[LAUGHTER AT BEGINNING]

RL: You want me to tell you the first thing I remember or something.

BL: I want to know how you fell into the company. Under what circumstances.

RL: Digital is the only company I applied to when I left college. And the reason was that Digital, the school that I went to, which was at the time called the Polytechnic institute of Brooklyn, got a PDP-8, straight 8, when I was in my sophomore year. And the school at the time had a computer center, it had an IBM-7040 or something. It had a big bureaucracy built around the computer center, and the machine was behind glass walls, and you submitted to decks of cards to people. And this was something you could go to touch. And there were probably a half a dozen of us who went and did that all the time, to the exclusion of everything else. To the extent that some of us didn't graduate because we got so involved with the computers. I graduated, but I had to pull a fast one at the last minute to get the gym credit to get my diploma. And this in some sense was a ... [UNCLEAR].

[INTERRUPTION]

RL: This was on a much smaller scale, kind of a repeat of what happened. I guess it was very early on. Digital gave some PDP-1s to MIT, right? And those were actually seeds, and a couple of years they reaped all these people like Alan Kotok. This was similar, mainly a PDP-8 came into the school, and about four years later, Digital reaped you know five or six people, who mostly went into the PDP-8 groups in various forms and started writing software for it. So it was a great strategy. The, so I actually joined DEC in '69. I think I was _____. Started working on, I'd been on a basic PDP-8 when I was in school. I kind of brought that with me to the company.

BL: A compiler.

RL: Well, a compiler and a run-time system. It looked a lot like, I don't know if you were around at that time, to see what was called a Dartmouth Basic time sharing system. It had exactly the feel of Dartmouth Basic only instead of being connected to a telephone line, the Dartmouth, you were sitting right there on your very own system. It was completely compatible, because why not? It was brand new. And I had done it in conjunction with another guy. And in fact we both wound up with Digital. And so the first thing we did of course was package this

up and put it, and couldn't in good conscience, Digital in good conscience couldn't sell it as products, it was developed somewhere else, but we cleaned it up and got it working real well, and put it into the Decus library. And then Digital needed some kind of system for the PDP-8. This was just at the time when memory was finally getting cheap enough, where people were actually buying some PDP-8s with more than a minimum amount of 4K, a minimum amount of memory. And we didn't have any quote operating system. That's not exactly the word I'd use to describe what we were doing, that would use more than 4K memory on the PDP. Digital was interested in _____. And I was ready for that, and Chuck Connoly at the time, who's still at Digital, or to the best of my knowledge he was a couple of years ago. And we went out to the University of Michigan, and they had a system which they had done called CPS, the Coolie (?) Programming System, at ____ Labs at University of Michigan. And we kind of made a deal with them. We'd come out and look at it, we'd maybe give 'em some hardware for it. And we came out and looked at it, and it was impressive, and it did use more than 4K, and so we made some kind of deal where we gave them some hardware, and they gave us the rights to it. And we brought it back and I looked at the sources for it, and it turned out it was not really what we wanted, in our judgement. It did use more than 4K, but it wouldn't use than 8K. It was very, very locked

into 8K. And it's user interface was completely incompatible with everything else. But again, just like me, they built a full retention system to look just like the interface to their IBM 360 time sharing system that they knew, and this was ... And of course by that time I had fallen in love with PDP-10s. So Chuck Conolly and I worked a deal. Chuck lied to his management for six months. He said, yes, we bought the Coolie programming system from Michigan, and we're making a couple of modifications to it. I threw it out completely, and started building a single-user system to the PDP-8 that would make the PDP-8 look like you were connected to a terminal on a PDP-10. And after six months, I had done enough work on it that we had something that we could demonstrate to Chuck's management, and say, well, this is the modified Coolie Programming System. These are the modifications, how do you like it. And of course it looked just like a PDP-10. They loved it, they said keep up the good work. And we finished the thing off and shipped and that was this thing called PS-8, which is the first product actually shipped out of Digital, Programming System 8. And that really set a real good tone for working with Digital. Because the message I got from that, it's the output that counts not the process. Right? As in fact for the next couple years, I and a bunch of other people. The way we would literally work is we'd go up and figure out what a neat thing to do

would be. And we'd kind of do it. And you wouldn't necessarily have to tell people what you were doing. All you owed them was a demo every once in a while, to show them that you weren't in fact just screwing off on company time. And if the demo was neat, then they'd get all excited and they'd say, ship it. And you'd go back and clean it up and make it shippable, and you'd have another product. And we did that with some of the extensions, the PS-8, which became OS-8 for obscure reasons having to do with the Nixon wage-price freezes. We wanted to raise the price on it because it was successful. We wanted to get more money for it, but you couldn't raise prices. So we renamed it and it became a new product and we could raise the price. I mean we added features too so it wasn't a scam. There was a system called RTS-8, which was a real-time programming system for the _____, which in fact again was done as kind of a midnight hack, and we demonstrated it to some people and they said, this is great? You know we productized that. But that was real exciting. And I was living just uphill from the Thompson Street parking lot in the Mill, which is now the visitors lot, which was a visitors lot then too. If you go to the far corner of the visitors lot from the entrance, there's a little dirt path. And if you walk up the dirt path, there's two houses. And I was living in one of those houses with a guy named Jack Burness (?) and a guy named Lenny Eldman

