Oral History of Fran Allen

Interviewed by:
Al Kossow

Recorded: September 11, 2008
Mountain View, California

CHM Reference number: X5006.2009

Al Kossow: Hello, this is Al Kossow. It's September 11th, 2008. We're in Mountain View, California. This is tape number one of an interview with Fran Allen. Thanks for stopping by.

Fran Allen: Well, I'm delighted to be here.

Kossow: Is the first time that you've been to the museum since it has moved?

Allen: Oh, no. I've been here for a couple of events.

Kossow: Oh, okay.

Allen: And came here soon after you got the Stretch machine. I think you came down from Livermore.

Kossow: The Livermore machine, yeah.

Allen: The Livermore machine, and didn't know that it had been moved here and was so delighted when I walked into the room and there it was.

Kossow: It was in the big storage room in Moffett, all the cabinets packed together--

Allen: No, it was actually lined up in the room of machines with where it was-- I guess, they ordered by date. Yeah, that's when I saw it.

Kossow: Tthere was a nice oral history that Paul Lasewicz did with you in 2003, so, what I want to do is just start out with what's happened since 2003. I know there was the Turing Award. What have you been up to in the last five years?

Allen: Well, I retired from IBM in 2002 after 45 years with the company and retired just because I wanted to manage my own schedule. But I had an arrangement with IBM, I'm a fellow in my emeritus position and I still have at IBM Research, the office I had before I retired and a lot of my papers are there. I go in pretty often. And so that's one of the things I do, but winning the Turing Award really has changed my life. I've gotten invitations around the world to give talks about the work I had done and I've been thoroughly enjoying that. Mostly, it's from universities, it's from companies. And I talk a lot about the history of computing because I started in 1957, so when I trace my own history from when I first started and relate what my own experience is to pretty much to where we are today, and it's not a definitive history, it's my history, but it is technical. That's what I've been spending my time on.

Kossow: Have there been any particular events then that you remember in specifics from the last couple of years?

Allen: Well, the Turing Award event, receiving that, and the lecture. The lecture I am giving now is envolved since then because there's been some new technical issues that have come up that I've become quite interested in, something called micro multi-cores and the parallelism that is becoming the way that we're going to get performance, software parallelism. And so I've kind of been looking into that and attending a lot of lectures, learning a lot about that and relating it back to what I understand about that particular topic from experiences I have had with high performance computing, which is where I spent my career on compiling and languages for high performance computers.

Kossow: Actually, one of the things that I wanted to talk you about were your opinions on what's going on with the potential for thousands of cores on a processor chip and just how we're going to deal with that from a software perspective.

Allen: I have a whole lecture on that and it's going to be extraordinarily challenging. I attended a little workshop recently on the 50 billion transistor chip and that's going to come. And how those transistors will be used, how many thousands of them will be little processors, how they'll be used for storage on the same chip is unknown. But all of our software's got to play a very much bigger role and it's the biggest challenge that we have. And I think-- and a lot of people are calling it a challenge, particularly John Hennessey had a statement that it's the biggest challenge computer science has ever faced, but my view, and I entirely agree with John, but my view is that it's also the biggest opportunity we'll ever get to actually reset a lot of the way computers are used and what their role is in society. It's kind of an interesting time. It's an inflection point for computing and everything is going to change. And there are a lot of good reasons why it should change. We've grown up fast in computing. It's used everywhere. It's pervasive but it's still pretty immature in a lot of the way we interface with computing and with how robust the computing systems are.

Kossow: So, do you think that these systems will stick with a von Neumann model or, just because of the number of transistors that are available, that we can move away from that to data-flow models?

Allen: Well, I think there'll be multiple models and it will very much depend upon the applications. But I think we don't understand the models themselves very much and so I think there will be data-flow models, there are streaming models, there are transactional models that are being looked at very closely as a way of solving some of the problems, and then there's the traditional models that we've had for computations. And I think that what we will see over the next quite a few years is an evolution going on in understanding models of all sorts and how to put them together and have them effective. One of the things I think the other thing that's happening at this particular point, other than just the number of transistors on a chip and the fact that each one is going to have to be probably less capable because of the heat problems of that comes up as a result of the miniaturization that's been going on and the fact that we have put a lot of capabilities on chips, which gets using up a lot of energy, and that's going to go away. But the physical thing there is one issue that's going to be driving what happens, but another issue is that we've brought together a capability built out of many, many, many separate capabilities. Computing grew up with computers, computational capabilities, separate from communications and in separate PCs and its being integrated into one commodity. And the integration is still very awkward. And I think that that's as big a challenge as the kind of multi-core thing, is the how do we integrate these very separate capabilities.

