PAUL LASEWICZ: This is an oral history interview, conducted on April 15th, 2003, with IBM Fellow Emerita, Fran Allen. We're located in Armonk, New York, and the interviewer is IBM Corporate Archivist, Paul Lasewicz. Welcome, and thank you.

I know you're officially retired now, but that doesn't mean that you're not doing anything. Can you briefly summarize some of the things that you're still involved in, and if you have an official title that is attached to the work that you're doing?

ALLEN: Well, my official title as a retiree is IBM Fellow Emerita. The very wonderful arrangement that IBM has for its researchers, some of its researchers, and for the IBM Fellows, is the privilege of a position at IBM, which doesn't require or allow any useful work [LAUGHTER], in terms of strategies in the company's current business.

But, [this position] does give the privileges of access to the library, the buildings and the colleagues that one has worked with over the years, which is really very wonderful. So, that's what my official title is.

What I'm actually doing is focusing on two things. One is the early history of the projects that I had worked on, and

the threads of the technical ideas that grew out of those early projects: how these ideas were formed early on, and how they influenced later thinking, as the field grew. And then the second thing is working on women's issues in the technical computing field. So, in both cases I've been doing talks, and research, and investigating these areas.

LASEWICZ: Sounds like a lot of fun.

ALLEN: Yes, it is great fun.

LASEWICZ: Why don't we just go back to the beginning. What schooling and education did you have before you began your career here at IBM?

ALLEN: I was trained as a high school mathematics teacher. I graduated from Albany State Teachers College, which has now been subsumed in the State University of New York system, and graduated in 1954 with a Bachelors Degree in mathematics.

Then, I taught for two years at the high school I graduated from in northern New York State. I realized I needed to get a Master's Degree in order to be fully certified. So, I went to the University of Michigan, which was the least expensive place I could find [LAUGHTER] and I got a Masters

degree in Mathematics at Michigan. That was in 1957. And, I was deeply in debt at the end of that time. IBM came on campus and I joined IBM.

LASEWICZ: When you were pursuing your college education, obviously math courses interested you, what attracted you to that in the first place?

ALLEN: Well, I was attracted to a number of things, and math was one, for certain. Ultimately, physics was also another one, and I had a minor in physics in college, and started to take some graduate work in that. Then I was also very interested in history.

And the reasons for all of those interests were teachers, teachers in high school and in seventh and eighth grade and even before that. These were people that really excited me about the topics.

I also, by the way, was very interested in English. But, then I had a high school English teacher that just wasn't very interested in English. She wasn't interested in English, and my interest just faded after that, though it's something I've always felt I'd like to pick up and pursue.

LASEWICZ: Can you talk about your work history at IBM? You mentioned that you started in '57. What did you start working on, and from there can you list some of the major projects that you've been involved in over the years?

ALLEN: Yes. Yes, I've been thinking a lot about this. I think it's a great story. Let me start with when I joined in '57. By the way, I only intended to stay long enough to pay my debt, because I loved teaching high school mathematics and I was going to go back.

But, I joined, and it was IBM Research where I ended up. It was in Poughkeepsie, on [Boardman Road]. I joined IBM Research on July 15, 1957. The FORTRAN language compiler had become a product on April 15, 1957. And there was an intent goal that all the scientists at IBM Research were going to use FORTRAN by September to do their work on the 704 that they had there.

There was a great deal of resistance by the scientists because they said that it wasn't possible for a high level language with a compiler to produce code that was good enough. And so, my very first assignment, having been a teacher, was to teach the scientists this language and this programming and how to program in it.

That was of course a learning experience for me too, because I didn't know the language, and that's what teaching is often about: you have to learn the subject as you go. It turned out to be tremendous. That really set my career, in some sense, that very initial assignment, because I understood what the possibility was for high-level language in a compiler, which was extraordinary. That compiler was an extraordinary compiler. The code that it produced was better than the object code that the users would write, in several ways. Surprising things in the code came out of that compiler.

That was my first assignment, and it was successful. There was never an edict that they had to use it, but people in science and research gradually moved over to using it, because it was so good.

I had a series of other assignments as a programmer. I was hired as a programmer, by the way, and not a scientist. I then ended up in 1959 on a project called Stretch, which was a project that had been started in 1955. Actually, in 1954, but really it got going in 1955. [Stretch's] goal was to be 100 times faster than anything else that existed, in particular, the 704.

I ended up on the compiler; I can give you a lot more details on all of that. But, that became the project that I worked on next, along with the Harvest Computer, which was related to that. I ended up being responsible for the compiler optimization and for the high level language that was associated with Harvest, and ended up on that until '61 or '62, and then joined the ACS Project, which was another great project.

I'd like to describe more about those. But, those two projects really, along with my teaching FORTRAN when I first started, really set my career.

LASEWICZ: You mentioned that Stretch started in '55, and you were brought on board in '59, that was a four-year delay. Was that reflective of any internal conflict about the need for an optimizing compiler or was it just a matter of waiting until the time was right to start the project?

ALLEN: There was early recognition that they needed an optimizing compiler, that they needed a compiler. In fact, there's a wonderful book that is no longer in print, I'm sorry I can't remember the name of the book. But, the introductory chapter was written by Fred Brooks, who is well known in the field, and was on the project. What Fred Brooks, who was on the Stretch early design, said in the

chapter in the book describing the project, was that it was recognized early on that [because of] a compiler's high level language, compilers were going to be needed in order to take advantage of this machine.

LASEWICZ: So in '61, '62, when you got involved in ACS, do you want to talk about that a little bit? Was this a natural segue from Stretch to ACS, or was it a different line of thought?