(?), who were both from Brooklyn Poly, and we would put in 90 and 100 hour weeks, because to a certain extent there was nothing we would rather be doing. I was the oddball of the group because I actually had a girlfriend in New York that I would take off and go. So I would put in my 80 hour week in five days and then go to New York for the weekend. And these guys would continue into Saturday and Sunday. Anyway, that was real good, and we worked real hard, and we wouldn't have had it any other way. And to a certain extent there are still pieces of the company that run that way. I've grown up and I kind of accept the reality that everything can't run that way all the time. But boy it was sure fun. So --

[TAPE CUT]

BL: -- must have been in building 5?

RL: No, I was in building 12. 12-3. In fact the whole, the software department, I guess was kind of in building _____, 12, and Larry Portner was the vice president. The, well, 5-5 was the PDP-10 group hardware and software. And we would go up there, until we got OS-8 running real well, we were developing our code on the cross assemblers on the PDP-10. Which is why we kind of liked the 10 so much, because it was a real neat machine to run. And so there were PDP-10s, there was a PDP-10 on

12-1, and there were a couple of others scattered around in various places. But if you really needed a PDP-10 to yourself, which is the best way to use a PDP-10, you'd run up to the manufacturing floor, which was I think also on 5-5, although I could be wrong there. And you just pick one of the machines that was in final test ready to ship, and you'd just stop the test and mount your own tapes on it and go use it for a while. And there was plenty of computer power available. But when we got the PDP-8 software working well enough, it became a matter of both convenience and pride that we would just stop using the PDP-10s for development, and use our own stuff. Because you use it, you use your own stuff, it starts working a lot better after a couple of months, because you fix all the stupid crocks (?) in it that are getting in the way of your work.

BL: How many people were doing software on the 8 at that time? It couldn't have been that many?

RL: No. [BOTH TALK AT ONCE] PS-8 was literally done by me. For OS-8, I think there were three people ultimately working on it. There was me, a guy named Paul _____, and a guy named Ed Freedman. And Freedman is around somewhere, and still see him. Paul _____ went back to school. About, I don't know, I guess 18 months or two years after I went to Digital, the second wave of

Brooklyn Poly people came to Digital. And of course they also immediately joined the PDP-8 group. [BOTH TALK AT ONCE] Literally, we dominated that, the PDP-8 development. Because while I was doing the PS-8 stuff, this guy Jack Burness, who came from Poly was working on what would later become the PDP-12 software, the lab software. And this guy Lenny Erlichman was doing a lot of stand alone programs for the 8. Like there was a stand alone basic that became something used system 10. So, we were all working on different aspects of it, but we were all doing PDP-8 code. And the next wave of Poly people came in and we started building FORTRAN for the 8, and we started building a commercial operating system for the 8. This was myself and a guy named Stan Rabinowitz and a guy named Steve _____, who's still at Digital. Was building what turned into [SENTENCE OBSCURED BY BACKGROUND LAUGHTER]. With the Diebold (?) language, which we, the Diebold language was done by somebody else and we either brought it into the company or stole it within the company from that somebody. But all the other parts of it, we did in Diebold, is still being used. There's a lot of commercial software that's still running. God which [?--garbage?], I'm amazed. The group of us kind of stayed with the PDP-8. The PDP-11 was going full steam at the time. But we really didn't like it very much, is the best way. Not so much the computer, but the people and kind of the software vision that they

had and they were bent on making everything as complicated as possible rather than as simple as possible. And so we kind of stayed away from the DOS-11 effort. And there was a lot of in fact kind of sniping at it, because they were trying to do a lot of things the hard way.

BL: Radically new instruction set and --

RL: Well, the instruction set was new, but it was more like, oh, they were building this formal file system that would have all these properties and all these things. And the problem, I mean, there was nothing wrong with that vision. What was wrong was that the size of the machine wouldn't support the vision. And so what they wound up with was a system that overlaid itself with DEC and ran very slowly, and that didn't have a particularly good user interface if you were a programmer or a hacker. And of course we were much more focused on the needs of one community, which was us. We were building these systems really for ourselves. And so we were very, very focused in on the user interface, because we were the users, and we were very focused in on things like speed of execution because we were the users. And so our systems were, were like real good, and you could, because they were fast, you could actually run the system on a DEC tape, which is a random access tape, and get

reasonable response. If you ran on a disk, it was superfast, and DOS (?) _____. So that sniping went on for a while, an in fact that sniping ultimately, and that the division into cancelled (?), and it became the beginning of RT-11. Which was the PDP-11 operating system but looked like OS-8. Now I had very little to do with the actual implementation of RT-11, only I was in the philosophical cheering section. But I stayed with the PDP-8 well into the PDP-11 era. In fact, probably until '73 or '74 or so, and then it became obvious that there was just a limited number of things we could do more to the 8. We were really starting to feel the pinch of the machine's generation, almost decade old instruction set by that time.