Kossow: That gets into system complexity and programming paradigms and just how do we--

Allen: Yes, how do we do that and how do we overcome-- we've got a lot of problems right now, the massive amounts of data that are available and how do we manage that data, how do we, in some cases, eliminate some of that information. And that's not my area of expertise, but it's I think quite an overwhelming-- how the growth of the future will evolve unless we have answers to some of the problems of how we communicate and manage data, right.

Kossow: The raw data problem gets back to Harvest and Alpha and the work that you were working on this in the early '60's. The Harvest system is really unique at that time for the amount of hardware that they threw at the problem of pattern recognition.

Allen: Yes.

Kossow: One of the things I've been curious about is just what the software system was like and did NSA specify what the Harvest box was to look like or how did that evolve, the hardware and software that the NSA wanted for cryptanalysis and pattern recognition?

Allen: Maybe I should mention a little bit about the Harvest was associated with the Stretch machine, which was being built for Los Alamos and for some of the grand challenge problems of the time, and which was a more standard machine, though it was a huge amount of number of things had to be invented for it to work. But Harvest was an entirely different attachment. And there were two parts to Harvest, it was the piece of hardware that did the analysis, and that had nine instructions, actually eight functional instructions. And it was looking for patterns in data and any instruction could run for days and it was programmed out of kind of a micro level and it was like setting up timings in the instructions and so things, these bytes of data, would pass by an instruction instead of doing comparisons and collecting up data, and that would be one set of instructions. Another set of instructions for Harvest was an instruction for sort and so they were just very different than the kinds of instructions that we think of in the normal arithmetic unit.

Kossow: And this was operating on billions of characters because of the Tractor.

Allen: Billions of characters, and that's where what was one of the most amazing parts of it was that the I/O of the time was pretty primitive, but there was a system that was called the Tractor tape, and I think I've got that information with me, but is the-- let me back up a little bit here. What this was doing to Cold War and the main thing that Harvest was built for was to analyze data from listening stations around the world and it could-- let me look up here.

Kossow: So, was it voice traffic or teletype traffic or just what sorts of--

Allen: I'm not quite sure what it was because I wasn't associated with that end of it. Let's see, I have somewhere here a--

Kossow: In Paul's oral history, you mentioned that the machine wasn't actually in the Fort when you were working on it, but it was--

Allen: What's that?

Kossow: It was off of the base when you were working on it initially?

Allen: Oh, it was in the basement. No, it was in Fort Meade.

Kossow: Oh, it was on the Fort.

Allen: Right. So, there was a tape system on which this was stored, which had a capacity for 3.3 million bytes per second, doesn't sound like a lot. But the most interesting part of the whole system was the fact that the data transfer rate to and from these tapes through the storage of Harvest-Stretch, and then from there through the Harvest analyzer, were completely in sync. So, data could flow off a tape, or the whole tape system, and through the memory, through the analysis and information recorded back on the tapes about what was found, without any stopping and doing, actually doing I/O, or doing loading. It just set up the system and it would run and could run for hours and days.

Kossow: And there were enough tape units that it would just stream across the tapes and the operators would mount the--

Allen: No, the operators of those tapes were never touched.

Kossow: Oh, so it was an early tape--

Allen: It was a big cartridge system. It was great giant cartridge system where the tapes had addresses and were automatically programmed to pull up a tape, bring it up to a reader and then take it off, then read it. But I think it's the only system that has ever been built that had the I/O, the memory and the analysis unit in-- I think of it as being on the same clock, but they were completely in sync. And that piece of it was designed by a man named Jim Pomerene, he's thankfully still around and doing well. And he was the engineer on John von Neumann's machine. He was hired by IBM for this project and had a long relationship, well, for the rest of his life, rest of his career, with NSA, liaison between NSA and IBM.