ALLEN: No, let me connect these two projects. I'd like to connect the Stretch Project with the ACS Project. It's really connected by people. Let me describe a little bit more about the Stretch Project.

The goal of the Stretch project was to be 100 times faster than any other machine at the time, particularly the 704, but there were some other machines coming out, by IBM.

And because of the state of the art at that time, a great deal had to be invented in the state of the technology. This was very, very early times in '55, '56, '57.

In order to achieve the performance that they had intended, this goal of 100 times, they recognized—and this was really quite interesting by itself—they recognized what

performance bottlenecks would exist in their intended machine, that had to be overcome.

One of them, which is quite interesting, is the time it takes to load data out of memory. The memory latency, it's referred to. It's also now referred to as the memory bottleneck. The problem is still with us. It's one of the major problems [of] computer architectures [at] the software level, at that level for high performance machines, [it] is hard to overcome that latency.

It's just gotten worse and worse over the years, and the reason is because the processors have gotten faster and faster and faster. Moore's Law is continuing, which means that the processors will double in speed roughly every two years, or 18 months, or whatever. There are variances on this.

But the bottleneck is getting the data and information that's stored in the memories to the processor. And the bottleneck is actually the real time it's going to take to get information from one kind of storage into another kind of processing storage.

So, it was recognized in early 1955, they had to overcome that particular bottleneck. In the process of doing that,

they'd built up an extremely complicated machine. It could have six memory references in flight at the same time; it was a lot of concurrency. I believe that it is a super scalar machine even though that word didn't show up for many years later.

But, one of the goals, from what I can deduce from reading, is to have 1.6 instructions executed per machine cycle. The definition of superscalar is that you had multiple instructions, executed in the same machine cycle. So this is a piece of work I have to do, to actually verify that, and see whether it is really indeed the first super scalar.

But it also may even have had a server at the beginnings of a cache, because the information was buffered up. They came back from memory out of order, they got it, it got put in order and then fed to the processor itself, which had very complex units, and was pipeline—additions and so forth were pipeline.

In addition to all of these complexities just to get the performance, the functionality of the machine was amazing. There were 1,700 instructions, and one could, for example, walk a list in two instructions, and the instruction itself could be very complex. And the variance on these instructions was huge.

It was, in my mind, a programmer's dream because one could find multiple ways of programming everything, anything, but it was a compiler writer's nightmare, because of all of these different variations, as one of the compiler writers it became a difficult—a huge, huge challenge.

The technology really wasn't there to do it from a compiling point of view, and we had a very ambitious compiler too as well as an ambitious machine. It was a wonderful project, in that the goals were extremely high.

But we failed. Officially at that time Stretch was not 100 times faster than the machines that came out when it was delivered to Los Alamos, that first machine went to Los Alamos in 1961. And it wasn't 100; it was slow.

Worse yet, the code that the compiler produced was also slow, which was a part of the problem. One of the codes that we had, [for] one of the machines we had sold to a weather forecaster, a 24 hour weather forecast took 18 hours to calculate. [LAUGHTER] So, we had problems.

And the net on the Stretch was that, at the time, the president of the company, Tom Watson Jr., who had announced the original project to the world—I mean the relationship

between companies and the press was quite different at that time—then apologized to the world for our failure, lowered the price, and in lowering the price of the machine, it essentially took it off the market.

We fulfilled the orders we had, but it was viewed as a failure. The man who had handled this project was Steve Dunwell, who was ostracized, completely ostracized, removed a position he had gone onto after the initial Stretch had been started. I remember him in the basement of the new Research building, in Yorktown, you know punching his own cards, working on an educational system.

But then, after 360 came out—360 by the way, came in '64—it was recognized [that] the technology in 360 owed a great deal to Stretch. And that was true of all of the subsequent architectures and machines. Stretch really did, with its very high goal in its attempts to do things, had actually produced a tremendous number of results and set the direction for the way things should be done, could be done, even though it didn't totally succeed.

It did work, of course. Steve then was made an IBM Fellow, and that was really a great move. Again [because] Watson realized and accepted that there had been great value to

the work that he had led, made him an early IBM Fellow as a result of that.

So, [for] that particular project, there was a companion project called Harvest. This was a machine that was even bigger in size than Stretch, which was built for the National Security Agency for code breaking.

That machine was started in 1956 and delivered to the National Security Agency down in [Laurel], Maryland, in 1962. That machine was a totally different architecture, it was a streaming machine. And there's considerable interest lately in that machine and in that kind of architecture.

It was a streaming machine, which could take information that was gathered from the listening stations that NSA had around the world -- mostly listening to Russia at the time, the Soviet Union -- and then take that vast amount of data, some coded, some open, and do code breaking on it, in order to find out the codes being used by the Soviet Union and the Soviets' information exchange, or by any other group.

The way it worked was: attached to this machine was a [tractor] tape system, which contained vast amounts of information, and the information could stream from the tape

system through the Stretch Harvest memory, through the decoding unit, the Harvest unit, and then back out—the answers, whatever the results, back out without ever stopping. It was a completely balanced machine, the IO and data rate and the analysis. It was an amazing engineering feat, done by a guy named [Jim Pomerene], who was an IBM Fellow recently retired from IBM a few years ago. We had hired him as an engineer to head this engineering firm, from Princeton, where he had worked on their projects down there.

My role in all of this was originally to be part of the compiler project, and I know you had asked the question about why did the compiler project get started so late.

Well, I think that much of the software got started late, and that's the typical story [LAUGHTER] and there's no good reason for that. Time was running out, and a number of us from research just got drafted for the software on the project. I was involved with the Harvest side of it, but also the compiler part.

