Oral History of Pat Gelsinger

Interviewed by:
David C. Brock
Doug Fairbairn

Recorded January 9, 2019
Mountain View, CA

CHM Reference number: X8899.2019

Brock: Pat, thanks so very much for joining us today. We really appreciate it.

Gelsinger: Great to be here. Love the Computer History Museum, and anything I can do to support it is really an honor.

Brock: Oh, that's great to hear. We'd like to begin at the beginning, and just ask you a little bit about, first of all, when and where you were born.

Gelsinger: I was born in Pennsylvania, March 5th, 1961, the Amish Pennsylvania Dutch Mennonite area of Pennsylvania, so I was born and raised into a farming family. Dad was number nine of ten kids, and growing up, I always thought I was going to be a farmer. And you know, dad, as number nine, never had his own farm. So all of the siblings, you know, number-- Grandpa helped number one, two, three, four-- it got down to dad at number nine, he says, "Just work with your siblings, right?" We don't need any more farms in the family, but had he had a farm, I'd be a farmer today.

Brock: Was this in Lancaster, County, Pennsylvania?

Gelsinger: It was actually Berks County. Lancaster, Lebanon, Berks, all in that same area of Pennsylvania. I describe it as the Cinderella career, because when I was 16, I accidentally took a scholarship exam to Lincoln Tech in Allentown, Pennsylvania, and I won. I wasn't supposed to take it until the following year, but I took it, I won. So I ended up skipping my last year-and-a-half of high school. And so literally at 18 years old, I graduated with my Associates Degree. I had gotten enough credits to graduate from high school, and Intel came recruiting. At 18-years-old, I'm being recruited by Intel to come out to interview on the West Coast, right? There's sort of an industry-wide shortage for technicians, so here I am 18-years-old, I've never been on an airplane, and I'm being given a free trip to California. It's like, "Sign me up!" But I promised my mom, "You know, they're crazy out in California, I'm not going there! I'm a Pennsylvania farm boy." And, a few months later, boom! I'm moving to California!

Brock: So before you get to that, the 16 years-- or whatever, before you went to Lincoln Tech, did you do any technical stuff? You have hobbies, or you must have had some interest beyond the farm or that was--

Gelsinger: A little bit. They had sort of a vo-tech school that I sort of played around with electronics a little bit, and sort of was intrigued by it. Obviously, I had a good acumen in math and science. And I thought English was a waste of time. I was sort of your natural geeky kind of guy. And then at Lincoln Tech's the first time I touched a computer, right? You know, some of those PDP-23s [PDP-11/23s]. I did punch cards. Did some of the early TRS-80s and played around with it. And this idea that I could tell something what to do, and it would really do it, was just like, "Wow!" Right? And I was good at it! And so I really got into the technology as I was coming there. And it was one of those, the first time you sort of tasted it. It's sort of like, "Ah!"

Fairbairn: So were there any of your teachers during high school or whatever that sort of told you, "Hey, you're really good at this math and science stuff, you ought of think of something beyond the farm," or was there any encouragement to do so?

Gelsinger: Yes, there was. A couple of the teachers, the vo-tech, electronics instructor in particular. I want to tell you at the time it was more, "Be an electrician." You first started touching, and doing wiring and turning lights on, and some of your-- you know, everybody has to program a little traffic controller, and those types of things. So some very rudimentary things. But definitely some encouragement there. And I was sort of this wandering kid where I always thought I was going to be a farmer, and what's a farmer without a farm? Right? You know, lost and confused. Every step of the way into electronics and electrical and eventually touching computers, it's like, "Yeah, this is the pathway." I was pretty excited about it.

Brock: Yeah, your father had, there were ten kids in his family. How many in yours?

Gelsinger: Yeah, we're four. So we were smallish in the family. An older sister, I was the oldest son, and then two younger brothers. And in a farming family, the first boy, I was special. I could do no wrong in my parents' eyes. And this idea of moving to California. "Boy, you can't let your son--" that was like, "In California they have earthquakes, cults," I mean, you're crazy out here, right? So there was a lot of those negative premonitions, from a very Germanic, very non-diverse community. And that's just what it was.

Fairbairn: Did any of your siblings leave the area? Are they all--

Gelsinger: They're all back there. So I am the black sheep, and when I moved out to the east coast for three years, I was back in Boston with EMC. And the family was sort of like, "Ah, the black sheep is coming home!" And then I moved back here to become the VMWare CEO, and it's sort of like, "He's hopeless." <laughter>

Fairbairn: What do you do?

Gelsinger: Yeah, and you know, really some of that is right. I had to explain my mom and dad was pretty [confused]-- eventually they got the idea of what silicon was, right? You could show it, you could touch it, you could feel it. "Systems, okay. You know, put lots of silicon things and put sheet metal around it. Okay, I get it." "Software? Huh? What are you talking about?! People pay you for that?! What do you--," <laughs> So it was pretty hard to get them [to understand] this idea of going from silicon to systems to software? "What on earth is that?"

Brock: Was reading a big part of your early life? Were you a reader, or were you more outside? I mean, those aren't a natural contrast, but was reading a big part of your life?

Gelsinger: Yeah, it was. Anything that was engineering oriented, and some of that was reading. I wasn't a big fiction reader, but historical reading, clearly, super good in math and science, physics, those type of subjects, I always did well at. But I was pretty outdoorsy, as well. I spent a lot of time working on the farm.

I was a hunter with all of my relatives. The first day of hunting season was a school holiday. So we were all there, right? That was probably more important than any other national holiday. The first day of hunting season. You never missed it. And so all those outside things were really important, and I'll just say it was a very diverse and wonderful upbringing that way.

Brock: Well, maybe we could talk-- I know that your involvement with religion has been a major theme throughout your life to this day. Was it kind of a major theme of your household growing up?

Gelsinger: Yeah, it was a very traditional religious environment there. You went to church on Sunday. Because that's what you did. Right? It was part of the social fabric of that community. As I sort of joke, I was baptized with full knowledge of what I was doing when I was six days old. <laughter> Because that's what you did. And I was President of the Youth Group, because I had sort of some natural leadership skills. But I really came to become a Christian when I moved to California. Which is sort of exactly the opposite of what you would have expected. And you know, as I like to joke, there was two good reasons to go to church when I was a kid. One was to meet girls and impress their mothers and grandmothers. And the other was not to get in trouble with dad. So well, that's pretty good. Meet girls, impress mom, and so when I moved to California, what did I do? "Hey! I'm going to go to church to meet girls!" And sure enough, the first Sunday at church I met Linda. Who eventually became my bride. And really, I'll say the young adults groups there really sort of brought me in, and I really came to that point, actually, fairly soon after moving to California, in February of 1980. I moved here in October of 1979, and became a Christian and really in many ways, that just became this fundamental shift of my value system, who I was. And a few months later, I felt like I was being called to become a minister. So here I am, and I'm so excited about computing and microprocessors, and working at Intel. I'm good at it. I'm now at Santa Clara working on my bachelor's degree. And, just everything is going great. It's sort of like, "God, what are you talking about? Become a minister? You know, that's not what I want to be!" And so I wrestled with God and argued with God for a few months over this, and came to a point where I said, "Okay, I'll-- if this happens, right, I will become a minister." And immediately, when I sort of have gotten to that point of giving up if you would, or really submitting myself, the answer was, "Be a workplace minister. You're a fulltime minister in the workplace." And that just changed everything about my perspective. Because a few people as Christians, or of people of faith are called to be fulltime ministers, or fulltime clergy, or fulltime Imans. Most of us are called to express our faith in the workplace, in the school place, in the marketplace, in the home place. And that's really been what I view as my calling since then. Is to be a great leader of my projects, technical contributor, so on, but also then being able to express my faith in very appropriate ways in the workplace, as well.

Brock: And that was, you had that sort of clarity as a young person at that time. I mean, you were as-- was that something that immediately came into focus for you, or did you develop more clarity about that over time?

Gelsinger: Well, I'd say somewhat both. Where it was like Day 1, boom! It just changed my perspective. This idea of being a workplace minister, and I really started to really resonate with that and what that means and how you really view yourself as working for God as your CEO, even though you're working for Intel. And at the same time, it was clearly something that, boy, I developed more and more

understandings as I became more both mature in my own faith, as well as a more meaningful leader in the workplace and as I became a manager. Well, what does that mean? And how does that look like when you have people working for you? What's inbounds, and what out of bounds. And as you became a second level manager, as you became a director, a vice-president, a senior vice-president, and now a CEO, how do you express ones faith in the most appropriate ways? And so it clearly is something-- that's a learning area as well.

Fairbairn: Right.

Brock: Fascinating. Well, maybe we could jump back a little bit in time.

Fairbairn: Yeah, we need to go back to Lincoln Tech.

Brock: Yeah.

Gelsinger: Yeah.

Fairbairn: And you sort of stumbled into it almost. Were you excited about going? I mean, was it-- or is it just something new that you said it wasn't intentional, it doesn't sound like.

Gelsinger: When I won the scholarship, it was sort of like, "Oh!" right? And so we went to Lincoln Tech, and I said, "Boy, you're supposed to be a senior when you take this, not a junior." And I said, "Well, can you keep the scholarship for a year till I graduate?" And they said, "No. You have to redo it for next year. So I would go back to my high school guidance counselor. And they said, "Well, you could take it. You've just got to get some more credits to backfill, and you can still graduate from high school that way." And generally, as I think a lot of junior/senior in high school, you're sort of getting a little bit bored, a little bit not challenged, and I was bright, I was starting to get into a little bit more trouble than I really should. So the idea of jumping into college early was like something that sort of emerged out of that, and I got super excited about it. And you know, some of that, of course, is moving out of home. You're sort of in some ways e a little bit trepidatious about that. But also excited about it! Moving into college when you're literally 16-years-old, moving away from home and so on, and I was a pretty independent soul. And I started taking classes at Lincoln Tech, and sort of zoomed to the top of the class. And it was like, "Wow! I'm pretty good at this stuff! I really like it. Really enjoying it."