BL: Which was a RISC.

RL: Yes, it was. It was an older RISC machine and it was a great design. And you know Alan Kotok did a great job of packing stuff into 12 bits. But the problem was the 12 bits. So I started working on PDP-11s at the time. I started with RISCUS and did a little bit of work on the RISCUS time sharing system, which was the one PDP-11 piece of code that the PDP-8 group kind of admired, because we knew some of the people and they were cool people, and they were building the best product that they could subject to the fact that they had to be DOS

compatible, which they dropped. But moved over and did some work for RISCUS and then wound up on a product called, well, it wasn't called anything at the time, but eventually became something called Tracks (?) and it was a total disaster for the company.

BL: That was Transaction Project [BOTH TALK AT ONCE]. Jumped into the PDP-11 bringing it into the commercial world.

RL: And I started out, let me think about this now, because I want to get the chronology approximately right, because this is a history. Actually it was when I was in the RISCUS group, in '75, I got a phone call to come down and see Gordon about some stuff. Actually, I went to see Gordon about something completely unrelated. But if you interviewed Gordon, Gordon, for all of the fact that he could look like _____, he also was very impulsive in a lot of ways, he lived in the present. And I happened to walk into his office, when he was, when what was foremost on his mind was the need to get more than 16 bits of addressing in a computer. And because I walked into his office at that time, right?, I got drafted as part of the effort to do the wider machine. So I guess this was April 1st, 1975. The VAX architecture committee had its open meeting on April Fool's Day. And we charted to go off and do something to replace the PDP-11, right?

It was actually going to be just to extend the PDP-11. And you've probably heard the story about the VAX architecture committee from several sources. But my version of it is, we spent about ... First, by this time, I had moved up to 5-5 with the PDP-11 and PDP-8 groups that were up there. And 12-3 was actually vacant. So we decided we'd sequester ourselves, so that in fact, we picked 12-3, so I moved into my old office again. And we would literally just, every day, we'd try to meet every day, and usually try to meet for the whole day. And at the beginning it was very unfocused discussions, and we started to call in people and just kind of grilled them on what they expected out of PDP-11 and what they expected out of the follow on. In fact, I remember we brought in Ken Thompson from Bell Labs, one of the inventors of UNIX, to talk about what he wanted out of PDP-11. And it turned out that was an unsatisfactory interview, not because Ken Thompson's a dummy, he's very much an un-dummy. But because we couldn't let him know what we were thinking about. And so he was focused on a couple of point extensions to the PDP-11 that would make his life a little easier. And we were looking at tearing it down and building it up again, right? But we couldn't tell him that. So it was a frustrating interview that the committee had with Ken Thompson. And ultimately we came up with a couple of schemes to extend the PDP-11, and didn't like any of them. And that was when we

decided, and in fact this essentially became some kind of committee motto, that there wasn't any free lunch. That we couldn't get to where we wanted to go by simple compatible extensions. We had to in fact start absolutely from scratch.

BL: As you said rip it down.

RL: And we said, we won't be compatible with the PDP-11, we'll be culturally compatible with the PDP-11. Which means we would treat data the same way, we'd store data in same formats, right? And we'd have the same flavor (?) of programming. But we wouldn't be stuck with absolute compatibility. And in fact the way I remember it at the time, there were, it was interesting. There were two schemes that were kind of brought forth in enough bloom (?) by individuals to be placed in front of the committee. There was a scheme that, I believe a guy named Steve Rothman, who turned out was an old friend of mine from high school, but was on the VAX committee as well, brought forward -- he's, by the way, still around to be interviewed; he does, the guys who do AI work in the Massachusetts, they have a machine that reads to you, not _____, but I can get you his address, and he'd be worth interviewing -- so he --

BL: Kurzweil.

RL: Kurzweil, that's it. He came up with a 32 bit architecture that was very, very simple. In fact, thinking back on it, it's a shame I don't have the documents, because it was kind of like a RISC machine. It was kind of like a 32-bit RISC machine. The details that make a 32-bit RISC machine what it is, about allowing parallel execution and so forth, weren't there. But it was the same flavor as a 32-bit RISC machine. And we looked at it, and at the time, of course, memory was real expensive, and programs in that style would have taken up a lot of memory, and so [BOTH TALK AT ONCE]. And Bill Strecher, basically I would say came up by himself with something that was pretty damn close to the final form of the architecture. We diddled it a lot. But the basic form of an operand, an operations specifier followed by a variable number of operand specifiers, and the operations specifier said how many operands in each operand, and how many bytes of extra data it took to perform that operand -- it was in his proposal from the very beginning. You know, his registers were set up differently and his choice of operand specifiers we mucked about with, but it was amazingly coherent. And anyone looking at it and looking at the final VAX architecture would say, this obviously came from that.

BL: Was he in the committee at that point?