I ended up being responsible for the optimizing section of the compiler for both Stretch and Harvest, and [being] the liaison with National Security Agency on the language that we were designing with them called Alpha, which was a

language for code breaking, which was a terrific match for the Harvest machine that allowed the cryptologists to express their solutions in a very, very high level form.

So, I was involved in that, and I had a group of about 17 or 18 people, great people that worked on this optimizing side. A number of us went down to the National Security Agency, and spent a year, I think it was, down there installing the hardware and software, it was all being done at once.

Those of us on the software side were down there doing [RG] bugging and bringing the compiler that we had built up. It was an interesting time because the National Security Agency was not known, and some of us had high clearances.

Actually all of us were housed in a kind of temporary section outside the gate, inside the main gate, outside one of the gates. And we were not supposed to know what it was about, even those of us with high security clearances, what the real purpose of this was.

But it was exposed in the New York Times when a couple of the National Security Agency people defected to the Soviet Union, and there was a write-up in the Times, and we all understood much better what we were doing [LAUGHTER] and

why we were doing this. But, it was actually known as the Bureau of Ships. That was the line in the.... The NSA was known as the Bureau of Ships. That was the line in the Federal Budget at that time.

In IBM there were various code names, of course, all projects in IBM have code names when they're getting started, which was unrelated to NSA or Bureau of Ships. But that's kind of irrelevant, though one will hear the reason for tractor is that the project was also called Farmboy. [LAUGHTER] I don't know what the histories of either of those are—just a curious little side of history.

That project ended up being very successful, and Harvest was used for many, many years there. When they could no longer get parts, when IBM was not interested in carrying or doing anything more in terms of developing the parts that were needed, they had to turn it off.

But, my final assignment there, the final task I had was to write the acceptance test for the Alpha language, running it through the compiler.

And what it was to be, because that was designed for code breaking and we couldn't get involved with anything that related to code breaking, was to automatically abstract

articles, Time Magazine articles, [which] were digitized at that time. And so, we automatically abstracted articles from Time Magazine. I wrote the code in the Alpha program for that, and I thought I was going to be probably at NSA for years trying to do that. It was an amazing thing to do, and it worked.

I think it just reinforced, for me, the importance of high-level languages, because it was fairly straightforward to write the stream analysis that was needed in the Alpha language, which was a great match for the machine. You could just stream through the text of the articles, collecting statistics on word frequencies, and based upon them, what words seemed to be most important, then stream back through the text again, and pick out relevant sentences and things like that. It was amazing how well it worked. That was that project.

So, I not only did the final acceptance test for that Alpha language compiler, but then I [also] wrote a final report on it. All of that stuff was classified and disappeared into NSA. So, there's no record, I was able to keep no record of those at the time.

What had happened, in the meantime, the Stretch people had kind of—most of them went on to do the 360. In that

period, people like Fred Brooks went on to be one of the key designers on the 360. And he had been one of the people on the Stretch early on. There are many people that were just about to start in that same period.

There was a John Cocke who had joined Stretch in 1956, ended up on Stretch, and he was just a young—[had] just gotten his Ph.D. from Duke, I guess it was. He ended up being the architect of one of the most complex components on Stretch. Basically, it was called the "look ahead," but it was the pipeline of instructions flowing through the computational units.

And because it was...these instructions would get way ahead of themselves, when an interrupt happened they had to back it out and bring it back to the state where it was correct. There was a correctness, it would be doing computations and producing results way ahead of what were valid, would be valid to hand it back out of it. Very, very complicated box in the machine.

John then went to Research. He went to Research and became something for ACS, the Advanced Computing System project, though at that time it was called Project Y, in Research, Y, the letter Y. That project was being built up in Research, and John and I came back then from the Stretch/

Harvest work. I had been interviewed, asked to take another job on the 360, which would have involved more languages and things like that.

I didn't like the job itself, because it was going to involve a huge amount of travel, and I was in a liaison between the Poughkeepsie work and the work that was going on overseas, and kind of coordinator of all of that in England and in France, and so forth and Germany.

So I decided that I'd go back to Research. Besides, I had enjoyed working with John on this, and so that's where the ACS project started—again, with a huge goal of being the fastest machine and a single threaded, single instruction kind of machine.

In Research, I worked on what was the experimental compiler then. I was involved with building the core of the optimizing part of that. And that then moved in '65, I think in '65, to California, relabeled Project ACS, moving from research to product. I went out there at that time. There was a wonderful conference called the Arden House Conference, which was part of the transition, bringing people together to talk about the product, what the shape of the product could be.

I have to tell a little story related to that. Arden House was one of the houses that the Harriman Family had across the river, across the Hudson, which Columbia University owned, I think at that time. Anyways, they used it, and IBM rented it for this conference.

It was one of these small castles in the mountains over there. When I arrived—it was a small invitation list, limited I think partially because of the size of the place, but they wanted the conference to be small—I walked in and the man was a complete stranger looking at the registration list, and I said, I'm Fran Allen. He looked up, and he said to me, you're a woman. [LAUGHTER]

Burst out, I was the only woman at the conference. And he said, oh, dear. I've doubled you up with Gene Amdahl, Allen and Amdahl. We've done this alphabetically. [LAUGHTER] So anyway, that was...

LASEWICZ: They had to do some scrambling.

ALLEN: They had to do some scrambling. Anyway, that was again a terrific conference. It was also a terrific project, in terms of pushing the envelope on everything. Unfortunately it was canceled in 1968 for business reasons.

LASEWICZ: Were they tied to the success of the 360?

ALLEN: No, it was because it was non-360 compatible, and IBM had just gone through the agony of getting the 360 out, and the cost that was a bet, bet the company project-cost-wise.