Fairbairn: What was the focus of the curriculum?

Gelsinger: It was largely to be an electronics technician. And you know, with that, so your basic engineering technician type of work. Everything from Ohm's law, to basic programming skills, to fixing TVs and radios and so on. You know, at one point I worked in a radio and TV station, fixing TV equipment. I even did a little bit of part-time D.J. work at nights. So I was the on-air voice for WFMZ in Allentown, Pennsylvania, giving the weather and news at midnight, 2:00 a.m. and 4:00 a.m. on Friday and Saturday nights. Hey, as a college student, you'll do just about anything to make a few bucks at the time. But really, really found enjoyment in the technology itself. And every class was just sort of like opening

another layer of the onion. And the deeper I got into it, the more I enjoyed it. I remember some of the first programming I did, it was like, "Oh, wow!" I was really quite excited about that. Building up some of your basic control systems, as well. I remember my-- one of my projects was building a single board computer there, which was quite-- I entered it into a competition and won, one of the local computer competitions. But they couldn't decide if I was a college student, or a high school student. <laughter>

Fairbairn: Which category to put you in? Huh?

Gelsinger: Yeah, it was a little bit humorous that way. It was before some of the robotics competitions, or some of things that might be today. But you know, I certainly loved it, and when Intel came recruiting, it was like one of seven or eight interviews that I received at the time, and everything else was East Coast, except for Intel.

Brock: So what was the year you entered and graduated?

Gelsinger: So I entered in '78, right, in I think it was January of '78. And then I graduated in August of '79. So it was a two-year program, really a 20-month program, and I accelerated it. Of course, I was always trying to be this over-achiever. So I finished the program in about 18 months, and Intel came recruiting.

Brock: Now Intel has recruited there before? I mean, how did they find Lincoln Tech, and--

Gelsinger: Yeah, yeah, it was very exciting. Ron Smith, who was a pretty-- you know, he was a Ph.D. from Michigan. He was born and raised in Lancaster County, Pennsylvania. And there was sort of an industry-wide shortage of technicians at the time, and Intel was in one of these hyper-growth phases. So he volunteered to lead the first-ever recruiting to Lincoln Tech. It's the first time they ever came there. And as he's-- and he interviewed a dozen of us. And he was going to-- he was interviewing a dozen. I think he was going to invite eight, and they were hoping to hire like five or six from our class, and I was Number 12 of the interviewees to Ron Smith. And Ron wrote this in his paper after he interviewed me. He said "Smart, aggressive, arrogant. He'll fit right in." <laughter> And got the invitation to come out and interview.

Fairbairn: Were there others that came also?

Gelsinger: Yeah, there was others that came out and interviewed, and three of us were hired—no it was four of us were hired by Intel. Two of the other guys became my roommates. My first roommates when I moved to California. You know, we thought we were pretty rich when we got those job offers from Intel, and then we saw what it cost to live in California, and we realized we were pretty poor. So we got a house together, very close to Intel on Bowers Avenue, not far from Central Expressway, so sort of right in the middle of the town there, and it really was, I'll say, this period of life we're looking back on, "Would I let my kids leave home at 16-years-old? Would I let my kids move to California at 18-years-old?" And I look back on it and say, "No," but yet my parents gave me the freedom when I was super independent soul to move out here and begin a journey that I thought I might be in California two years. You know, I'm an East Coast boy. And then it was 30 years later.

Fairbairn: So what was magical about Intel? What persuaded you that you really needed to do that?

Gelsinger: Well, a couple of things that really impressed me, and David A. Brown was my first manager. It was actually I was either going to work in Ron Smith's organization or David's organization, as a QA Technician. But the thing that separated the Intel offer from all the other offers is they gave me tuition reimbursement to finish my bachelor's and continue school here. And gave me a super flexible work schedule to do that. So basically, as long as I worked 30 or more hours a week, and got B's or better, they would pay for my bachelor's/master's/ Ph.D. work. And my parents were both eighth grade one-room schoolhouse educated. So and they just pounded into us from early age is, "Go to school, go to school, get your Ph.D. ." And you know, I don't even think my mom knows what a Ph.D. is. But it was one of those things, "You need to go get your Ph.D." So I was pretty committed to continue past the two-year degree from Lincoln Tech, and Intel clearly gave me the best offer to make that possible. And so I moved out here in October of '79, and I started at Santa Clara University in January of '80 to start working on my bachelor's. So I would go to classes in the morning. And then I would work in the afternoon and evening and study at night. And was working fulltime and going to school fulltime. And it was a pretty unique experience, and then I continued into Stanford, being paid for by Intel. I remember my first Santa Clara tuition bill, I had no money, right? I'm this poor farm kid. So I put it on a credit card, and I lived in fear that I wasn't going to get a B or better, because I couldn't afford to pay back. And the idea of carrying debt on my credit card was just this, "What will I do if I can't make the payment at the time?" So it caused me to work a little bit harder in that first--

Fairbairn: Sounds like you got A's on all of those courses.

Gelsinger: Yeah, yeah, it was one of those, I remember I got a B on my first midterm exam on one of my classes at Santa Clara. And I was just, <gasps>, "I'm going to fail! I'm not going to be able to pay! They're going to fire me! I'm going to have debt! What am I going to do?" But it certainly motivated me to study hard and work hard in those first classes, but that was part of what brought me to Intel. It was just the flexibility to go to school. I was actually excited about working for a microprocessor company, and that was so intriguing at the time. And you know, the microprocessor, I mean, it wasn't a new invention, but it was still very new in terms of popularization and moving into the industry. So it was a thrilling time.

Fairbairn: So it was in 1980, you started Intel in--

Gelsinger: '79.

Fairbairn: Right after you graduated in August.

Gelsinger: Yeah, August, '79, began at Intel in October of '79, started my bachelor's at Santa Clara in January of '80. And worked as a technician for two years at Intel and was really a QA Technician for microprocessors. And you know, that was my first sort of taste of really getting close to what I considered the holy of holies. A microprocessor of Intel.

Brock: What did you really learn in those two years? I mean, did you learn some key lessons about the company, about the technology? You know, what was that first two years of working like?

Gelsinger: Yeah, to me, it was just-- I was like a sponge coming out of the desert. I mean, just every-- just learning so much, and so excited. And you come with a work ethic from the farm, right? And to me working on the farm was, you know, you really learned to work hard. And to me, if the horses weren't kicking me, the cows weren't biting me, and I wasn't covered in hay dirt, "Hey! This is pretty good!"

Fairbairn: Good day!

Gelsinger: Yeah. And boy, it's an air-conditioned building. You get paid overtime? Can you believe that? Right? I was just like, "Wow!" I've never been paid overtime in my life before as a farm kid. So I was just manic. I was either studying for Santa Clara, or I was working at Intel. Linda and I joke that we squeezed a year worth of dating into three years. <laughter> Because what did I do? I worked and I went to school and I was loving it, and I've always had a sort of a natural metabolism. I slept about five hours a night, so I had more hours per day than most other humans do. So I was either working or going to school, and you know, if I wasn't in school, I was setting overtime records at Intel, because I loved it! And so the work ethic aspects, but also the first really learning some of the basics of programming, learning how to build systems. I remember one of my first experiences as a reliability technician at Intel was, "How can I go work with the Design Team?" And again, the Design Team, oh, they were like the high priests, right? It's like, "Oh! That's really unique." And us lowly technicians and QA, you know, if we'd walk into the Design Department, it was just like we were walking up the hill to Jerusalem or Mecca. It was like it. And as the Reliability Technician, I was trying to run reliability experiments on microprocessors, how to make them reliable, how could we make that whole reliability process? And that's sort of what's started to get me sucked into the Design Team. And I started to do some reliability testing work about how can we design more of the reliability functions into the microprocessors? Which in 1980 and '81, that was super early for that whole idea of design for testability, and built-in self-testing kind of circuits. You know, none of that was done at the time. So I began to study the whole topic of built-in self-testing. The first paper I ever had published was a built-in self-test thing for microprocessors. And as I started to work with the Design Team, they were, "Hm, here's this bright young kid, what's he doing hanging around here?" And I'm trying to get them to do these reliability tests and functions, and build in these capabilities. And they're sort of on the side, they're teaching me about chip design. Chip design. Oh, my gosh, I truly had gone to heaven when they were starting to teach me about chip design. I just didn't think the world could get any better than that, because I was designing the microprocessors of the future! It was just like, "Boy!" I really was thrilled by that! And the idea of being able to be part of the Design Team for the 286, then I was Engineer Number 4 of the 386, and it all started by saying, "How can I start building self-tests into those chips? How can we start building some of the reliability mechanisms into the chips themselves as opposed to just testing them after the fact, start to build the capabilities directly into the chip?" And the Design Team just sort of sucked me in, and I had some pretty good computer skills and they were sort of saying, "Well, why don't you run our UNIX machines in your spare time, as well, so we can get out of those terrible IT folk?" So I was running a UNIX instantiation on the IBM Mainframe for the Design Team. And they all hated the CMS or the VAX environments that we had, because those were run by IT, so I essentially became the system admin for the entire design team as well, and they would teach me how to design

chips in their spare time. And I'm going to school to learn how to design at the same time. It was really a magical period in that way.