RL: He was in the committee, he was active in the committee. [BOTH TALK AT ONCE] Gordon was not very active in the committee. Gordon launched the committee, was there for the first whole meeting, and possibly the whole second meeting, and then would drop by occasionally to say how you doing? The original committee, if I can remember it, myself and Steve Rothman, and Bill Strecher, and Tom Hastings. And why am I drawing a blank? Let me come back. Later on, people like Dave Rogers, who was at the time a designer on the next PDP-10, but when the committee finally got to the point where we knew we had some kind of thing going here, and we knew that something had to give for us to be able to build the first one of these machines, we killed that project. I think that was the Dolphin project, but I'm not sure. And Dave Rogers kind of came over from the Dolphin project and started working with the committee a lot, because he was going to have to build the damn thing, which gave us a good dose of reality. And Dave Cutler, who wound up on the committee at the end, they came in later when we got real focused on implementation and in fact around, I guess, we started in April. Somewhere in September or October of that year, we produced the first version of the architecture manual. And there was a lot of review, and around December I think we produced the second version of the architecture manual, and at that point, the committee

decided that its work was over, and we disbanded, because we had, you know, we had a reviewed, updated copy. However, the second stage of the committee, the people who were trying to build, architect a set of machines to this architecture, finally decided that some of the features that we put in were making their lives too difficult. But they were missing a couple of features that the software and that they were willing to sacrifice a couple of features to get them. And that there were things that were making their lives difficult. Specifically memory management was too complicated. And I forget the deal with the instruction set. But so there was kind of an implementers' revolt that produced the third version of the architecture the next spring, and that was the architecture that actually finally got implemented to. And so that worked pretty well. It wasn't the formal process to do that, but it worked pretty well. So then, at that point, when the committee disbanded I went back to the software group. And we were doing Tracks, we were going to get into the transaction processing market. We're still trying to get into the transaction processing market a few years later. We were going to do this transaction processing kind of a kernel (?) for the PDP-11, and it was going to have all these features. And I didn't know anything about transaction processing, but there were a couple of people in the company who seemed to. And so I talked to them. And

there was a guy named Andy Wilson who came in from England. He's still with the company. He's worth interviewing. And there was this woman named Audrey Reed (?), who had just come from Xerox, and there was a guy named Dick Hudspeth, who had just come from Xerox. And Xerox had in fact built a time sharing system with a bunch of interesting commercial features and stuff like that. So Audrey was on the project, and through her I met Dick, and we'd do a lot of talking. Dick was of course up building VMS. Dick was super sharp. Audrey was no slouch either.

BL: Was he with Xerox Park (?) ?

RL: No he was at, I think, you could get this better from Audrey, but I believe they were at what used to be SDS, the company that built the [BOTH TALK AT ONCE] and then when Xerox took it over, they were a part of Xerox. But they weren't part of Xerox Park, they were part of _____ building the continuation of the SDS machines for Xerox.

BL: So Dick was wildly working on VMS-2 [2? -- too?] at the same time.

RL: But Dick was the kind of guy that you'd come in and you'd just ground him what you want to talk about and

you'd get good advice right? Dick and Audrey weren't married at the time. In fact, Audrey was about 100 pounds overweight, right? And decided at one point she really needed to lose weight, and so she lost that 100 pounds, and then she and Dick got married a couple of months later. So we were doing this, and the big problem with Tracks was that, it was a problem that comes up again and again with Digital -- we were trying to enter into a new market, and we were trying to enter into a market that wasn't a natural for us. We weren't building the computer for ourselves, right? So, the real question was what this market wanted, and everybody had an opinion. The project got tugged this way and tugged that way. And somewhere during that process, I became convinced that the project was doomed to failure, and I wasn't having any fun, right? Because literally the requirements would keep changing, and some new guy would come on board and give it a different vision of what the market wanted and da-da-da, right? And we were at the time kind of the implementers pool. I mean, it was just like, okay, tell us what you want and we'll build it. But this is a case where we don't know what the customers want, right? And the input was missing. So I got real frustrated, and just around the time we were building another PDP-11, called the 1160, and the project got into trouble. So I volunteered to help them with their microcode. Because it was a good way to do something

that was fun that I understood. I had used the PDP-11 instruction set enough that I knew how to, I knew how it worked, and the users of it were the programmers for PDP-11s, and so I was just writing a program, if you will, that would satisfy the user, and it was very well designed, and so sure great.

BL: Much more fun than --

RL: Oh, a helluva lot of fun --

BL: Had you done microcode?