So they were not willing to come out to have a project that would be totally different, architecturally different and at the very high-end performance. We've always struggled in IBM with that market. It's the prestige market; it actually in the end drives so much of the other technology, as a fall out of trying to stay at the very top of the performance curve. But, it's costly.

So, those were the first two projects which very much shaped my work, my career, my technology, my way of thinking about problems in this space and that space.

I ended up understanding so much about the hardware and the software tradeoffs, because, something I should say about ACS, we built, as a result of our experience with Stretch, we built the compiler before the machine, in order to be able to design the machine.

We had a great simulator, the compiler we really went after trying to produce the very best code we could. And we

drove the output code through a simulator, which could then analyze what kind of performance these code streams could get, and also, what the codes looked like. We ran a lot of the engineering scientific kernel codes through the compiler, and were able then to make decisions about the depth of pipelines, about the number of registers, about a lot of the things that ended up in the architecture, because we knew the shape of the codes as it could come out of a very powerful compiler. So, that compiler was really formed in that, we formed actually the basis for the theory of program optimization, and the pragmatic structures for program optimization. We owe a huge debt to the early FORTRAN compiler, which was very significant.

Thinking about the role of compilers in designing machines: we also shaped a lot of the architecture as a result. [Compilers] really have to be considered at the same time, or ahead of time, because it really is ultimately how the performance gets delivered. You can't put it all in hardware; software can't do anything if the hardware can't do it either.

We've lost a lot of that. The next machine that was built a few years later, it was a big impact, the RISC machine, really did the same thing, it really built the compiler first, and then the hard work came out of that afterwards.

In a totally different machine this time, it was very, very simple and straightforward.

LASEWICZ: The RISC machine you're referring to is the 801...

ALLEN: Yes, right.

LASEWICZ: So that leaves us in 1968. Is it safe to assume that you did more stuff after that or...?

ALLEN: [LAUGHTER] One of the things I did, and it's kind of too bad that it isn't done more often, is I did a lot of writing in that period.

The project was over, and we're all sitting out in California. People kind of left the project going in many different places, some formed their own companies, some joined competitors.

I wanted to keep the compiler work going, so I got in touch with the software division in IBM, and got some compiler work going out there. I was working for Gene Amdahl at the time, who wanted to build a 360 compatible version of ACS, using some of the ACS technologies.

So, I was working for him. This was before he left IBM to form his own company. But, I also spent time writing up things that we had learned in compilers and languages and that sort of thing.

It was those things that we wrote up in that period, which initiated a lot in the field of optimizing compilers. So, I just can't encourage people enough to document their failures, we don't do that very well.

Document what they learn, or document what they think is important. We don't typically take time out to do that within an industrial setting. We jump to the next project and go on.

But that was a critical piece of work to have done at that stage. In the seventies then, I took a sabbatical, IBM Research has a nice sabbatical program. I guess I got back to research, whatever it was.

I spent a sabbatical at the Courant Institute at NYU. John Clark had spent a year before there; Jack Schwartz who was heading the department down there had been deeply involved with ACS. [He] had some wonderful work going at Courant Institute, compilers, and in fact for many years we hired many of his students into research.

There was a combination of theory and practice. There wasn't just theoretical work, it wasn't just practice, the shape of the work was both. And I've always felt that theory without practice is maybe nice and maybe pretty, but it's not going to influence computing as much as the practice side. But the practice has to be backed up with the ability to talk about, reason about it, and formulate it so that it can be reproduced.

So, I spent a year there, came back to Research to join kind of an infamous IBM project called FS. [LAUGHTER] It was called Systems A, I guess it was [LAUGHTER], unbelievably awful. [LAUGHTER]

I got myself into some trouble. I was supposed to work on the compiler, but part of the thesis of that project is that they said they wouldn't need compilers, that it had a high level form, and that would be the interface rather than low level instructions, and high level form. So, there wasn't going to be any need for these optimizing compilers and stuff.

Anyway, I wrote a memo at one point, when they were thinking of moving it out of Research into product, and I

wrote a memo saying it's not going to work. I had very specific reasons—it went back to the memory bottleneck.

It was this idea that you've got to get information out of memory to the processing unit, very fast or in parallel, in some way it cannot be the bottleneck. And this machine had [double], at least, the bottleneck of the...made it much worse.

So, I wrote a memo on that. I was considered, I guess, not a good team player [LAUGHTER] and I was essentially just kink of put on the shelf for a while, no salary raises, no promotions, no kind of job in some sense. Somebody from England came over who was a good team player, and I had to work with him, yes, and it was just a nonsense time.

LASEWICZ: So much for wild ducks.

ALLEN: [LAUGHTER] that's right, when they get in the way...[LAUGHTER]

Anyway, we've done this many, many times, it's not the only time. John Cocke in fact had multiple times, just hit these kind of...He couldn't take going in the direction that was the strategic direction, because it was clearly not the one, he was interested in following this case. It

was not the one that was going to work, and it was so clear. But that wasn't the message they wanted to hear.

So, anyway, after that I then got involved with something called experimental compiling system. I started that up, which was, in my intent, use the compiler technology to make compiler writing easier.

My goal has always been—and this is not a goal I've achieved—is to, through my compiling work, support languages that are useful by the applicator writer. Useful by the physicist, useful by the person who is solving a problem, and have a language or set of languages or tools, or whatever, that are natural for the way that person thinks about the problem and the way the person wants to express the problem.

In some sense, original FORTRAN program did that, original COBOL did that. They were very different languages, addressing very different users. And they really were attempting to provide a way of allowing the user to solve the problem in his own language or as close to it as it could be.