Brock: Could you talk just a little bit more about that reliability and the self-test in that engineering? That connection between let's say logical design, and the physical reliability of these electronic components? I'm not sure that everybody reading or watching will understand that.

Gelsinger: Yeah, and imagine, even a chip like the 386, 282,000 transistors, approximately. And it was encapsulated in a package that, if I recall correctly, had 64 logical pins associated with it. So how do I wiggle 64 things to test 282,000 things? Because when you're testing a chip, you know, do all 282,000 work? And you can imagine that it was designed to functionally do things, run busses, signal addresses, be able to stop/start, those type of things, but how do I make sure that there aren't defects laden inside of that 282,000? And the 286 had 127,000, the 486 was 1,204,486 transistors. I added a few to make it round.

Fairbairn: Had to be the 486, right?

Gelsinger: Absolutely, I added a few to make it round to that. <laughter> But these ratios of the amount of external visibility versus internal potential defects was an extraordinary problem and how do you wiggle those pins to be able to make sure that all of those transistors are functioning properly? So this whole issue. And because as Moore's Law was now really in the sweet spot of Moore's law, we were leaping from thousands to tens of thousands to hundreds of thousands to millions to tens of millions of transistors, and yet, your IO connectivity was going up maybe generation by generation by 20 percent? So that ratio of transistors to signaling pins to test things was becoming extraordinary imbalanced. And in fact, one of the papers I wrote was that the cost of test was also as that problem was becoming worse that I basically demonstrated that just if you looked at the cost of test per chip, that the testing cost per chip was going to exceed the cost of the chip in that period of time. So that led to this whole topic, and it was a very active conversation at that period in the early '80s in the industry. In fact, there were entire conferences dedicated to built-in self-tests. How can we start building that into it? And it was a humorous, because the test guys at the time, and this is the Teradynes and those kind of companies and Schlumbergers and so on, were the testing company at the time, you know, they were drooling! "Wow! These tests costs going up," because you know, we--

Fairbairn: More machines.

Gelsinger: Absolutely! Right? So I was the anathema, right, for the testing guys, because I said, "No way are going to let the test cost per chip become more than the cost per chip! We're not going to spend more testing it than we are on the package, the silicon, etcetera, associated with the chip." And that became the nexus of this idea of built-in self-test. And so we started to build. "Instead of having a million functional transistors, let's take a few percent of them and dedicate them for testing." Because even a few percent of a million was far in excess of the number of pins. I could dedicate 30, 40, 50,000 transistors to testing, you know, just three percent of the transistor count. And if I could self-test, or create automated test patterns or other mechanisms that could lower testing cost by a factor of two, I more than paid for the

three percent that you might have put into transistors inside of it to test it. And those transistors that you put in, all of a sudden became very, very powerful mechanisms. You do diagnostics, you do debugging, all sorts of other things emerge from that, as well. Which today, these are standard functions of chips today, the idea that you can stop them, you can debug them, you can read out state associated with them, you can run testing patterns. In those earliest days, that was one of the breakthroughs. My first achievement award at Intel was built-in self-test in the 386. My first patent was in that area. My first articles that I had published are in that area. And in fact, I resigned from Intel at one point to go finish my Ph.D. at Stanford, and it was going to be in this area of self-test, test automation, formal proof of validation, formal proving of logic trees and complete test coverage, etcetera.

Brock: Wow. So--

Fairbairn: So-- I just wanted to finish up on-- so I understand working with the design team to work on the testing problem. But in the quality problem, you know, the reliability problem, did you also work with the process engineering folks?

Gelsinger: Yes.

Fairbairn: And identified certain process quality problems and that sort of thing?

Gelsinger: Yeah, test coverage problems and quality problems were often very hard to separate the two. You know, at that level, and a lot of those things were, ended up with funky transistor structures that would be susceptible to certain process defects or process weaknesses as well, so I interacted a lot with the process technology folks, as well, just were wonderful days, and we did ion-beam editing of the mask of the 386. The first time that ion beam was used to actually edit the mask, so you didn't have to go through a full tape-out radical production process. So we went in and did it. You know, specifically to address some reliability issues in the fab. So I got bunny suited up, I'm in the fab, we're running these etch patterns and figuring out how to address of the early reliability issues of the 386 exactly that way. So it was just delightful days. And the first 386s were manufactured over at the Livermore fab. We had the Fab 3 for Intel was over in Livermore at the time, and I remember I and another guy was written up in the *Wall Street Journal*, the midnight ride, Chip had a Mustang convertible, and we went over with the top down listening to the Rolling Stones. We came back with the top up holding the first 386 wafers! We thought it was like the wise men. It was like gold, frankincense and myrrh. We had the first wafers. And I remember everybody anxiously waiting, because the debug team was ready to jump on these and get started on it. And we, of course, were young. I'm 23-years-old at the time, something like that, So we decided to play a prank on the team, and we had some wafers that were of other chips, and we bring them into the conference room. I trip, we drop them on the floor. We sort of ended up falling on them, breaking the wafers. <laughter> And so on. And I remember Jan Prak, one of the Design Managers on the 386, he starts getting down on the floor and trying to find a full complete chip out of one of these wafers that were sort of mashed up, and you can just see him, he's like almost ready to cry, right? "Oh, my poor 386!" And then we bring the real wafers out and just everybody had a good laugh. Even though I thought he really was going to have a heart attack at the time.

Brock: Oh, what was I-- oh, it was just so with this devoting some of the transistors to self-test, they are in essence, you can interrogate them so they can check all the other transistors. Is that, roughly speaking, the case?

Gelsinger: Yeah, and some of the transistors, you get very clever for how you can insert the transistors. And the first things were like some of them were like big structures, like memories, PLAs, Programmable Logic Array. How could you make sure that those were entirely tested? So we did pattern generators and I remember linear feedback registers using polynomials that you could actually get very quickly that you could generate patterns and be able to test results and be able to essentially produce a hash of the results, and be able to verify it. In other cases, it was to actually give you visibility, where you could control state, you can inject things into certain portions of the chip. You could see what it's really doing inside. You didn't have to guess. And again, in the microprocessor you might be many clocks later until you can actually see some of the state. So if we could pause the microprocessor and then actually interrogate the state directly, very powerful, not just for testing, but also debugging the chip. Or many of those functions became software visible over time, where software writers could use them as well and be able to interrogate and say, "Oh, how did I get to that state, or that wasn't want I thought that protection register should have looked like." So those basic mechanisms became useful not just for validating the chip, not just for the reliability of the chip, not just the debug of the chip, but even turned into software visible functionality as well.

Brock: Hm, that's fascinating. Well, could you-- so as you're kind of moving into the design group for the 286, could you describe sort of what that group looked like? You know, how many people was it? What sort of tools were they using? You know, how was that group designing the 286?

Gelsinger: Yeah, and it was this wonderful time of the design phase, where a lot of the computer-aided design tools that we just take for granted today weren't invented yet. So you still were, you know, we're past the rubylith phase, so you could be putting polygons and you had mask designers who were laying out polygons, you could put those in the libraries and structures. But a lot of it was still hand-done, right? There was a loose correlation between the logical design and the physical design. You had to do a lot of work to correlate those two and it was really before RTL [register-transfer level] was a formal mechanism to describe chips as a higher level language. It was before HDL [hardware description language], or VHDL [VHSIC description language] was invented at that point. You largely were writing fairly rudimentary code to mimic what you saw in the logic. And then you had a process by which you take the logic and actually draw schematics. And those schematics would be handed to a mask designer who would then be translating those. And then you'd have tools that would be helping you, make sure that what actually ended up in the logical design, right, into the schematic design, into the physical design of the chip. And then you had a whole separate tool set associated with circuit integrity, circuit speed as well. These were largely manual processes, and so a lot of my early successes in the Design Team were bringing CAD tools, as we would know and love today, into the design process. And some of these were super rudimentary processes, just that you could essentially be capturing logic traces and doing correlations with RTL runs, right, where you could be running very basic patterns that the circuits would be functioning properly. One of my early awards from Intel was I helped to invent the first-- we called it IHDL, Intel hardware description language. We wrote the first HDL that was actually a synthesizable

language, where you could actually synthesize logic. So we could eliminate a lot of the errors that might be introduced by translating a higher level description into logic. We also helped to create the first synthesis system, as well, and that was part of the 486 where we actually synthesized it into logic, not just validated it as well. But today you look back on that and you sort of say, wow, you guys were using your hammers and chisels out of stone compared to the tools that we have now. It was quite rudimentary in comparison. You were wrestling with computers to get them to stay up and run because many of these jobs had enormous memory spaces, much larger than computers could run at the time. And one of my favorite stories in that phase of my career was actually late in the 386 design where I was in charge of taping out the 386, which is assembling that big database. And today it's a trivial database but at the time it was a massive database for the size of the computing set that was available. And I had to give an update to the executive staff of Intel. So here I am, I'm 23 years old, something like that at this point. And I'm presenting to Gordon Moore, Moore's Law, Robert Noyce, the integrated circuit, Andy Grove. <laughs> They're in the front row. Here I am, I'm updating them on the chip assembly of the 386 as that project is going. And what I did, I chewed them out because the computers were not stable. So here I am this little 23-year-old precocious brat who is chewing out the gods of the industry Andy, Robert, and Gordon, "You have to fix these computers if I'm going to get my chip out the door." And it was like wow. I was working so hard. I was so busy and trying to get this thing done and computers were crashing and jobs weren't finishing. And a few days later Andy he calls me up. The phone rings. I pick up the phone. And I said who is it? And the voice comes back, "Andy." Andy who? And he says, "Andy Grove." I'm down here and I just sort of stammered and shocked and so on like that and he starts shelling me with questions. "I was impressed by your presentation the other day." Oh, I thought I chewed you out but, anyway. <laughs> "What do you read? What are you studying? What's your next career goal? What's the next job that you want?" All of these career and aspirational questions. All I wanted to do was get my chip out the door. And he said, "Those are lousy answers. Be in my office in a week with better ones." And that began a mentoring relationship that lasted over 30 years with one of the gods of the industry. It's one of those most unique experiences that emerged out of that early period of my career of wrestling with CAD, before we had CAD tools creating them and computers and so on began a relationship that I consider one of the most important of my career and life.