RL: No, I'd never done microcode. [BOTH TALK AT ONCE] Microcode is complicated as hell, so I really liked it. So I worked on the 1160 for a while, which was never really a good selling machine, but I have a soft spot in my heart for it, because it got me into microcode and to a certain extent into hardware. And in fact, I not only wrote some chunk of the PDP-11 microcode, but I actually unwrote a lot. There problem was --

[END OF SIDE 1]

RL: The problem was that they had some of the microcode already written, but the microcode that they had was overflowing the space that they had, and it wasn't

complete yet. So I came in and kind of crunched it down, and then wrote code to fill in some of the spaces, and finish the functionality. But while I was at it, I could resist. I wrote, I turned it into a PDP-8. I wrote a different set of microcode for it. Actually, it was not different. It was compatible, where you could put the machine into a PDP-8 mode and work on it that way. I got a phone call about three or four years ago, from the people who were working on WPS, which was for a long time, a real good selling, right thing for PDP-8s. It was what allowed us to sell, I forget what they were called at this point, they weren't DEC stations, DECmates. DECmate 2s, and DECmate 3s -- we were selling them on the strength of the word processing. I got a call about three or four years ago from the WPS group to tell me that they were finally retiring their 1160 development machine that they would use to develop and simulate the WPS code, because that's the fastest PDP-8 in existence, and they could do all their time sharing. We actually had a guy named Mark Bramwell (?), who was instrumental in building RISCUS, and a real bright guy, an old-timer who's still with the company, and somebody else you could add to the interview list, to, I told him what I was doing with this microcode. And he immediately went and built a RISCUS what's called a run time system, for the PDP-8. And so you'd run a PDP-8 program, and it would say what run time system are using, and you'd go

that route, and it would go to the run time system, and that would put the machine in PDP-8 mode, and you'd be running in PDP-8 for your time sharing code. That was a good hack. To get back to try and track the company history. So I used the, the 1160 project to kind of wean me from Tracks. At the time, I didn't know it at the time, but I was also weaning myself from software, which is good, software was getting too complicated, and learning about hardware. I was a math major at college, I didn't know anything about hardware. And hardware was of course fascinating, because it was a field with all kinds of arcane rules, and things that could go together in certain ways but not in certain other ways. So, at the time there was a CPU advanced development manager I guess named _____ _____, who had an idea for building an innovative kind of multiprocessor. And I talked to him and wound up on this project which is called pulsar. Which is an obscure advanced development project. And we actually built the machine. We built an 8-processor PDP-11 that worked and ran RSX and everything. It took us about, it took a little over a year to build it. And then probably another six months to, to get all the, all the stuff down to the point where it was running RT-11, I mean, RSX-11. And I actually did a little bit of hardware, like baby steps kind of hardware design on that, and did some of the microcode to the LSI-11s, and did RSX modifications. But during that

time, the 11780 project, this was now '76, '77, the 11780 project was under full steam. Dave Rogers was doing hardware, oh my mind is going, last name is Hughes, first name I forget, was doing kind of the analog parts of the design. A guy named Judd Leonard, who's name will come up again and again, and especially if you talk to the KL-10 people, was in charge of the microcode. And they were running out of space for their microcode, even though they had allocated a shitload of space. And so I went on loan from the Pulsar project, back into the VAX world, to help those guys out, which was a lot of fun. Because this microcode made the 1160 microcode so simple. And I've gotten away from CPU microcode, but I've heard that the 8800 microcode makes, then the 9000 microcode makes this stuff looks simple. And this kind of helped out this time as more of a hired gun. Came up and did maybe three or four months of work on the project, and pulled a lot of code out and then the guy who was doing the floating point left the company. The guy who was doing the floating point microcode I think left the company. So I took the floating point microcode over, but it was mostly written at the time, and all I did was tune it a little and get some bugs out and things like that. And then went back to the Pulsar project. And when Pulsar was near completion, a guy named Barry Rubinson, who I'd never really met before, came up to me and said that the storage group was building a facility

in Colorado, and that they were going to do all kinds of wonderful things with storage and they were going to do architecture and they were going to build advanced storage controllers, and stuff like that, and was I interested in interviewing for a job out there. And I'd been at Digital in Massachusetts for nine years at the time. I was married, and you know, so it was kind of like, yeah, I was 30 and it was time for a change, and I thought yeah!

BL: That's interesting because --

RL: And wound up moving out to Colorado not quite with the first wave of storage people, but kind of in the second and a half wave when we were still in temporary quarters and things like that.

BL: That must have been an interesting move. Did you feel like you were going to get away from the central, the heart of the company.

RL: Yes. Yes. In fact that --

BL: In fact your reputation at that point was huge.

RL: That was a big issue, and I think one of -- well, there was a big fight because Gordon accused Barry of

stealing resources and stuff like that. But Barry's got a very hard head, and knew what he wanted. And Barry works very, very well in this kind of entrepreneurial start up mode. I think what was going through Barry's mind was that if he got enough good, good people who were [BOTH TALK AT ONCE] had a lot of connections back, [BOTH TALK AT ONCE] we wouldn't be as cut off. Exactly. But at the time we moved out there, there was no electronic mail. I still have the first mail message I ever got, because I saved it, you know it's like the first dollar you ever make. I have the first electronic mail message I ever got, and it was Mark Bramhall. It was in the spring of 1980. And we moved out in the summer of '78. And did things the old way with telephones and typed memos and, you know, right? And it was not easy. It was Digital's first real attempt at remote engineering, because Seattle had not started up yet. Tom Stockebrand had already moved to Albuquerque but Tom Stockebrand didn't move to Albuquerque to start a major effort. Tom Stockebrand moved out there to get away from the bullshit. And the fact that he was able to do some work down there was completely incidental to getting away from the bullshit. This was the first time that group that was, it wasn't me that did this. It was people like Lance (?) _____, and Mike Riggel and Barry Rubinson who strategized doing this, deliberately moved a group out. This was during, Digital had these periodic fits of

oh my God we're using up the labor pool in Massachusetts. Which they were, all of New England. And plus, one of the factors in choosing Colorado I believe, I've been told, is that you get engineers from California to move to Colorado, but you could not get them to move to Massachusetts.
[BOTH TALK AT ONCE] And there was already a telephone, there were already plans to put a customer service center in Colorado, and a lot of that influenced the choice of Colorado Springs, plus Colorado Springs made it very sweet for Digital to move in. What was your question?