That's what I've always felt was the ultimate role of compilers: to hide all the details of the hardware in the

system, still exploit it, but hide that from users, so that they can get on with solving their problem and be comfortable with the results that they were getting, in terms of performance and cost time, and everything else.

We've taken a bad direction in languages and compilers. We didn't do it, though we could have, I think, done better, "we" IBM. But, the languages that came out at some point really were there, got popular because they didn't require (and actually couldn't support) an optimizing compiler.

It was: the user can do this all by himself, herself. And so they gave them the access to all the machine gorp and stuff, and it really put a lot of us in the compiler business, set our great technologies back a lot.

Anyway, when I did it, that hadn't really happened at that particular time. But, what I wanted to do with the experiment on the compiler system, and again there was a lot of good fall out from that, [was] to build a system that would allow compilers to be built for automated, for multiple kinds of source languages, or for multiple target machines.

The idea would be that you would be able to have a course system that could specify the characteristics of the source

language, and the characteristics of the target machine, and be able to then develop source languages around these. That could then be useful to different classes of users, and also very useful on different machines.

In IBM we do have current product, a wonderful variant of that notion, but it's not done in quite the same way as what we were envisioning there. So, I built up a group to do that, and we worked on it for a number of years.

We then got interested in parallelism, and in fact, it was Irving Wladawsky-Berger who came to me one day—he was head of Computer Science and Research for a short time. I first met him on ACS, when he was a student in Chicago; he was doing summer student work for us.

He just came to me one day in research and said, why don't you start some work on parallelism? IBM wasn't in the parallelism business, and so we got into it. Irving knew it was important, and this was really kind of the next big challenge in compilers, so I got into it.

I hired some great people from Dave Cook's organization at the University of Illinois. He was the father of a lot of the compiler work and parallelism, and we still have a great relationship.

We got into parallelism as a compiler problem. And over the next few years, we built PCs, we worked with quite a few IBM projects, the 390, the projects that had parallelism in the small level, the research project called research parallel RP3.

RP3: we did the compiling for that. And there are a number of others [for which] we're actually doing our own research, and at the same time building real compilers for real machines. That was a wonderful period too.

I had a PTRAN group, Parallel Translation Group, just a fantastic group of young people. These were amazing people. They were young from Illinois, and NYU, and they just published papers, produced code, it was the beginning of their careers. At that time, it was just exciting to be able to have that kind of group. And they were just opening up the same kind of thing that I'd been involved in on Stretching and in ACS—building their careers, building the ideas, things just broke open.

The work there was a stack of papers, a huge stack of papers, which had vast influence on the direction of the field. My role was very different—it was kind of setting the direction. You know, making the connections and once

in a while seeing an idea that I would say, oh, my goodness, that's the idea we've been looking for, for years. [LAUGHTER] It was great.

Into the '90s, at some point I had a manager who knew nothing about computer science, or compilers, or anything else in Research, and he canceled everything.

He said, IBM is going out of the compiler business—you're done. He didn't even have the decency to come to my office and tell me that. He called me on the phone. It was not one of IBM's finest moments.

LASEWICZ: This was the early nineties? The new IBM?

ALLEN: No, I don't think that was it. It was about '94 I guess it was. I wouldn't say it was the new IBM.

And we'd done dumb things before. [LAUGHTER] But this was pretty bad. Fortunately, I had been very active in the Academy, and so I ran for President. I ran and I became President of the IBM Academy in 1995 and started the transformation of the Academy.

The Academy had started in 1989 and I had been involved right from the beginning. It was designed to be, it is

designed to be, to represent the technical views to the rest of IBM and to address technical issues, mostly that come from the Academy—things that are falling between the cracks sometimes—directions we're not taking, [which] we know as technical people we need to take.

The Academy has about 300 of the top technical people in the company. I shouldn't say the top technical, but they're all selected for their technical expertise, selected by their peers, unless they're IBM Fellows, in which case that comes with being an IBM Fellow. And they're there to give advice to the executives and the rest of the company, on technical issues.

It was kind of a new role for those of us technical people. You know, the company's always been driven by the strategic goals that are set. And for products, mostly products, the business goals and the technical people had rules within their organizations, like IBM Fellows in Fishkill and what have you, [gave] a lot of input on what should be happening there.

What the academy did and allowed us to do was to make statements about things, independently of assignments. If we were given an assignment by an executive, we could decide if we weren’t going to do it or not. [LAUGHTER],

Which is a very empowering thing because we'd been struggling for a few years, between 1989 and 1995, with "were we an honorary organization," "how bold did we want to be." And I felt we should take a very strong stand on some things and do a lot of work moving it.

So, I started kind of a change in the role of the Academy. I became the first person who was, full-time head of the Academy. Before it had just been kind of a side job for everybody who had been president.

I got a lot of support from Jim McGroddy who was head of Research at that point. He was a big help and so were some of the other executives. And so, we changed the direction of the Academy. I started it, and then I was followed by another guy, and by a series of terrific presidents who have really transformed the Academy into what it is today.

And since then, I moved on to doing quite a few things outside in terms of boards in Washington, at the National Science Foundation and at the National Research Council.

I'm counsel for one of the boards associated with that and AE kinds of things, all those kinds of activities. Also, giving advice to anyone who would listen. [LAUGHTER]

LASEWICZ: It sounds like Blue Gene is one of the areas that you had some involvement in.

ALLEN: Yes, right. Blue Gene was to me another of these great projects that had come along. Somebody here almost never gets a chance to build a machine, build the software, build the applications, and deliver the application. It was a total system once more and on the high performance end. It was a terrific opportunity.