Brock: So you were working in this QA responsibility that's interacting with the design team. Who moved you over? Did somebody call up your boss and say, "Hey, we need him over here." Or you just started spending more time over there? How did that process happen?

Gelsinger: Yeah. At the end of the 286 project Jim Slager, who was a well-known guy at Intel at the time, one of the early design managers, he was in charge of the last steppings of the 286. And he and Jan Prak became the first two design leads, logic guy and a circuit guy at the time. Jim was the logic guy and Jan was the circuit guy. And so they were finishing up the 286. I started to work with them. And they saw this young, aggressive, arrogant, smart computer savvy guy and they just sucked me into the design team. Literally one day I was the QA technician working on the 286 and the next day I was a dweeb design engineer entry-level, didn't know anything about design working for Jim Slager and he brought me over to be part of the 386 design team. And I was engineer number 4 in the 386.

Brock: I was wondering there's an interesting relationship between-- well maybe you could characterize-- there's another big microprocessor project going on at the same time, the 432?

Gelsinger: Yes. Yes.

Brock: Which was, as I understand it, sort of like the computer scientist's dream sort of…

Gelsinger: Yes.

Brock: As we know in retrospect the 386 became so critically important for Intel as a company. Could you talk about the story of the relationship between those projects and how it appeared from your vantage point?

Gelsinger: Yeah. It was just a stunning time in the industry because the 286-- you know, at the time, this idea of instruction set compatibility was a fairly nascent idea where sort of there was limited outside of some of the things that IBM had done pretty much every machine was designed with the new instruction set. You're sort of jumping from 8-bit to 16-bit to 32-bit and so on. So this idea of instruction set compatibility and the business plan-- there's a formal business plan for this at Intel that the 386 was a sort of a stopgap that would have a fairly short lifetime and it would be usurped by the grand new architecture of the 432. And we were working on the dumb machine, the 386, and all of the really smart architects and designers were working on the cool machine, the 432 and that was up in Oregon, was the center of the 432 development. And we were the dumb folk working on the 386 down in Santa Clara. And this was tribal warfare at its finest. The teams hated each other. It was incredible the battles that were going on between us and we'd fight over computer resources and ideas and whatnot. I remember one day Jim Slager comes into my office and he hands me the book, *The Soul of a New Machine*[1], the Tracy Kidder book. And he says, "Read it. You're living it." And we were. Because if you remember the book, the Pulitzer prize-winning book, it was the two design teams, the North Carolina design team and how they were competing with each other and which one would win. And sort of day by day it became visible that the 386 was more prominent. That it wasn't a real stopgap. Well, it's going to be a little bit longer. And the 432 was a complex design so it was slipping. So the role that the 386 was playing was getting bigger. And we were told that well to help make the transition to the 432 easier you need to make the buses the same so that we can build boards for the 386 and then we can drop the 432 into the same buses. So then we had the bus wars that we were fighting because we had to have these-- and the bus it was a transactional bus architecture for the 432 which, you know, a basic address data response bus in the 386, we looked at this like this is like from Mars. Why would we put this complexity? But the management team decided this was what you had to do. So we had this little stealth project going on and the bus unit was like we made all of our interfaces into the bus unit very clean and precise so that we could get rid of that complex bus because every day their bus logic got bigger and bigger and bigger. It was consuming more and soon the bus unit from the 432 that we had to take was almost bigger than the rest of the chip. So we had this little stealth project going on the side that was a nice simple address data bus architecture that we had to mine a pipeline data bus architecture. And at some point we just stopped working on that one,

[1] [Interviewee's note] Kidder, T. (1982). *The soul of a new machine*: Penguin.

“When you guys get it done send us the design files and we’ll think about how we plug it in,” but we agreed on the interfaces to go do it. And the project just kept getting more complicated because the architects were these object models and security models. It was architects gone wild. And really as history has now shown it was just so extraordinarily complex that the design never came to fruition. There was the big partnership with Siemens that Intel formed at the time to bring systems together associated with this. And it’s sort of like every day the project got bigger more complex and later. And inside of that the importance of the 386 became more and more important as that became-- and all of a sudden the role of the 386-- so that was sort of battle one of the 386 versus the 432. But then battle two became the IBM battle because, “Thirty-two bits, wow, that's much too powerful for a personal computer.” IBM had decided that 32 bits was only suitable for minicomputers. So at that time this is before the PC compatible era had begun. So the second battle was who was going to introduce a personal computer using the 386? And it was before Compaq was anything. Dell wasn't even an idea yet. And all of this era of the industry is waiting on the IBM. And what is IBM doing with the 386? Nada. They are stalling because they had good profits at that 32-bit level and, “Everybody knows that 32 bits is much too powerful to put on some desktop running some personal computer operating system like MS-DOS. Clearly 16 bits is enough forever for that.” And so as we’re working on the 386 we're starting to-- we’re taking trips to Houston and trying to get the Compaq guys and taking trips to Boca Raton and trying to move the IBM guys. And every visit to IBM seems like we're just mired into the sand pits. Where are the designs? Why aren’t you guys giving us feedback? And so on. And then, of course, the day that defined the industry was when Compaq said, “Hey, we’re going to introduce the 386 into a personal computer. And we’re not going to wait for IBM.” And as that started to gain momentum, IBM rushed to be the first one to market with the 386. And then the doors were wide open. The personal computer industry was unleashed from the IBM Boca Raton, we’re going to control it, manage the industry, we own all the buses, the intellectual property, and the race began into tomorrow. And it really was a stunning period. And looking back on it, you had just no idea that you were writing history. You’re getting on the plane and going to Houston and they're coming along but we’re all “What's IBM going to do? Can we actually fork from IBM? What if they do it a different way? Are they going to sue us?” And all these crazy things are going on at that period of time. And then out of that came the personal computer industry with open interfaces, with standard buses until they attempted to get things back under control with microchannel because, of course, we can do a better job than those stupid PC guys and that became the second great battle and out of that came PCI bus and the industry standards that weren’t controlled by IBM and Microsoft innovations and software and the whole Wintel generation just explodes in innovation and creating a new industry standard.

Fairbairn: Can you put some timelines on these? We sort of rushed through. When did you join the 286 team? And then when did you join the 386 team?

Gelsinger: Yeah, I joined Intel in ‘79. I joined the 286 team in early ‘81. So not quite two years as a QA technician sort of kicking around the design team. And then the 386 formally got underwear underway in late ’81. At that point that was about a four-year project.

Fairbairn: That was very quick. You sort of moved through those very quickly.

Gelsinger: Yeah. Yeah. It was less than two years that I was into the 386 team and from my time starting at Intel. And truly I thought I had just gone to heaven when I joined the design team. It's like designing the future microprocessors of the planet.

Fairbairn: Were you doing logic design? Were you doing architectural design? Were you doing-- what level of design were you involved in?

Gelsinger: I was mostly doing logic design, right, was where I was but late in the project and being one of these over-achiever manic worker kind of guys, I had the D unit the data units. So I had the ALU. I did the ALU. I got a patent on the non-regular-carry look-ahead adder on the 386, which was a big circuit design problem and got that done. And then I did the I-unit, the instruction decoder unit. Then I did the test logic for the complex protection modes of the 386 as well. And then I was in charge of chip assembly and tape out for the 386. So I did some circuit design. I did some physical design but most of my work was in the logic area.

Fairbairn: So it was at the end of the 386 and the sort of tape out blizzard that you had this encounter with Andy and things took a dramatic turn.

Gelsinger: Yes. That was sort of the career defining moment and we launched the 386 in ‘85. So this was late ‘83, early ‘84 when we’re working on that tape out phase of the chip and had my first real encounters with Andy Grove and my career began to take a very different turn than I ever expected as a result.

Brock: I can imagine that, you know, maniacal work ethic and that sort of public confrontation about a problem that you're having both of which would've appealed directly to his style of operation.

Gelsinger: Yeah. And Andy was one of those guys where he just didn't care if you were a head of state or an entry technician. If you were smart and had data he was interested in interacting with you. And, as I said, you had to be smart. He had no time for fools. You had to have data, defend your perspective, and a point of view. And if you could do that he didn't care what level of the hierarchy that you were. And it was just incredible and he was intensely focused. And to be in a room with Andy and be quiet was one of the worst things that you could do: “Why are you consuming oxygen when I could have somebody else in that chair who actually was contributing with data and points of view to the conversation?” He was very confrontational-oriented because he felt in that conflict came better ideas, better understandings. If he’d ask you a question that you couldn’t give a good answer to then I know the next five questions he’d ask you if you were a waste of time because if you couldn’t defend that one, have you really thought about, worked hard to understand the next three or four? So to not have good answers for two or three questions in a row to Andy, oh my gosh. <laughs>

Brock: So how did that then morph into you then became-- what was your position at the 486? What was that next step? And how did Andy affect that?