Kossow: So, was the operating system on the Stretch-Harvest system an extension of MCP or was it completely unique to--

Allen: It was called Hops. It was kind of built on Harvest operating system, and it was built to be compatible with the Stretch, with the MCP, I guess it was. Now, but Alpha was the language, the high-level language, that the analysts at NSA would write their programs in, and that was way that was. I was

involved with the Stretch Harvest compiler and with the liaison between IBM and the analysts at NSA on the design of that language, though I knew--

Kossow: How did Alpha end up being designed? Did they give you a set of requirements and then you tried to match it to the hardware?

Allen: It was designed to match the hardware, basically, and match their problem. And it was a very high-level language in one in which they could very succinctly describe what the problem itself, and it could do that in part because translating that language by the compiler to the machine was pretty much straight forward, but because the whole system had been designed to solve this one problem.

Kossow: Was there a pretty close correspondence between the instructions in the language and the instructions on the machine?

Allen: Yes, it was a pretty straight forward mapping of that.

Kossow: So, let's say you were going to do a keyword search for 'foo' in a data stream, how would you do that?

Allen: Well, what that would be, 'foo' would be considered like one data stream that just was held, and then there was one instruction called Stream Byte Byte Byte, which always took it and put two data streams, or one of them could be fixed and 'foo' would be fixed, and the streaming then would be managed by how the comparisons were done was expressed, say, in forms of how many K bytes you want it to look at at a time, you know, eight bytes at a time, 10 bytes at a time or something.

Kossow: I'm just trying to get a feeling as a programmer what it would have been like to use Alpha.

Allen: Oh, well, in preparing for the talk I gave here a few years ago on Alpha, you wrote a little program in Alpha to do DNA mapping and looking for patterns in DNA mapping. And I think I had about 16 or 18 lines of code. It really was very, very trivial to do that in order to do a complete map. It was so straight forward. Oh, and one other thing I have to tell you about Alpha, the data itself, of course, the sources of the data had from these listening stations around the world, and it wasn't just these in the Cyrillic alphabet, they could be in any alphabet. So, one had built into the language, into the total system, was a way of mapping a language to numbers and you could actually specify that in the Alpha language. This is the Cyrillic alphabet and not in terms of how the alphabet looked like, but if you saw this number and it was the ordering--

Kossow: The symbol mappings.

Allen: Yeah, the symbol mappings, and then that was what was used. So, you could specify supposedly the Chinese alphabet, but I don't know how that worked but certainly the other alphabets, all sorts, could be specified in the program as to what kind of data it was and what the ordering was. And we had to

introduce two special, in addition to the members of the alphabet, we introduced something called a Pad and a Scab. A Scab was when it was unreadable, when you had bad data, and a Pad was for in between pieces of data, so chunks of data. So, you'd have words that would be split by Pads, by blanks essentially, and also might contain a lot of dirty data. And it was part of what the whole system would work on is recognizing-- since it was statistically based, working on filtering out the bad data, as well, because from those listening stations there was a lot of it. Even though I had a high clearance, I didn't know anything about a lot of these details until much later. Part of what I had to do as the liaison and in the bring up the compiler in the Harvest machine for the Alpha system was to also write the acceptance test for the Alpha language. I knew that I couldn't leave Fort Meade until that was done and I dreaded it for the whole year I was there. And the acceptance test was to be automatically abstracting *Time* magazine articles because that's, you know, kind of the analog of what the real problem was about.

Kossow: So, was it keyword strings?

Allen: Yeah.

Kossow: Oh, okay.

Allen: So, I got the method from one of my colleagues about <inaudible> and did it and it wrote again it was just a few lines of Alpha code and it worked. And it was like I was expecting to spend another month in the basement of Fort Meade. Instead, the second or third time around it produced a wonderful abstract of these articles.

Kossow: So, how many software people at IBM ended up working on Alpha and the compiler side of the system?

Allen: Well, it's a little hard to break it all down, but the actual bring-up team, the team that was down in Fort Meade, assigned to Fort Meade, which consisted of about, I think there were 10 of us, and that covered the operating systems, as well as the compiler and some other problems, issues.

Kossow: One of the things I was wondering about, because this is an early example of data mining, is did any of this knowledge make it out?

Allen: Nothing made it out, not even my final report, which I forgot to make a copy of before it was stamped confidential.

Kossow: Just all of this knowledge that went nowhere.

Allen: But that's all been declassified now, all of that has-- pretty much all of that has been declassified.

Kossow: So, as far as you know none of the people that worked on this did any further work.

Allen: No.