LASEWICZ: Full circle.

ALLEN: Yes, it really was. And a lot of in between was working on pieces of things, but this was once again a whole system.

LASEWICZ: Are there professional associations that you participate in for the technology field, the sciences?

ALLEN: I am a member of the National Academy of Engineers, that's elected, by members of the academy. And I do some things there. In fact right at the moment, I'm involved in helping out on a book that's being put out on the 20 greatest engineering accomplishments of the last century.

So I'm helping out with the computer section and the household appliances and on electricity. Each one had gotten a small committee of people. Anyway, that's one of the things I do there.

I'm a member of ACM, Association for Computing Machinery; I'm a Fellow there. I'm a Fellow of the IEEE, a member of the American Academy of Arts and Sciences, a member of the Philosophical Society -- I was just recently elected to that a year and a half ago.

And that is a great thrill, let's see, one gets elected to that for contributions to useful knowledge. It's a great group of people.

I'm involved with the Institute for Women and Technology; it was a small institute that was started by a former student of mine. This student had taken a compiler course from me at the Courant Institute at NYU, Anita Borg—she died recently. But she is a role model for all of us, I think. She started this institute, and I will continue being very deeply involved with that.

LASEWICZ: How have you found these associations to be of value to you either in terms of your career or anything else that they contribute, friends, networks...

ALLEN: Some of the associations, the networking is very important. I didn't realize the importance of that when I started, but it just kind of happened naturally because I was very active professionally at conferences, particularly those associated with ACM.

Early on I was involved with starting some conference, compilers and things like that. These networks are the foundations of my interactions outside, they're very important to me.

When Anita Borg died and I had gotten a call about it, I found myself writing to dear friends, to a very large number of people I hadn't talked to in a long time, but you know, we all thought about each other as friends.

I never had an official mentor and never thought of anyone as a mentor, but certainly there have been people that have guided me, going back to my parents, of course, and then the teachers, several of the teachers I spoke about. Also, some of the colleagues.

I think that John Cocke was certainly a wonderful mentor, one wouldn't name him that, but he was a dear friend and a

great colleague and gave advice occasionally that was awfully useful.

The same thing is true of my network of people. I think that mentoring is relatively new within the technical community. Within the people on the executive track, they're often assigned mentors, or even over the history of IBM, there were often mentors that brought along people that were viewed as having a lot of executive potential. But, I think in the technical community [mentoring] has become extremely important, particularly to the women in the technical community. They sometimes feel like they don't know how to deal with the environment they're in. And it's catching on in a wonderful way. It's still something we don't do well, don't do enough of, and don't understand. We're struggling with understanding it in the technical community.

LASEWICZ: Fran, we're focusing on women's issues today in technology, and one of the things I'd like to ask you is, what do engineers do? What do scientists do on a day-to-day basis in terms of their career? This would be from the perspective of: if you were a high school woman looking at choosing engineering as a career, what would you need to know about engineering as a career or science as a career?

ALLEN: Ah, that's a big question, because what we do is very much dependent on what work we're involved with at the time. I think the best way of characterizing what one does, is solve problems.

Now, sometimes in solving problems it involves sitting at one's desk. Lots of times it involves meeting with people, talking with people who know about the solutions, coming up with new ideas. So, it's understanding what one can do, what one wants to do and then spending time building it.

Now for the ones in software, that's probably going to involve building a piece of software. If it’s in hardware or a lab maybe, it involves actually putting something physical together.

If you're on a manufacturing line, it could be making sure the line is running correctly to make sure the processes are being done most efficiently; could be actually looking to see why there are errors, why things are not coming out the other end the way they should.

So there are many, many aspects of what an engineer or a scientist does, but it all boils down to finding problems, solving problems and getting results. And it can be very exciting, it is exciting.

Everyday can be very exciting. It can be drudgery sometimes. As one tries to get that next bug out, tries to figure out why something isn't going to work, one feels [like] hitting one's head against a wall.

But the really great thing I think about engineering and science are the colleagues one has. Now, ultimately it's up to the person how one solves the problem. Talking with colleagues is the best, one of the great ways of solving problems. Not reaching the solution, but gathering the information that one needs in order to reach a solution.

I have a colleague, a great scientist, who I always go to him when I have a problem or new idea and I want to see what's wrong with it. Because I know he's going to be able to—he knows a tremendous amount about everything and then he can always tell me every single "why it's not going to work." And he's wonderful about that.

But I also know that he's very pessimistic about things. So, I come out of my discussion with him knowing all the details that I have to handle in order to make what I want work. Or adjust what I want work.

One has to realize, you use your colleagues, you use the teams you're on, and you work with the teams you're on. Most things are teamwork. But in the end, it's your own, what you yourself brings to the solution, that's going to be important.

I'm glad you brought that up, because I hate to use the word competitive, but when one is making advances in an area, no matter what it is, it does involve a certain amount of competitiveness. In other words, people may go in different directions [and] have different answers, so competitiveness by itself can be very, very stimulating. But, it can also be a little bit overwhelming sometimes.

Again, one has to figure out how to succeed while competing. And that can mean knowing when to give up and knowing when to take what you know and look for an answer elsewhere, [and] why you're giving up and why you need to go in a different direction. And that can certainly happen. It's a choice one sometimes has to make. The whistleblowers don't always succeed in their careers, but they are respected. For me, being honest to myself has always been important, not just saying, “oh well, we'll just let that go.” I've always tried to do what I felt what was the right thing or the best thing. That sounds a little Pollyannaish, but I think I became an IBM Fellow

because I didn't always follow and didn't always do what I was told.