Gelsinger: Yeah, so I became, and the design team sort of birthed me, into being the first architect of the 80486. So I was architect number one, sort of that early product planner architect design phase basic micro architecture. So I was engineer number one in the 80486. Became part of the design team as we started to grow the design team. Other people started to come along the design team. And then I resigned for the second time from Intel.

Brock: What was the first time?

Gelsinger: When I was in the middle of the 386, I was going to go work in my Ph.D.at Stanford.

Brock: And how did that turn around?

Gelsinger: Yeah. And that one was-- it was interesting because I resigned in frustration that the design management and teams they weren't taking the steps they needed to do to organize us for success. I was going to go work for IBM and they were immediately going to put me as part of the Ph.D. program and they had a very lucrative program to go do that. So as a result of that, the 386 team was restructured. There's a number of things they just had to do if we're going to get the chip out the door and be successful. So my first resignation sort of resulted in the reorganization of the 386. And, obviously, we got the 386 done. So that was it. And then I left the 486 to-- I was well in my master's program now at Stanford and, again, my mom said, "Get your Ph.D." So I was going to go get my Ph.D. and was ready and applying for it and John Hennessey was going to be my thesis advisor at Stanford. So he was my master's advisor and was going to become my Ph.D. thesis advisor at Stanford. So I resigned from Intel and I was an early architect and the project was going well and getting underway. And Andy Grove came to me and he says, "You can go there and learn on a simulator or you can stay here and fly the jet." And he offered me the job of being the 486 design manager. So here I was at 25 years old, 24-25 years old. I had been an architect in the 486 for a year. I was ready to go to do my Ph.D. program at Stanford and Andy made me the design manager for the 486. And at that point in time nobody on the 486 team was younger than me. There are 100 engineers working on it. I'm the youngest person on the design team which, in retrospect, when you put the youngest guy, the least experienced manager in charge of the chip but Andy really saw that leadership. He had been mentoring me for a couple or three years at this point in time, and he put me in the role of the 486 design manager which I went from being the first architect of the chip to becoming the design manager for the chip and really hiring, firing, assembling the team, laying out those strategies and plans to bring the 486 to fruition. And part of the reason that Andy made me the design manager was-- became the second battle. We had the 432 battle. The next battle became RISC/CISC. Right? And if you go back to '85, '86, '87 this is when the RISC doctrine was emerging in force in the industry. CISC versus RISC, this idea of complex instruction flows versus highly optimized simple pipelines. Move that complexity into the software systems that fed the pipelines. And there was a whole flurry of RISC chips that were emerging in the industry.

Brock: Something from John Hennessey, your advisor, right?

Gelsinger: Yeah. Well, we'll get back to that one in a second, the extraordinary story there. So this is emerging but we also had one at Intel called the 860. Right? And there's the 860 RISC chip because

that's the future. That clunky x86 thing you're not going to be able to keep up; all of these graphs were showing how CISC chips were going to flatten out and you weren't going to be able to keep up with Moore's Law on all those systems. So, once again, I was working on the clunky old boring compatible chip. But Andy and I were convinced-- and at that time the dogma of instruction set compatibility was deeply ingrained inside of me. And part of the reason when I left Intel it was like hey, everybody wants to go build these RISC chips. You're not investing in the 486. That's clearly the right thing to go do. I'm going to go finish my Ph.D. at Stanford. My mom wants me to get that done anyway. And that's why I resigned. And Andy, as I said, he makes me the 486 design manager in part because I believe so deeply that instruction set compatibility in the 486 was the future at that point. And so many of the other leaders at Intel at the time were convinced that the RISC dogma was correct and I was convinced it was dead wrong. And I remember I had a debate with John Hennessey where he said, “CISC chips will never break two clocks per instruction.” It was like this fundamental barrier that he had theorized that you can never get instructions per clock lower than that. There's just too much complexity in the CISC chips. And that RISC chips hey they’re going to go right down to one clock per instruction. This is going to be elegant. This is before any of the superscalar machines had started to emerge as an expectation. And when I had this argument with John I knew in the back of my mind that I was already running simulations at 1.8. And he's telling me it can't break 2.0. So now we had him as my master’s thesis advisor. I'm the design manager and architect of the 486. I'm about to enter the Ph.D. program. He's Mr. RISC because he's now starting MIPS Co. on his own. He's just published his book at this point the seminal computer architecture book is coming out, the first version of it. And we had a mutual friend that found out that we had Mr. CISC working as a student of Mr. RISC, the commercial versus the university, the old versus the new, teacher versus student. We had public debates of John and Pat. And Bear Stearns had a big investor conference, a couple thousand people in the audience, and there was a public debate of RISC versus CISC at the time, of John versus Pat. And I start laying out the dogma of instruction set compatibility, architectural coherence, how software always becomes the determinant of any computer architecture being developed. “Software follows instruction set. Instruction set follows Moore's Law. And unless you’re 10X better and John, you're not 10X better, you're lucky if you're 2X better, Moore's Law will just swamp you over time because architectural compatibility becomes so dominant in the adoption of any new computer platform.” And this is when x86-- there was no server x86. There's no clouds at this point in time. And John and I got into this big public debate and it was so popular. *Microprocessor Report*, you can go back and see the verbatim debates of John and Pat that we had was published in those archives. And we repeated the debate, I think, three other times that we had it. It was so popular at the time. Mr. RISC versus Mr. CISC.

Brock: So the claim wasn't that the CISC could beat the RISC or keep up to what exactly but the other overwhelming factors would make it the winner in the end.

Gelsinger: Exactly. The argument was based on three fundamental tenants. One is that the gap was dramatically overstated and it wasn't an asymptotic gap. There was a complexity gap associated with it but you're going to make it leap up and that the CISC architecture could continue to benefit from Moore's Law. And that Moore's Law would continue to carry that forward based on simple ones, number of transistors to attack the CISC problems, frequency of transistors. You've got performance for free. And if that gap was in a reasonable frame, you know, if it's less than 2x, hey, in a Moore's Law's term that's less

than a process generation. And the process generation is two years long. So how long does it take you to develop new software, porting operating systems, creating optimized compilers? If it's less than five years you're doing extraordinary in building new software systems. So if that gap is less than five years I'm going to crush you John because you cannot possibly establish a new architectural framework for which I'm not going to beat you just based on Moore's Law and the natural aggregation of the computer architecture benefits that I can bring in a compatible machine. And, of course, I was right and he was wrong.

Fairbairn: Now, as I remember, I had the oral history with John Crawford. And my interpretation of his remarks were that you found ways to embed a good number of the RISC ideas within the CISC architecture and so that you could close the gap that way; that it wasn't a pure RISC versus CISC. That you could actually take some of the core RISC ideas and embed them in the CISC.

Gelsinger: Yeah, very much so. And John deserves much of the credit -- John and I became partner partners in the late 386 days. We started to work together in the early days of the 486 and John was much more of a software architecture guy and I was much more of a logic microarchitecture kind of guy. So we had a great partnership. But the comments there are exactly correct because the belief was, you know, the 386 was about four clocks per instruction which when you're looking at an optimized RISC pipeline where they were demonstrating in simulation 1.5, 1.6 clocks per instruction that's a pretty big gap. But the 486 was very much a RISC-like internal pipeline where we could actually suck a lot of the complexity into the instruction decoder and present to the rest of the machine very RISC-like instructions in a very pipelined fashion. So we were sort of able, exactly, as you say sort of to pull the RISC ideas and implement them inside of the microarchitecture of the machine, never exposing the architecture or the instruction set externally to any of those optimizations that we did internally but in that be able to be compatible from the software perspective and benefit from some of those best ideas. And then the next-- and that was this optimized pipeline machine was the core of the RISC genre at that time. And then the next big step was moving into superscalar. We actually started to do parallel execution. And then the next big microarchitecture step was out of order execution. So those are sort of the bigger generations where you went from a true CISC machine to a highly pipeline machine very RISCy to a superscalar machine you started to do in parallel and then you sort of execute out of order instructions where you actually started to, essentially, out of order, superscalar retirement machines as opposed to single issue architectures. And that was sort of the major x86 architectural progression.

Brock: And Gordon Moore, at one time, described it as kind of the two approaches eventually converged and RISC became more complicated and kind of CISC became more RISC-like. And so really they approached one another. Do you think that's fair?

Gelsinger: Yeah. Because what happened was this idea of instruction set compatibility-- because as soon as the RISC guys put their mark on the wall they were all of a sudden bound by the same things. Well, they couldn't go changing as they went from MIPS code generation from one to two to three, they couldn't just willy-nilly be changing the microarchitecture because they had to run some of that old code. So all of a sudden they became bound by the same instruction set compatibility limitations. So they became more CISCy, exactly like Gordon said. And we were finding clever ways to hide the instruction

set complexity early in the machine, in the instruction decoder and so on and making the rest of the machine look more RISCy in that sense. So the ideas really sort of did start to meld together over time and what was an enormous religious to debate of the right way to build microarchitectures and that ‘85 to ‘92, eventually gave way to just an architectural battle.