BL: Did that actually happen?

RL: That we could attract people from California? Yes. We actually did. In fact we had a lot of success recruiting people from Cal Poly. We recruited a substantial number of people from Cal Poly in San Luis Obispo, plus, and got some seasoned engineers as well. And we had a lot of people we got from Massachusetts, because they said the same thing I did. Ooh, Colorado -- mountains, low humidity. We moved out there, and as I said this is our first experience in remote engineering. And that was one of the things that forced kind of a formal architecture process on us for storage. First of all, we were from Digital and Digital was architecture crazy. So, Digital, when you started contemplating

anything big, you'd say, well, first we need an architecture. Well, of course, we contemplating something big in storage so first we needed an architecture. But also, the architecture really helped when you were doing this remote engineering process because it gave you a formal spec (?) _____.

BL: And you could work in parallel.

RL: Mm-hmm.

BL: But you never developed --

RL: The biggest problem we had was in fact not the architecture, not the communication with the operating system groups. That worked very well because of the people involved. The biggest problem we had was that there was a set of, let's call them, non-written corporate engineering rules, which would tend to get formulated on the fly by people like Gordon Bell and let's say Bill Strecher even. Which were formulated for good reasons, all right? They were things like non-proliferation of designs maybe is the best way to put it. Namely, don't design your own widget, if there is a widget that somebody else is designing that you can use in your product instead, right? And that's a sensible thing in general. So, for instance, there was a large

push, there were cheap microprocessors coming on the market right around this time, starting with the 80-85, which was actually a few years before that. But Motorola was promising the 68,000 and other people were promising various things. And we needed a microprocessor to use for a piece of intelligence in the machine. And their was an edict saying, you shall use a PDP-11. Because we build them, and we're building this low cost chip set, which was the 11, turned into the 1123.

BL: Maybe another non-written engineering rule.

RL: You shall use a Digital architecture, because Digital architectures, CPU architectures are as good as anybody's. And we, our Hudson (?) plant needs the volume, and da-da-da. And the two big thrusts that were coming out of central engineering was, that we shall use a Digital processor, and that we shall use a Digital backplane (?) bus as the heart of the kind of interconnection system inside the big controller that we were building called the HSC. And we fought back on those, and we won one and lost one. We won the right not to use a Digital, an already existent Digital bus, which the people were pushing the BI as that bus. The BI was a nice bus, but it turned out it was a real good thing we didn't use it, because first of all, it shipped after we did, right? And secondly, it would've been way too slow

for what we wanted to do. We actually convinced them of that, and they backed off okay. But we lost the war on the PDP-11, and we were forced to use a PDP-11, and in retrospect it was a terrible decision, because we ran out of address space very quickly on the 11. An in fact, I even -- after Gordon had left the company, we were having a conversation about something else, and I kind of stuck that in his face and he admitted, yeah, yeah, we were dumb. Well, you know, his, you know, you can't blame Gordon for it, because at the time he was high enough up in the organization that he couldn't be a pure engineer anymore, he had to be an engineer-slash-bureaucrat, and had to worry not only about the niceness of the designs, but of how much money it was going to cost the company to do everything that was being done. And so from his point of view, it was, he was fighting the good fight. So --

BL: You used the 68,000?

RL: Mm-hmm. The group in Colorado wanted to use the 68,000. Now it turned out the 68,000 Motorola was promising, the date that they promised it for and the date that they actually delivered were probably nine months to a year apart. That would have caused us some pain. But in the long run it was absolutely the right choice. You'd have much bigger address space. You could open it, you could move upward to a family of chips that

was much faster and got a lot more performance. But the HSC was a successful product despite it, but it would have made people's lives a lot easier, saved us a lot of money, and it would have been more successful. The other thing was, and this I think was part of the kind of separation blues of having engineering in two places, that we came up with a plan for the HSC, and it was quite ambitious. And when the people back east looked at it and they noticed the HSC had within it, most of the distinguishing characteristics of a computer system, right?, they got very alarmed. Those guys out there want to build computers in competition with us. No, no, no, we are building a special purpose -- really. That wasn't a formal battle. That caused an undertone on all the other battles, right? Gordon was especially going -- you know, the main reason was that Gordon had a, I don't know what you want to call it, a philosophical theory which he expounded in some of his books, and which has a lot of truth to it, called the wheel of reincarnation. Which was that people initially do a function with a general purpose computer, and then this function is identified as being a performance problem, and so you build a specialized piece of hardware to perform the function, and this makes your life a little easier it turns out. And so the next time you build a more elaborate piece of special purpose hardware to perform that function even faster, and you continue to do this until in fact you

have taken this specialized piece of hardware and expanded it to have most of the characteristics of a general purpose computer, at which point you're handling this function on a general purpose computer, and then you wind up building a specialized piece of hardware within the context of this specialized piece of hardware that you were building to do the function, and this is the wheel of reincarnation. He was worried that we were going down that garden path. But, let me see, keeping this corporate, the other interesting engineering issue, if you will, came up shortly before I left Massachusetts. We got a real scare from what was at the time a start up company called Tandem, Tandem Computers.

