOFF, it doesn't ever XON. If that is said to network hardware within a switch in a big room full of machines, pretty soon, that turns off transmissions from other switches. And pretty soon, after less than a minute, your whole data center is locked up and nobody sent any ~~eight~~ packets, if you have such a bug. It's really impressive to watch what happens when this bug arrives. <sucking sound>

Spicer: You're good at shutting down data centers.

<laughter>

Sites: I didn't do this.

Spicer: I know. I know. I'm just kidding.

Sites: I'm just a naïve observer. I did shut down one data center in Hong Kong over 4th of July one ...

Spicer: Want to tell us about that?

Sites: I did a bunch of international text code while I was there. So I did all the Unicode input encoding detections from all these different encodings in the world of text bytes. And detecting the encoding and then converting the encoding into Unicode UTF-8, I wrote all of the production code for that. And I also wrote language detection for UTF-8 text for 160 languages, some of which, 80 or 100 of them are in the Chrome browser. That's my code running on your machine when it says this webpage is in Romanian, would you like it translated?

Spicer: Neat.

Sites: And it's in the browser because it was 10 times smaller than the previous iteration of language detection and 10 times faster. But at any case, the Unicode input conversion, we shipped. And the general style, I think, in all big data centers, but certainly at Google, is you build new versions of whatever, in this case, the front end of search, say, on a Monday, every Monday, and you deploy it to four machines on Tuesday. And if they don't do unusual things, then you deploy it to 20 machines on Wednesday. And if they're okay, you start deploying to entire data centers around the world and eventually, you get them all on ... you start over. So they deployed to a handful ... some of my input conversion code to a handful of canary machines and everything was good. We deployed it to more and everything was good, so we deployed it worldwide.

Spicer: What's a canary machine?

Sites: Something that you just try to see if it falls over. The reference is to canary in a coal mine, when they used to literally have canary birds in coal mines because if the bird keeled over, that said there was too much carbon monoxide, or whatever, in the air and that the people needed to get out. My wife at one point worked at a [semiconductor] fab line in Danbury, Connecticut, where the canaries died and they ...

Spicer: Oh my gosh.

Sites: ...got everybody out and closed the fab lines for six months.

Spicer: They literally used canaries for the ... as their ...

Sites: They had some gas ... they had some unknown gas leak.

Spicer: Oh my gosh.

Sites: It was making people sick.

Spicer: Yeah. Well, are nasty gases used in semiconductors.

Sites: Yeah.

Sites: So anyway, it passed all tests and got deployed. And then, the following weekend, the Hong Kong data center machines started crashing and rebooting regularly. And so, the people running the data center tracked down the problem and it was my code that was crashing. And it was one of these things where, as often, there's this whole series of five different events that led to making that data center much less productive than normal. <laughs> They ... normally when that happens, they would roll back to the previous week's release of whatever, in this case, web search, except they didn't have the previous release because they had run out of disk space and shipping bits across the Pacific was, at the time, slow and expensive. And two weeks back, they had ... but there was some serious bug that had been fixed in parallel that they didn't want to re-introduce.

So the people in charge of running that data center looked at the crash rate, did a little back of the envelope arithmetic, and said we can live with this until Monday. Just let them crash and take 15 minutes to reboot. And the load over the weekend is low enough that ... but it ... so I went to the postmortem for this, which was a formal process. Here's this incident in a data center. Here's the people involved. And it was incredibly positive. I was just blown away. It was entirely about what to do we need to fix? And it was not a whit about who did what wrong? Not at all. So somebody had already put out the written timeline of what happened. You know, what they observed when in the data center, what they decided, who they decided, and how they found out what was wrong, et cetera, and notifying me. And I apologized to the people on the video call in Hong Kong. I said, "I'm sorry that I took up your time." And they said, "Oh, no, no. This is our job. We get paid to do this." <laughs> And there was a sequence of about five bugs, the most important of which was that my code bug was only triggered by Chinese text in the GB 2312 encoding. And the canary machines that ~~machine said~~ we ... that Google tested on up to then was basically a few in the U.S. and a few in Europe. And so, bug number one was: also test in Southeast Asia. <laughs>

Spicer: Yeah. <coughs>

Sites: Which, when they did that, revealed some other issues that had been slipping by. So we went through all the things that had missed a step and did bugs for them and I fixed the code and it went on. But it was one of those wonderful experiences of expecting a bad meeting and finding that it's a really supportive culture.

Spicer: That seems very unusual..

Sites: I think it is. I think the industry needs a lot more of that. It's like the airline industry, where pilots can report incidents without getting dinged.

Spicer: Yeah. Exactly.

Sites: And that's how you find out what's really happening, instead of ...

Spicer: You're right.

Sites: ...everybody keeps it covered up until 300 people die.

Spicer: You're absolutely right. When there's culture of fear, people don't speak up.

Sites: Yeah.

Spicer: Yeah. Well, that's wonderful. Wow. That's really neat.

Sites: So I think that covers most of Google.

Spicer: Okay.

Sites: And then, my retirement plan, starting in 2013, was to spend some time teaching after I retired, going back to my UC San Diego days, because I really enjoy teaching. And so, I put together a course description for teaching students how to observe and measure data ... complex data center software using this ... user/kernel transitions. The good thing about the user/kernel transitions is they're not nearly as frequent as subroutine calls or lines of code or whatever. So they're at a ... they occur at a manageable rate to actually capture, but they give you this wonderful picture of everything that's going in every CPU, nothing missing. So there's ... if you have a trace of 30 seconds and somebody says, oh, I think this is slow because of the disks, you can look at the trace and say there are no disk interrupts. That cannot be the problem.

Spicer: Yes.

Sites: Right?

Spicer: Right.