At the beginning it was wonderful. IBM has always been proactive, in hiring women and bringing women on board, and having reasonably good programs for them. But, of course, the situation one finds one's self in is not always the most conducive for success in the particular area.

When I first started at IBM it was really wonderful. I was in Research, as I mentioned. Percentage wise—and this is observational conjecture, collaborated by others who were there— more women, professional women, were in IBM Research, there was more room in research. Most of us were programmers and lab technicians and so forth, more women than there are now, or than there were a couple of years ago when we really had a big effort in IBM Research to hire more women.

I think the reason was that the computing field was totally immature at that time. There were not courses at universities, computer science didn't come into existence until the mid '60s and so in the mid '50s computer science did not exist. And yet, the field was building up rather rapidly, so people with all kinds of degrees, men and women with different kinds of backgrounds, were being hired. And

the rate of hiring made a difference in the fact that one didn't have to have any particular training in the [computer science] field, it didn't exist.

That changed actually in the '60s, mid '60s, as the field turned into a profession, as the people being hired had to have credentials, provided by having gone to engineering schools which, at that time, had something like two percent women graduates, in all of engineering. It was a period in which the number of women available to meet the requirements fell off rapidly, because it was a profession, and [also] because processes had been put in place. Certainly as a side effect of our experience with 360, where we were building software without processes, the engineers had some processes and they took over the software catastrophe that we were creating [LAUGHTER] and imposed processes. Along with processes comes management structure and management structure was built up. And by tradition I think, much of management has been white males—at that time, for sure. Women just hit glass ceilings everywhere.

I felt, in fact, that after the Stretch/Harvest the world was still all mine. I had been very successful as a manager. On that project, three of my four peer managers, first line managers, were women. This was around 1960, not

uncommon. But then after finishing with ACS, coming back, it was a very different environment in research. And it was very hard to succeed in the management level or in that way. In fact, one almost couldn't.

Year after year and I think I still do it, where I go to a big meeting of managers, I count the women and see what roles they're playing. It's still been an uphill struggle in the computing field, we know it. It's not just IBM; it's much worse elsewhere. And we're still working on it.

One of the themes that this Institute for Women in Technology has is 50/50 by 2020: 50 percent women in computing by the year 2020. That's getting to be a bigger challenge every year. Ultimately, we need to reach that in order to be able to have a comfortable environment for everyone.

It was pretty much a continual slide from the '70s, early '70s. This is just observational conjecture and people don't disagree with me on that.

LASEWICZ: Was the company's attitude toward women in technology pretty much the same during that whole slide, or was there any point where you would say, “they're making an effort, things are getting better,” and that maybe some of

the reasons for the continued decline were external as opposed to internal?

ALLEN: Well, in the early '90s—in '93 I'd say—as IBM became much more focused on our customer, it became much clearer that we had to look more like our customer.

IBM has kind of consistently had a wonderful history of attracting women, but the environments in which women find themselves vary a great deal, project by project, divisions by divisions, side by side.

Some places have been quite marvelous. I should say in my own field of compilers and languages, we own that, women owned that field at the beginning. We headed the labs. We had more, I'm not talking about research, there was a woman heading the Cambridge Lab. I think there may have been two in a row there. They had loads of women in it.

They were at Time Life—that was a center for compiler construction—and they...a lot of women were in all levels of management, a huge number of women there. The field itself was almost known as a field that attracted women. Now, of course, like attracts like, but it was early. If you look at Grace Hopper, who was one of the first compilers, and at Jean Sammet—an IBMer who should have been

made an IBM Fellow, absolutely—she was deeply involved with the COBOL side, and has written books on the compilers, or at least one. She headed a lot of that kind of thing, and was technically deeply involved with that whole era. There's a number of other women too, lots, I could just go on naming people. Women were very, very involved.

I walked into a birthday party recently, or anniversary party for some guy who used to work for me; he was celebrating 35 years. This was my old group in compilers, and I used to have lots of women who worked in my group. I walked in on this group; there were 40 people in the room, not a single woman. I thought, what is happening to this group? And that was last year.

I don't know what has happened other than, the focus has not been there and sometimes you have to keep the focus on it. Right?

[TAPE CHANGE]

LASEWICZ: What have been some of the greatest challenges that you've faced and how have you overcome them?

ALLEN: That's a challenging question. [LAUGHTER] Well, there are several big challenges.

One is that most of my projects, because the goals have been so high (some of them set by myself, some of them by others) have been wonderful experiences to be in, but real downers to not succeed. And not succeed at the moment when Watson says, I apologize for this project [LAUGHTER] and its failure. When ACS was canceled, that was a terrible downer. I think those kinds of things [have been challenges], not being able to convince people, management or others, of the importance of the work. In fact, one can't be out front and always win. In fact, one mostly will lose in our industry because the out front people will show the way but they don't necessarily have the business case, and that's what counts. Or, they don't necessarily have the understanding of what is really going to sell in the marketplace because the marketplace doesn't exist if you're really out front with something. There's no way of building a business case. Over and over again I have hit against that kind of thing.

Also, starting the work on parallelism, that was a very uphill battle within IBM. Even John Cocke didn't believe in...He believed in parallelism but didn't ever want to work on it. He went back and forth and back and forth on that. He wanted to continue to pursue the single instruction line. The parallelism side was something he

knew needed to be worked on, but then he would dismiss it. Being in that kind of situation almost all the time, of working on things where I would be fairly certain of the vision and ahead of the others' thinking, and then failing to be convincing about it...

Sometimes it was a matter of my not doing the nuts and bolts that needed to be done in order to sell it. People would have great faith in me, in the idea, but then in the end, when all of the business people needed to [LAUGHTER] it would be shoveling.