BL: Because in '77 they started coming out with --

RL: Right, in '77, they started coming out with things, and these were fault-powered machines. And Bell Laboratories, not Bell Laboratories, but the Bell Telephone Company, AT&T, which hadn't been broken up at the time I believe, took one look at these machines and said, to Digital, these are the kind of machines that we need to run our switches. You know, we had been selling PDP-11s into AT&T to use in phone switches. They said, these are the kind of machines we need to run our switches and if you're not going to sell us these kind of machines, we will buy them from Tandem. And of course

was a huge customer of ours at the time, and this threw a real scare into us, and so Gordon started up a project which was the project to build some kind of DEC-fault (?) tolerance system. And we took a look at the Tandem system and understood how it worked, and came to the conclusion or decision that the way that they built it was okay for a company that was starting from zero, because they built a lot of fault tolerance assumption right into the guts of the machine. Buses were organized in a way that was fault tolerant etc. And it wasn't that that wasn't a good idea, but it wasn't the right idea for us, because we had, we were already in the computer business, and we were already making a huge line of computers, and to start from scratch with a fault tolerant design would be a significant diversion of resources, right? That we somehow had to figure out a way to layer fault tolerance on top of what we had.

BL: Again, your compatibility penalty.

RL: Right. And so there was a big kind of kickoff meeting in which we debated ways to do it, and I believe it was Gordon who drove the meeting, said, what we need is a bus to hook the machines together with, in a fault tolerant way. And laid out the vague characteristics of that bus, and then again, as Gordon would do, at the kickoff meeting, and like take a hike and let people

ferment. And over the next couple of meetings what emerged from that was a bus called the CI.

BL: That was part of the interconnect strategy.

RL: Well, at the time there was CI, NI, and BI kind of all going on at the same time. And people were doing Old McDonald jokes about that. Because the Ethernet effort was just kind of starting up. This was '77. In fact, I'd say Ethernet really didn't start up until we had a chance to catch our breath from the 7080 in 1988, or at least that's the way I remember it. But I think the CI stuff actually started before the 780 shipped, because it started, or maybe just shortly after the 780 shipped. It started I'd say in early '78. Because I left Massachusetts in mid '78, and it was already well under way. This is also around the time I guess that the VAX group kind of picked up and moved to a deserted shopping center in Tewksbury. A really wonderful work environment. There were no windows. So we were out in Colorado building storage controllers, and we had the problem about what bus to use to connect the storage controllers. There were hoses (?) that had we come up with our own bus, etc. And so the one remaining _____-yank (?) around, if you will, that occurred to the HSC project is that when the CI finally, when the spec finally settled down --

BL: They could use this.

RL: Right. Instead of just using the CI to connect computers, somebody came up with the idea, and it wasn't us I don't believe, about why don't you guys hook to this CI thing directly, and then it won't just be a computer interconnect, since that was its name, despite its name, why don't you guys hook to it directly. And so that caused us to kind of change course. This was exactly the right thing in the long run, of course, right? But it did cause us to take a hit, back off, get reinvolved in the CI process in a very intense way and essentially I wound up doing that part. And the other thing that did -- you see, it solved a lot of our problems, because we were going to build a special storage hose. And not only did we have to build our end of the hose, but we had to build the other end of the hose too. The board that plugged into the
[BOTH TALK AT ONCE]. -- gave us that board for free. So we were all in favor, right? But there was a lot to learn, etc. So we --

BL: It probably didn't fit into the architecture you were doing either.

RL: No, it did. Surprisingly enough, it did. We knew

that because when we did the Digital storage architecture, we had a choice of let's say fundamental paradigms to use for how you talk to storage. You could, for instance, talk to storage like it was memory. In fact, there are people advocating that paradigm today, but nobody's successfully really done it well. But the paradigm that we chose, which was not all that unobvious, was that you talk, you use a network, it's a network paradigm, a message sending, you know, paradigm to talk to storage. And given that, we could in fact layer our storage devices on anything that looked like a network. And the CI looked enough like a network that it worked. No, we had no problem with that. The problem we had was that we were designing to a moving target that we had no control over. That was our fundamental problem, right? But, on the other hand, the benefits of doing so were so obvious, that that didn't bother us at all. However, what it did to, and this is kind of a shame I have to stick this in, is that if you ask Ken about those guys in Colorado, and how they did, he'll tell you, oh, HSC is a wonderful product, but they shipped it late. And ra-ra-ra-ra. And of course we shipped it late, because you couldn't ship the HSC until, I mean this is not to cover up the fact that in fact we were off our own schedule. But the people that we had to talk to were sufficiently off of their schedule that in fact, if we had shipped it earlier, it would have just sat there.