Sites: If you do any kind of sampling, you can't say there were no disk interrupts because they could have occurred when you weren't looking.

Spicer: Yeah.

Sites: If you look at everything, you can say definitively it is not a disk. In fact, you can say here's what this transaction did. It did ... it ran on this CPU unit. It ran on that CPU unit. It ran this software thread and that software thread. Then, it waited for this lock. Oh, it waited for a long time for that lock. And then, this other thread freed the lock. Ah, the real performance bug is in the other thread holding the lock too long. Okay. You can't see that in any kind of commercial tool.

Spicer: Wow. It seems so beneficial. It's hard to imagine there's ...

Sites: That's why I decided I would teach a crop of students about it and have them use the software. After I left Google, I rebuilt the software from scratch and, in the process, made it four times faster. And so, I did that in Singapore for a semester, in 2016, and then, in Switzerland for a semester last year, 2017. And decided, when we got back, that I had enough material for a book from the class notes. So I started it in February of this year, 2018, working on drafts of book chapters. And I'm about three-quarters of the way through the first draft and starting to look at some place to teach out of the draft next fall. And then, try to get it published as a book. And it's entirely about doing careful measurements of disks and networks and memory and CPU and locks. And then, eventually, building Linux kernel patches, compiling them, getting them running, and then using them to see what the real dynamics are that are hidden by nearly every other tool. Or you use some existing tools, like Strace, or Ftrace, or Dtrace, that have interpreters and text output and stuff, and they're 30 times too slow and you can't use any of them in a real data center. And in the process, I'm also educating a bunch of students about what's going on inside

data centers that they never get to see. You can't go inside. And to bring them up the curve a little bit on the complexity of layers and layers and layers of software and how to think about them and how to design to being able to see where all the time goes. Because if you have, like, some database scheme and you never pay attention to when did I ask the database for something and when did I get that result, you never know whether the database is too slow or whether youre asking code is too slow or whether some other layer is too slow. So you have to design all that in. And when I was interviewing people at Google, that they didn't ... most incoming students have no clue about large layered commercial software. So it's hard to hire people with any kind of experience in understanding performance issues. It just takes several years of on-the-job training. So in the disk server space, I found one dynamic, one place where, suddenly, all the disks on the server would start getting served very slowly, and that would go on for minutes. And then, it would go back to normal, apparently spontaneously. And while it was serving slowly for minutes, everything built on top of that disk server was slow. And [after] finding the root cause, [it] took about 20 minutes to fix and paid for 10 years of my salary.

Spicer: Oh my gosh.

Sites: And it was from data that had been lying around forever, that no one had looked at, and I built ... I spent a while building the tools to be able to draw pictures of it and see what was really going on at a blow-by-blow level. And as soon as this page came off the printer, I was doing PostScript at the time, of this nine-minute interval at 7:11 A.M., of doing stuff and looking at the individual disks, I could see immediately what the issue was. And so, I hope, as part of my retirement, to train ~~try~~ a few more students to get good at this sort of thing.

Spicer: Sounds extremely germane to today's technology, like a skill that people would really like to have. So that's good.

Sites: And then, after the book, I'll sit at home and my wife will strangle me.

<laughter>

Spicer: Well, this has been great, Dick. Is there anything else you'd like to say as we wrap up our oral history?

Sites: Not that I can think of.

Spicer: Reflections on your career at all that you want to think about, you want to mention, or ...

Sites: I've been very privileged, and I've had the support from all these people every year, suggesting jobs, encouraging jobs. I mean, the two overseas teaching things came through asking a lot of people where might be receptive. And somebody I knew knew somebody in Singapore and somebody else I knew knew somebody in Switzerland, and they made the connection and then, it happened. I would discourage millennials from filling out online employment application forms and doing nothing else. Go find out more about the company. Go write a resumé for that company talking about ... since you've

researched them, talking about what they're trying to do and what skills you bring to that ~~the hat~~. And a different resumé for another company or another institution about what they're trying to do and what skills you bring. Don't do one resumé and send to 200 places. They'll all fall flat. And I think that's something that a lot of students really are never taught. And when people of my generation, your generation, graduated, there were jobs, particularly in the tech field. So you'd just go out and get a job and keeping going. But the last 10 years or so, finding jobs has been really tough for people coming out of college and an extra skill of really explaining to a company who you are and why they want you ... it's your job to explain yourself to them. It's not their job to read your resumé and realize what a wonderful candidate you are.

Spicer: <laughs>

Sites: You have to tell them.

Spicer: That's right.

Sites: … and you can do that in ways that are relatively modest. You don't have to expand your credentials.

Spicer: Well, it'll set you apart from everyone else, for starters. Because I don't think anyone does that, practically.

Sites: Yeah. When I interviewed people at Google, I would often ask them why are you applying to Google? Why Google? Why not somewhere else? And some would say, oh, well, there's this project driving cars and stuff that I'm really interested in, and here's my background. And that's good. There are other people who would say, oh, it's such a great company. I think, well, I know that. But why is it a good match for you?

Spicer: Right.

Sites: And why Google is better than Facebook or Microsoft or Yahoo! or Amazon or whoever? What's the match? If you can't articulate the match, I don't think we should hire you.

Spicer: Right.

Sites: <laughs>

Spicer: Yes.

Sites: And some people were shocked by that statement.

Spicer: Yeah. There's certainly a lot you can do to prepare yourself. There's no excuse for not being prepared.

Sites: Yeah.

Spicer: Yeah. Well, this has been great. And thanks so much, Dick. This is a ... what a wonderful ...

Sites: Thank you.

Spicer: ...interesting life you've had. So I'm glad we got it down on tape, as we call it.

Sites: Well ... <laughter> Okay. I appreciate it. Thank you.

Spicer: Thank you.

END OF THE INTERVIEW