Brock: It sounded to me like when you were describing the work you were doing on the 386 you could almost in a quick summary kind of say making developments and using software to design microprocessors. That was a big thrust of what you were involved with. When you kind of came back for the 486 and were in charge of that project was this the case where you were-- it was even more of that, of learning how to use software and big computing resources to really effectively design this a new microprocessor?

Gelsinger: Yeah, very much so. It was a continuation. It was one of these things where as I was a student at Santa Clara there was the EEs and then there were the software guys. And those were pretty separate disciplines. Right? You're over here using oscilloscopes and soldering irons and they’re over here programming. The disciplines were actually quite separate from each other. And what we were observing was that we were using far more computer science. In fact, all of our hardware silicon design time was spent doing software. So we were very quickly saying how can we start using more of those software for CAD tools and automation and tests and so on because were spending all of our time on the computer. When was the last time any of us were in the lab? Oh, okay. We need to go run a few circuit experiments. But the vast majority of time was being done on the computer so how can we do more computer automation? And one of the innovations late in the 386 design cycle was computer synthesis where we could actually do logic synthesis directly which allowed us to push the logic design much later in the design process; just give me a block and the chip and I'll drop in the synthesized logic gates into that. We became much more automated in terms of placement route technology for big blocks, for circuit automations. And so many of my early patents and innovations and awards at Intel were in exactly that area of how can we automate, synthesize, softwarize more of the chip design process itself? And, of course, that gave way to then extraordinary architectural innovations as well.

Brock: Does this relate, and forgive my ignorance about it, but in the kind of VLSI design world Carver Mead’s ideas about silicon compilers, does that relate to what you're talking about here about synthesis? I don't know that I personally am following it exactly right.

Gelsinger: Yeah. And much of what Carver Mead really did was how do you turn a gate into polygons that go on to the silicon? So it's absolutely the same generic idea but he's actually lower in the stack because here if I say make it a NAND gate, how do you go create a NAND gate with a polygons of contacts and metal wires and polysilicon layers and wells, right? That was really where Carver Mead-- and it was fundamental work at the time. Today we take that for granted. But when Carver Mead did that that was like revelations of how to generate early VLSI designers, that you could demystify. You didn't have to have a Ph.D. in thermal physics to know how to put a gate down. And that's what Carver Mead did in the very fundamental way for the industry. Some of these later synthesis were how do you take a high level description, something that looks like a program, and turn them into logic gates? And then Carver's work was very much how do you turn logic gates into polysilicon layers? And then the other big

question is then how do you make them perform it? How do you have circuit designs and now getting a signal across the chip and power optimization? These are tens of millions, now billions of transistors that are operating in harmony with each other.

Brock: Okay. Thank you.

Fairbairn: So Andy put you in place as the design manager, the youngest man on the team. There must've been some fallout from that bold move.

Gelsinger: You know, it was bold. And as a young manager I had a lot to learn. I look back on it. It's like I'm stunned that anybody stayed in the team. <laughs> But sort of after we got over that gestation phase as I'm sort of figuring out how to be a manager for it and the design team grew to be about 120 people in the core design people team. We did the project. We worked together as a team for four years and we only had three people leave the team in four years because we became passionate. And, I think, in that sense the 860 and some of this RISC rhetoric we were going to deliver a high performance compatible 486 machine that changed the industry. You know, the team became so committed almost with a religious fervor to doing exactly that. And the team became so united against that cause despite this young, green, inept management leader we were passionate about that goal. And, as I say, for four years the team was so extraordinarily united. I remember we enjoyed-- we had a softball team. We played in the Intel softball league in the summer. And we were there as a team all showing up, even people who didn't go on the field. I remember we had some of the folks from India who knew how to play cricket but not baseball so we're teaching them how to run bases and everything like that. It was historical. And the 486 picnics that we had, everybody shows up with family. We created an extraordinary team rapport and a commitment to success that despite some of those we were going to get that chip out the door. And, obviously, we did. It was an extraordinary successful period of the industry.

Fairbairn: The other thing I'm curious about is that you sort of started your work on the 286 and even 386 was sort of logic and physical design. At some point, you were an architect of one of the most complex computer systems being built. Where did you pick up that knowledge? Was this working with Hennessey? Was this sort of osmosis going along with their mentors sort of bringing you in? Because that's a that's a much bigger step to from the world of logic to the complexities of out of order instructions and those sorts of things.

Gelsinger: Clearly, as I was rising in management roles and responsibilities, you're sort of doing less of the work yourself, you're much more about hiring and bringing those people into place. That's clearly where people like John Crawford became more-- you know, he was a compiler operating system kind of guy so bringing that. We were bringing other people into the team. I'm scurrying as fast as I possibly can. I'm taking operating system classes and compiler classes at Stanford and, you know, running up the stack and learning, computer languages and some of these architectural techniques. It was, , this extraordinary period for me personally because literally I'm taking master's and Ph.D. classes at Stanford by day and designing chips by night right and really learning it. So it was really all of the above. And many of these architectural principles, , that we take almost for granted today weren't invented yet. Some of them were done by the seat of our pants at the time. And, that now have become highly rigorous in the

understandings that went behind it. And, we made some extraordinary choices and some mediocre ones in the process. And, you know, in that it really was-- and I think you find that in almost all great I'll say technical achievements that the team is so focused on getting a goal done, right, that you're forced to be creative at these different pieces. "Oh, well, we don't know how compilers are going to handle that." "Well, let's just go write a code generator tonight." "I don't know. Who knows how to write a code generator? Oh, you did one in college for Pascal. Great. Go get one done. I need it tomorrow." You know when we're starting locate we're doing code generation. Okay. We have to get some instruction set data. What's the latency of jumps? And how often do we take jumps? And how often are they conditional or not conditional? Oh, I don't know, let's go right start calling through programs and let's go, you know build a code set analyzer. So boom we'll have that done next week. And these things that today you would take for granted, and hey they were pretty crude and ugly at the time, but we figured out quite quickly how to start getting some of those capabilities in place to guide the chip design. And like I say, people like John Crawford and some of the others who came along from the software side. I remember some of the great Microsoft debates we had, right, because Microsoft was very quickly as well seeing the extraordinary-- because at some points it wasn't clear if x86 was going to win. So Microsoft had projects that they were doing trying to figure out where else do I port MD-DOS for? What else do I write software for? And I remember some of the early meetings that we had with Bill Gates and some of the other architects from Microsoft because they were basically coming in and saying that we were stupid chip guys who had no idea how to build instruction set computers. You know that we're performance. So they certainly brought ideas to the table. But I remember my first meeting with Bill Gates and, this is in the 386 time and I'm now rising up. I'm becoming a pretty capable chip guy at the end of the 386 and Bill Gates comes and he has this pile of simulations that Microsoft engineers had run. He throws them on the table and he starts chewing us out on the 386 design team. And I'm sitting here it's like this is my chip. You know, it's like you're calling my baby ugly. Right? How dare you? Right at the time and we're just starting the first 486 design work at the time and I'm in charge of that for it. And I start arguing with Bill because some of his data was bad and he starts arguing back. And again, I'm here and Bill Gates is here and a very technical guy, a very capable guy. And we're going back and forth and back and forth and soon there's 20 people in the room, but there's only 2, Pat versus Bill. And we are arguing at each other and explaining why he's stupid. He's explaining why I'm stupid and we're back and forth and so on. After about an hour of this we got to a natural breaking point in the conversation and the senior Intel guy comes to me at the time. He says Pat, "I don't think we'll be needing you for the rest of the meeting." I was furious. <laughs> He was calling my baby ugly. How dare he have the audacity to do that! But you know out of, I say, some of that creative tension, also, clearly Microsoft had a role to play in some of these early trade-offs and even though they were fiery and people romanticize the Wintel partnership, , there was nothing romantic about it. This was raw intellect. Who is going to be the biggest dog in the industry? Who has the best ideas? We would fight more than we would partner with each other but out of the fight became better ideas, better solutions, sustainable technologies.

Brock: Yeah, because in some ways there's a divergence of interest between the two sides, not just even in a business sense but a technological sense about, you know, the desire to push forward the capability and the capacities and what the microprocessors are embodying. And, you know, what Microsoft would desire for, you know, spreading the use of their software. And that was a kind of a creative tension or just a tension?

Gelsinger: Yeah. I kind of viewed it that we were going down this road together. It was like two porcupines going down the road together and, I mean, we're on the same road. We are out to create the personal computer industry. Right? We didn't know it when we started the journey together, but that's what we're out to create. And that personal computer industry would redefine all of computing. Right? It would eliminate minicomputers, make mainframes irrelevant, it would enable the future-- but none of us knew that at the time. But we're going down this road together. The industry is pulling us. We're racing with the industry, the ideas, the competitive architectures, the competing companies into it and then we would squabble on that road and fight and yell and we'd run away from each other and throw stuff at each other. And then we'd realize "Oh, we're still going down the same road together" and then we'd get back together running and then a fight would break out and the porcupines would yell at each other and so on. And we'd poke and squabble and so on. And then we'd figure out, "Oh, we're still going down the same road together" and we'd keep going on. And out of that fiery tension came an industry. And I remember some of the meetings that I was in with Andy Grove, one particular meeting was Andy and I with Bill Gates and Jim Allchin, one of the senior architects at Microsoft, at the time, and it was the one of the famous Bill meetings where Bill took his shoe off and he's banging his shoe on the table yelling at Andy Grove and I. <laughs> It was just incredible. Right? In retrospect, I was there. Yeah. It was just one of those kind of things where boy that was fiery tension between those guys. And I remember, in fact, it was after that meeting when, Bill and and I says, Andy I says, "You were incredibly calm with Bill." And then he said to me this line that I didn't realize the significance of it until I read, *The Swimming Across* [2]book, but he says, "Oh right. I've seen far worse. I've looked in the eyes of dictators." And he had seen the eyes of Adolf Hitler. At the time and it's sort of like, okay. Yeah. Yeah. You can have a ranting Bill Gates or an Adolf Hitler, Andy had seen them both. Right? And that was part of this deep extraordinary respect I always had for Andy. But meetings with Bill and Andy, these were like two intellectual titans coming together and you've seen how the mountain sheep butt heads. That's the image that was always in my mind. It's sort of like, here they come, here they come, here they come. Bam! They'd hit each other. But you'd show up for a meeting with Bill Gates. He had done his homework. He was not only the smartest guy in the room, but he had done his homework, and to me that was always one of these kind of things where these early shaping forces of the industry were really quite extraordinary.