So, in the large scale, those would be the things I found most frustrating, but having the most rewards at the same time, having been able to do a lot by trying, by setting high goals.

The other part I think was the period in the seventies, it was very clear that women were not going to succeed in that era. That was the first time I really hit "you can't go any further."

LASEWICZ: How did you deal with it at the time?

ALLEN: Oh, it was not easy. The problem was that I didn't handle it very well. I was angry. I wouldn't say I've always been [LAUGHTER] liked by management [LAUGHTER]. I've had great managers, who I have huge respect for. But there was a period in there when I felt that things...I was angry. I became the angry woman [LAUGHTER], which is one way of not succeeding in things.

LASEWICZ: Looking back would you have handled it differently?

ALLEN: I'm not sure I would have handled it differently. I don't see how I could have, but in some sense I did by just sticking very close to the technical area and succeeding on the technical line, and hammering away on that, knowing that there was no possibility on the executive side, beyond the first line.

LASEWICZ: One of the things that struck me surrounding your publicity was all that you do outside the company.

ALLEN: Yes.

LASEWICZ: That raises the question of, how do you manage to balance your personal life with your work life? That's

something that challenges a lot of people even today, and you seem to have found a way of doing it.

ALLEN: Well, I can't answer that question very well, because I don't have a family. I was married and I had stepdaughters, and that was a challenging time, I understand [LAUGHTER] what's involved.

But that lasted about 10 years, and so I haven't had to balance the family issues nearly as much as so many of the people, particularly the young people. It is really a tough problem.

I guess in retrospect, I've always taken big vacations. I get quite intense about what I'm doing and get very involved with work, and then I take my vacations, which have usually been climbing mountains or skiing or doing a lot of exploration. I particularly like exploration.

I've been to the Arctic, the high Arctic in western China. And I continue to do some of that. That was a very intense experience, because in many of these situations, particularly before the radio systems and GPS and good maps and all of that sort of stuff, one knew "I always stay with my group," mostly with groups of friends, and that one's survival depended upon being careful, in very high risk

situations sometimes. So that was one way I got away from work totally [LAUGHTER]. There was nothing more important than knowing how I was going to survive over the next period of crossing a field in the Arctic, high Arctic, or way out in western China where there was no possibility of anybody coming in.

In the winter for many winters I would join some friends and we would have a helicopter drop us off in British Columbia in the mountains, and we'd camp out in the snow and do ski mountaineering for a week and have him come back and pick us up [LAUGHTER]. One doesn't make mistakes in those situations.

LASEWICZ: You need to be focused.

ALLEN: Yes.

LASEWICZ: What do you consider to be your most important contributions to the fields of science and technology?

ALLEN: I think it's probably the...well, I could talk about the technical contributions, you know, the theory and practice of compiler optimization. But probably more than that, it's the next generations, at least one or two that have come along that have picked up on that work, the many

who have worked for me and are contributing immensely today.

So and I think it's in establishing a piece of the field and then working to develop it through others. Getting parallelism into IBM was a big one, [LAUGHTER] but actually, then motivating others to be part of it.

LASEWICZ: Just to follow up on parallelism: that was a case where the business case became apparent fairly soon, and you impacted the SP product line with the work that you were doing?

ALLEN: That was later, but yes, that was a continuum. Yes.

LASEWICZ: If you had to pick an item or two that has made you successful, what would it be?

ALLEN: I guess it would be...well, it's easy to say, right place at the right time and all that. But I think it's because I have had a lot of confidence in my own combination of vision and pragmatism.

I work both sides of that. And I think that that's served me very well. It's not just knowing what the vision is out

there, but having deep concerns about how to get there, the pragmatic side of it. I'm an extremely pragmatic person who has high goals, and I think that has been a big factor. I like to know how things work, I have to know that something that I have developed is going to be useful in a product, it's not just a nice pretty result. I think that that comes from the very early experiences of doing research, as we all did at that time, everybody, on these high goal projects.

LASEWICZ: Well, we've covered an awful lot of ground, and so I suspect there's probably not a heck of a lot that you'd like to add, but do you have any insights or anything that you'd like to say that we haven't talked about, that you would like to add to the record?

ALLEN: I think I would like to talk a little bit about the future for women in computing. It's very hard, at this particular juncture, to understand where it's going to go. I mentioned trying to have 50 percent of women in it by 2020.

I've looked at the education issue and at the NSF data recently—the number of women graduating with Bachelor's Degrees and Master's Degrees, with PhDs, since 1966. We're not doing very well there.

But, I think that where we are right now in the computing field is that the future is ahead of us. What has happened, up until about the nineties, was preparation for the future. We went through with the integration of communications, information and computation—those three pieces got integrated in the mid '90s in a very real way, which brought information and webs to people's homes, to desktops everywhere.

We haven't absorbed that tremendous transformation fully yet. We're still working on many of the pieces that are going to make it work well. In this is going to come a tool for women and a place for women, globally, which will enable them to be much more active as entrepreneurs.

As of a few years ago, more than 50 percent of the new businesses were being run by women in the United States. Probably has changed, but I think that's going to continue overall. And I think that women need to realize it's not about the technology that's in there, it's about how you use it.

We're moving into a "how to use this," this technology that we put together in the first 50 years. Women are, I think, flocking to how to use this, how to take advantage of it,

how to change society, all of those things. And this is a great time for that.

I'm less worried about women not taking computer science, than I am about women not being ready to understand the opportunities that this technology, that we've put together, will enable them to use. We're at the beginning of the big result.

LASEWICZ: Great. Well thank you.

[END OF SEGMENT]