Because there was no way to talk to it. So it was the need to do that coordination that turned the HSC from an isolated effort into a joint effort between a group in Colorado and a group in Massachusetts, two groups in Massachusetts, a hardware group building the CI adapter, and the CI, and a software group building the VMS, you know, software to deal with this monster, that was as responsible for that slip as anything else. Yeah right, he said defensively. Let me go back and see if there's any corporate things that I should have talked about but didn't.

BL: You have wonderful tales.

RL: One thing I should mention because I don't know how many other people will talk about this, in terms of corporate architecture. The whole storage effort in Colorado, the whole Digital storage architecture was actually our second cut at a storage architecture, okay? And that was the one that was initiated kind of by, as I said, Mike Riggel and Barry Rubinson. The first one was actually initiated by Gordon. It was something called the Mass Bus. And what Gordon perceived at the time, and he perceived it correctly, was that we had -- at the time the company was divided into PDP-10s and everybody else. Because this was pre-VAX. And that the PDP-10 group had their own storage organization that was busy buying disks

from people. Yeah, they were all buying disks from people, and putting them on buses and building controllers and all this stuff. And the rest of the company was buying storage and also building storage, and putting them on different buses and working them up. And we were spending a lot of money doing this. And so the Mass Bus was designed with, in some sense, the wrong set of goals in retrospect, namely it was designed so that we could use the same storage on all of our systems. It wasn't, its goals didn't necessarily include cost performance, etc., but commonalty was the main goal. And in fact it achieved that. But it was. It came at a very unfortunate time in history, namely the Mass Bus was conceived of around probably 1974, and the first microprocessors hit the market around 1974.

BL: So it didn't use the intelligence.

RL: The Mass Bus was a bus that not only didn't use micro-, didn't make any concessions to microprocessors, it was in fact designed so that it could not be implemented with microprocessors on either end of the bus. It had to be implemented with hard logic. And this is a standard case of either being in the wrong place -- it's probably mostly a case of being in the wrong place at the wrong time in history, right? And, because in fact nobody at that time really could have anticipated

what microprocessors would do to the industry. I mean I was involved with them early on, but mostly because in certain self-contained systems like terminals, it was real obvious that we were going to just put a microprocessor in there and blow the functionality of the terminals all to hell, you now. Blow it right out the roof. And I'd done a little bit of work on the side kind of with the terminals group on some of that. But in terms of these little dinky, slow, hard to program microprocessors in big systems, oh no. That wouldn't happen. Not for a couple of years at least. So the Mass Bus was set up that, the bus protocol was set up that demanded a very, very quick response to messages, which microprocessors couldn't handle. And it also had a, because we would buy disks from outside primarily, and these disks would come with a lot of logic that did a lot of things for them, because of course the manufacturers for these disks, you know it was hard to handle a disk. And these guys knew how to handle their disks, so they put all the logic into handling the disks, we would buy the disks with that logic. And so the Mass Bus was set up assuming that the disks were very intelligent, and the host was pretty dumb. And what that meant was that when you bought a big storage barn (?), you were paying over and over again for this rather piece of expensive intelligence in the disk, and the disk still sat on a bus that would only allow one of them to transfer at a time,

because it was a real-time bus, the Mass Bus. So in some sense you had the worst of both worlds. You were paying a lot of money, you had these big thick cables running around, you had this large system configuration problem. So, DS, the Digital storage architecture stuff tried to solve that. In fact, and this is just an illustration of kind of how the decisions flip flop with technology. At the time we looked at it, logic was relatively expensive. And so we said, well, it's stupid to be replicating this expensive logic in every disk drive, so in fact we designed, the basis of the architecture at least the SDI part of the architecture was that the disks would be very dumb and the controller would be smart, but the controller would handle a bunch of disks so the expensive logic of the controller would be amortized over all these disks that it was handling. And in fact we built an architecture on that, and that is the architecture that the HSC uses, and the _____, and all those other controllers. And of course nowadays, logic is very cheap, and in fact we're back to the Mass Bus model in some sense. Except that the bus is not a real time bus that connects you to the drive, it's a system bus like a scuzzy (?) or something else, but in fact, all of the intelligence is back in the drives. Go figure. All these decisions, the Mass Bus decision, the DSA decision and the ____ decision are the right decision at the time, but the technology, the world changes and then everything

you know is wrong. That's enough babbling.

[TAPE CUT]

BL: Actually, I just want to ask one more quick. Someone said that you were the most respected person in microcode land, because you had this law that you could always squeeze. How did they go?

RL: This was, this was, the original version of that was actually when we were in Brooklyn Poly, and we called it Erlichman Law, after this guy named Lenny Erlichman --

[END OF CASSETTE]