Brock: During your time designing the 486 is when Andy Grove becomes CEO of Intel and it also becomes where I think the period of time, correct me if I'm wrong, where the real implications of like being the sole source for the 386 comes like the success-- the upside of that strategy begins to show itself. So did that, when you're working on the 486, did that add to the kind of weight of the project, or the context in the atmosphere that we're going to sole source this thing, again? And we're going to keep this dynamic going. Could you talk about that part of things?

Gelsinger: Yeah. The sole source decision was made in the 386, or I'll say there was a lot of sources for the 286. I think at one point there were 14 different companies that could produce a 286. There was, essentially, only two: IBM and Intel for the 386. And the 486, part of the architectural battle that was underway-- remember you had the National 32000. You had the Motorola 68000. You had the IBM 601

[2] [Interviewee' note] Grove, A. S. (2002). Swimming across: A memoir. New York, NY: Warner Books.

and all of these competing microarchitectures coming out at the time. Part of the battle for those was Intel was here making the x86 family proprietary. And it was before AMD had really risen up as an architectural competitor. And that added to the importance of making the 486 successful but it also added to this risk and alternative architecture perspective that we have to succeed because we can't be solely dependent on Intel. So it was an incredible, I'll say, crucible of pressure of getting the 486 done. The industry was already well underway with the PC compatible industry, so there were ripe and ready customers. Some of these were big names. Dell was now starting to really emerge. And IBM, of course, had built their architectures and their attempts to rebuild the architecture with microchannel were not having success. Compaq, of course, was racing forward to become the biggest computer company on the planet. But then you had all these others, you had NEC and Fujitsu in Japan. You had Apricot and Siemens and these companies emerging. I remember in the early days of the 486 we had-- and I think it was maybe the first 500,000 chips that we had sold. I get a spreadsheet that showed me the top 10 customers for the 486 and I only recognize five of them because the personal computer industry was born; this compatible industry, that a motherboard out of Taiwan could be assembled with a microprocessor, an operating system from Microsoft and compatible software. And boom people were showing up in China and India and Europe delivering personal computers with names we had never heard of at that point in time. And as that emerged, it really became this period of time where, okay, Intel won. And it wasn't clear at the end of the 386 if we had won or not. Right? But in that period of time when the 486 and the DX2 Intel was ahead process-wise, architecturally-- the DX2, in fact, was faster than any RISC machine at the time. So not only had we been able to win the compatibility argument but it was even winning the performance argument something that was impossible to have conceived of; and, of course, we did it because it was compatible. We were able to bring the chip to market so rapidly, benefit from Moore's Law, benefits that we had as we were able to jump into the next process technology; hit 66 and eventually a 100 megahertz before anybody else had gotten past 50. So even though there was some instruction set compatibility overhead, which was there, we just overwhelmed it with Moore's Law and board and operating system and compiler compatibility that the industry would never be the same and x86 just became this dominant architecture. It was in that period of time in the late 80s where all of those things were starting to come together and by '92, '93, let's say as the saying goes the fat lady might not have sung yet but the architectural battle was over. The x86 had won and the nineties were just really seeing that come to full fruition.

Brock: So, how long did you stay with the 486? When did you begin your next thing? How did that work?

Gelsinger: Yeah. So, you know the 486 was super successful. I was then managing the PC business for Intel, sort of moved into the next layer of leadership. I was then asked to go lead a project. Andy asked me to lead a project in personal video conferencing for a period of time. And I went to do the job largely because Andy asked me to, you know. Had he asked me to climb the Eiffel Tower I'd climb the Eiffel Tower. <laughs> He was this extraordinary figure in my life, but Intel was getting worried about what's going to be the thing that consumes these MIPS and that was becoming the dominant. And video and this is before video was anything of significance on the PC. We had yet to have-- the first MP3 players were just starting to emerge, to the idea of video being played on a personal computer. Oh, wow. Really? You know, things that we just take for granted today. And we were showing these, you know, 480 frames, 6 or 8 frames per second. And we were so excited. And I remember one time showing it to my wife. She said, "That's ugly." <laughs> It's like, how can you call it ugly? That's a technological achievement of great

success. Let me tell you about how DCT worked and great key frame worked and all this other-- and she's like, “It’s ugly.” But that was in that phase. So I went and worked on that for several years and we made a fundamental error. We bet on ISDN. So we bet on that, as the communications, the thing that would replace POTS at the time, the PSTN network. Obviously, that was a bad choice and we struggled. And so that was my first major failure at Intel. And the product became called ProShare. And, in fact, after working on that for a couple of years and the failure of that project at the sales conference of Intel we had a parody on ProShare and it was called PrayShare. <laughs> You know, we pray that it's going to be successful someday and we're going to create video conferencing.

Fairbairn: I didn't even realize Intel had a video conferencing business.

Gelsinger: Yeah.

Fairbairn: Was it a business or only a project? I mean did it turn into it?

Gelsinger: It was a project. We launched product for it. Eventually we sold those projects to PictureTel eventually, as we exited the business. But the business never got that large. And it was my first major failure. But I learned an extraordinary amount because now I had software engineers working for me, communications engineers. We started to work with the FCC on ISDN. And I find this over and over in your career, my career and other people's careers – that loss often your greatest periods of learnings come out of your failures. So extraordinary period of learning for me. And my next major assignment after that was to become the CTO for Intel.

Brock: Flashing forward though from-- I mean video on computers has become a gigantic-- I guess what Gordon Moore would have called a transistor sink. You can call it a cycle instruction sink or whatever. Maybe the timing, maybe the particular technology, but I mean video conferencing, video over computer I mean has become a huge resource.

Gelsinger: Yeah. This period of time when video was seen as too hard of a workload to run on the x86. And you might remember there were companies like Chips and Technology, and it was the multimedia companies and others like that. They were doing special purpose video processors to add into your computer to get at work. And Microsoft was coming out with some of the basic Windows media capabilities. So all those were starting to emerge but we were pushing the edge of the technological envelope. It wasn't quite there to be mainstreamed yet. And a whole lot of companies thought they were going to establish sort of the next major architectural footprint by doing those workloads. And then what happened? Well we added the multimedia, MMX, to the Pentium. So now instructions that were truly optimized to running media and video workloads. Now it became popularized and Microsoft did the right things in terms of the operating system to deal with media workloads, audio and video; and, now, of course, everybody does it. You do it on your phones and everything like that but at that time and you know that early nineties era, it was we just couldn't quite get there. And, obviously, Kodak was a big player with some of the core digital video technologies at the time. And we, of course, know what happened to that story and many of those companies that were doing special purpose processors at the time, specifically in that area like C&T, Weitek, others, made a business and then the business

disappeared as the x86 processor became possible in doing it. We added the floating-point processor on chip in the 486. It was a big deal because you might remember the 387 was a huge business for Intel. Weitekh was doing those a special-purpose chips. And that was really, as I called it, the big sucking sound of Moore's Law. Anything that was on that motherboard that could be brought into that big transistor pool of the x86 sitting at the middle would be right, caches and media and floating point and bringing those on. And once you put it inside of the basic processor, every software application could rely on it. And that was the magic that you now started to unleash at the software spiral as we called it because if it was special, well, then you had to add a board to your computer and you had to get the special software and it was heavy on the user at the time.

Brock: Not everybody had it.

Gelsinger: Right, at the time. But when we add it to the microprocessor, magic because then the operating system would assume it's there. The compiler would assume it's there. And software could always build on top of it. And then this was an extraordinary period for Intel and the personal computer industry where-- and I remember we started what we called-- as I became the CTO for Intel we started what we called the Intel Developer Forum. Literally that was the place the industry came together. Why? Because if it was in the Intel platform everybody could build on it. If it wasn't, nobody would build on it. It was really this defining point where, you know, that was where the industry future was defined.

Brock: It's really interesting to hear you describe it that way because it's almost like, you know, there's so much kind of software within the microprocessor, you know, in terms of well microcode and all this sort of things.

Gelsinger: Microcode, BIOS and drivers.

Brock: But, also, in a way that, you know, some of the logic that's in there is equivalent to software. So you're almost talking about like this software platforms that are put into the silicon that really facilitate the wider use of software that does particular kinds of applications be it video, media, whatever it is, communications, security stuff.

Gelsinger: Yeah. Speech processing, et cetera. Security. Yes.

Brock: I had never really thought about it in that way of how the microprocessor becomes like an enabling that really very directly and enabling platform that makes certain types of software more accessible, practical, cheaper. Is that a fair picture?

Gelsinger: Mm-Hm. Yeah. It's a very fair picture. And, in fact, we spent extraordinary amount of time in that that period of time building up. Today you'd call it an ISV program. Oh, the hardware guys got to go work for the software guys today. Today, you take it for granted. At this point in time, nobody did that. We really were inventing what that meant to be able to get software and get software guys really building on it. Because you always end up with this time lag problem, “I need you to write software for hardware that hasn't been invented yet. Can you do that?” And, of course, nobody would do that because you couldn't.

You might have emulators and stuff like that, but they were always so crappy that unless you were super motivated for some reason that drove you, you would've never even tried that. And then it economically wouldn't make any sense because then you'd say, "Well, until that hardware was broadly available, why would I ever commit to a piece of software that I'm trying to distribute in the industry?" So trying to define that hardware/software boundary in the normal time lag and enabling of these key new software functions was a huge issue for us through the nineties. And that was very much how to break through that and crack many of those problems is a key aspect of what I did, as I say, when I became CTO for Intel. I was CTO for almost five years and just an extraordinary period of my time and history as I was, in the early nineties, also helping to create the key hardware standards. It's where USB, Wi-Fi, PCI, AGP, graphics interfaces were critical; and we were responsible and I was critically involved in many of those things. When I get up and speak to audiences today I describe that I've had the Cinderella career. And if you've ever used a personal computer, connected on Wi-Fi or used USB I worked on it. Like, of course, how many of you haven't used one of those three things?

Fairbairn: So which years? Was it 2002 to 2005 that you were CTO of Intel?

Gelsinger: Yes. Yeah.

Fairbairn: Okay. And was that a new role?

Gelsinger: Yeah. It was great because Craig Barrett was CEO for Intel right there in the late '90s early 2000 period. And I had taken sabbatical. Intel had a sabbatical program and when I left on sabbatical I didn't know what I was going to do when I got back. So I get back from our sabbatical and there's a note on my chair to call Craig. And basically it's, "We'd like you to be CTO." It's like well, it sounds pretty interesting. What's a CTO? And he says, "I don't know figure it out and tell me." So what I did was,as we were starting to formulate what a CTO meant for Intel at the time, I went and interviewed 50 CTOs. I remember it was Craig Mundie had been named the CTO for Microsoft. Greg Papadopoulos was the CTO for Sun. Bill Raduchel the CTO for AOL. I remember those four, but I interviewed 50 CTOs. And what I found was the CTO the most ill-defined C-title of any of them. You know what a CEO does. You know what a CFO does. You know what a chief counsel does. But a CTO? some of them had R&D, some didn't. Some were in charge of intellectual property, some weren't. Some had labs, some didn't. It was a very ill-defined job. I decided there's no definition for CTO, so I'll define it in the my own image for exactly what I want to do. <laughs> And then I can be entirely successful at it because it's exactly what I want to do. So really sort of defined the job spec for being a CTO Intel. I got Craig and Andy to agree on that definition and became the first-ever CTO for Intel; and it was a glorious period of time.

Fairbairn: How did you define it?

Gelsinger: Well, I decided at Intel that it had to be lab-- you had to have earnest money; to get something done at Intel you had to show up with real resources and real ideas. Otherwise, the business units, the fab guys, you had nothing. So I said we must have a lot of these advanced development labs working directly for me under my budget. I insisted that we pull these labs together. It was a bit of a political battle inside of the company at the time, but I said it has to be mine. And I got Craig to support that. Also, I had

to support research funding, what we were doing in the upstream pipeline, in the universities as well. Also, I took over the patent program and I owned the senior technical ladder for the company. So I was in charge of who became fellows, who became principal engineers, et cetera. So really, in some ways it was controlling those things gave me power but it, also, said it created real technological leadership, as well. So pulling those things together became the role for CTO, but it very much things like we already touched on like with Intel Developer Forum and so on we were going to set the future of the x86 industry. And out of that came things like USB. We were going to attack these hard problems like upgrading a PC, opening it up, shoving in boards and so on. No. We're going to define plug-and-play which today, of course, that's obvious. Right? Well, in that period of time it wasn't obvious. You know, we're going to create wireless connectivity. Oh, you know, I need an Ethernet jack. Remember, Intel helped create Ethernet. No. We're going to do it all wirelessly. Oh, wow. How's that going to work? We had to fight battles. Today, you take Wi-Fi for granted. But in that period of time it was not for granted. There were competing standards in Europe, particularly, in China. One called WAPI that came out of China at the time, that they wanted to control their own and they were driving that as a standard into the ITU. Right? So I learned more about international standards, ITU, global diplomacy. I took seven trips to China in one year specifically fighting some of those battles. I remember we had the former chair of the FCC on the board of Intel at the time. And I started describing him the WAPI -- Reed Hundt -- and the WAPI battle with him. And I'm giving him some of the technical ins and outs of the issues. He says, "Pat, stop. Shut up. This has nothing to do with technology. This is brazen politics." You know, here's the FCC chair giving me my tutorial lesson in international politics which is just extraordinary and how to negotiate with state departments and international trade and ITUs and so on. So just some extraordinary things came out of that period of time, things that today we take for granted came out of that period of time when I was a CTO for Intel. It was really a superb opportunity to grow, learn and really help to shape the industry that we take for granted today.

Brock: So I had question about that. You said you defined the future of the x86 or the personal computer. What about the next step, the things beyond, you know, the personal computer, you know, the mobile, the other things that have evolved since then? Was that-- how did that fit into Intel's strategy and your role in that?

Gelsinger: Yeah. And some of those things, clearly things like Wi-Fi, Centrino, et cetera, were sort of in the tail end when I was CTO. In January of 2005 I moved back to take over microprocessor development for Intel. It's when we were in the middle of the mobile, when we were in the middle of the multi-core and AMD Opteron battle for it. So I moved it back over to become in charge of microprocessors and desktops and servers and so on to take that battle on because, you might remember, Intel created the industry standard x86 server at the time and AMD was eating our lunch. So I left the CTO role, and about the time when I left the CTO role was the period when a lot of these mobile conversation started to come to the fore, as well. And Intel struck out and not just three strikes, like five strikes in its mobile strategy. Absolutely dominated personal computers, absolutely dominated data centers in the cloud, and absolutely struck out right when it came to mobile computing and mobile devices. And most of those things didn't work for me at the time and I was sort of where you got some of the things, some of the technologies under way. But you might recall some of the-- that's when Qualcomm's ascendancy was really starting to emerge. The mobile phone with starting to be seen as an important form factor into the

future; moving from analog to digital in that space. And Intel made some very, very bad choices ones that, you know, I argued quite strongly against that the time, things like WiMAX, et cetera, you know, were big investments in the part of Intel. And my role had largely shifted by that point in time, so I wasn't in charge of it anymore. But I was quite the critic of some of those decisions for Intel.

Brock: It seemed like the strategy on mobile was kind of whip sawing around. But it seemed like on the server side, it was very, you know, focused on that kind of x86 architectural hold, Moore's Law strategy. Is that fair?

Gelsinger: Well, several things were occurring. One was we had gotten I'll say very thoughtfully started to build what an industry standard server could look like and the idea of defining the server, the hardware components, the BIOS components and starting to build that up, a real ecosystem that could build server motherboards. Because remember this is the period of time where the RISC machines dominated the server landscape. You had HP-UX running on the HP architecture. You had SPARC as an architecture and the Sun platform. You had AS-400s from IBM running on RS-6000; so it was a very verticalized industry. And we took a very PC Centric view of that but building that industry standard server. So that was getting underway quite nicely for us at the time. So it's a very thoughtful strategy. We feared Sun, right, and what they were doing with SPARC at the time. And, you know, they were really seen as the enemy. And then when Opteron launched and started to eat our architectural lunch because we created all of the BIOS standards, the networking standards, how all of this was going to work, the x86 server industry is starting to take off in that time frame and then Opteron is launched in 2003. We were up here at like 97 percent market share in x86 servers. It dipped down into the sixties.

Brock: Wow. I hadn't realized it was that.

Gelsinger: Yeah. And the average selling price had halved. So our market share had a halved and the average selling price had halved at the hands of AMD Opteron. And this was at the end of the period when I was the CTO for Intel and I remember I laid out the multicore strategy for Intel in this period of time. We're now 2004-- late 2003, 2004. One of my famous papers that I wrote, was the nuclear paper, which basically described that the power density of us staying on the single core approach to computing would create thermal densities that would rival that of the sun and how unattainable and unachievable that is. Just like I talked about the test industry before the cooling industry was betting on those thermal densities. And, in fact, there's a professor at Stanford who I just met with the other day and he was complaining that I killed one of his startup companies that was doing a thermal cooling technology specifically to solve that. But, out of that multicore transition I moved out of the CTO role, had laid out that strategy, then moved back into the development role and really began that multicore strategy, the tick-tock development model that Intel became famous for as well; and began the assault on AMD. And I'll say, part of it was we had done good architectural work, good business strategy work and how to create that standard industry. And we had a clear enemy named AMD Opteron. And I'll say, good strategy, clear focus, who you're out to compete with and you know that began a journey from 2005 to 2009 where it was we are going to win them back and we are going to crush them. And, obviously, we did a pretty good job. Intel's market share after that was higher than it was before and our ASP was higher than it was before. So not only had we recovered everything but we had gained more. And, of course, that became the

foundation for today all forms of data centers, all forms of clouds, all forms of public clouds have been built on those fundamental industry standard scale out multicore server components. And every one of us uses those many times a day every time you access a website and you don't even think about it.

Brock: In our own personal computer there are multicore.

Gelsinger: Yeah.

END OF THE INTERVIEW