Kossow: That's a shame.

Allen: Yeah, but it was such an unusual system, too. I mean, there weren't many-- because it had the Tractor system, that piece, and the, you know, and the big piece of hardware, that was a piece of hardware that was much larger than Stretch, and just in physical size and very, very unusual piece of hardware and also the Alpha language and so forth.

Kossow: So, then I guess this would have been '64, '65, you started working on ACS?

Allen: What I did after that, after coming back--

Kossow: So, the acceptance was in '62?

Allen: '63, I think it ended up. Yeah, right.

Kossow: So, then ACS started to ramp up in '64, '65?

Allen: Yeah, it was in '63 I came back from that, I think it was. In fact, the Cuban Missile Crisis was in the fall of '62, and that's when I was down there and it was almost a year later that I returned. That was an interesting day at Fort Meade.

Kossow: Is there anything you can say about it or just everything got locked down?

Allen: Well, when I walked into work that morning, it was into Fort Meade that morning and, of course, it is an interesting place to be because one couldn't get to the machine room without passing pairs of, not just on that day, pairs of marines with guns in two or three places and then hundred digit combination locks that got changed every once in awhile, so to get into where the machine was. But on that day, I walked into the building and immediately knew something was wrong. It was visceral that there was something very seriously wrong. And when I got to the machine room, the head operator, whom was supposed to be on vacation, was there and not talking to anybody and the printers were all draped in black because nobody was to see what was being printed. And I think there was a few hours, or maybe a day, between the crisis and the point it was kind of you do not talk to the people. But it was interesting to know that something was going, something very serious and not to know what it was.

Kossow: So, we made it through that.

Allen: What?

Kossow: So, we made it through that.

Allen: Right. But the Harvest machine was a huge success, I was not wanting to leave that machine for a bit, but it was a huge success and they eventually a part for Tractor tape was needed and nobody was willing to supply it or something. Anyway, they were very pleased with it. There was a final report that a man named Snyder wrote and that's publicly available, I had that. At the 10 year anniversary of the system, there was a party down in the machine room and it involved a small number of people, but including the heads of IBM were there, Cary, I think it was, and some others. They were our biggest customer for awhile. Maybe they still are, I don't know. And they had every kind of machine in that basement all partitioned off. Anyway, so when I came back-- that's what you had started to ask about. When I came back 360 was in-- 360 being put together and I was asked to join that project. It involved a lot of travel and I decided I didn't want to do it. It would involve PL/I and some other things. But, John Cocke was in research and was starting up a new project, it was called Project System Y at that time. And John had been on Stretch and he was interested in building the world's fastest computer. And of course, Stretch was targeted to be 100 times faster than any other computer. And John had contributed to the hardest part of that, the look-ahead unit. And so he was very interested in going ahead with the same challenge on a new computer, things we had learned from that and he asked me to join that project and work on the compiler. There was three, four of us on the compiler there. So, I returned to research.

Kossow: Had the 6600 announcement been made by then or was that still in the future?

Allen: I don't know. I think that may have-- I don't know. It must have been made by that time, yeah, come to think of it. Anyway, that was another nice project. But on Stretch and Harvest, we had, you know, really pushed the envelopes but we were really inventing all the time when under great deadline pressures because of the short delivery requirements. And so it was kind of a wonderful time to consolidate, come up for air and we had learned a huge amount of things in all aspects of--

Kossow: And circuit technology had moved on so it was better circuits to--

Allen: Yeah, and the base technologies had moved on.

Kossow: One of the people driving this was Livermore?

Allen: For the--

Kossow: For ACS.

Allen: For ACS, not driving it in the way Los Alamos had driven Stretch, and Los Alamos was very much embedded in designing some of Stretch, particularly the numerical analysis aspects of it. Of course, for Harvest it was dedicated to NSA. But Livermore, it was not-- we worked with Livermore. We also worked with Los Alamos, it was a good arrangement for the kinds of things we wanted to tackle. And I was

working on the compiler, as I said, and that was another wonderful project in the sense that it was the team stayed very small.

Kossow: Let's take a break and launch back into it on the second tape.

Kossow: This is tape number two September 11th, 2008 of an interview with Fran Allen. So, we just had started talking about ACS and John Cocke and the work out here. I guess it was the ACS-1 and the ACS-360?

Allen: Oh yes. I forgot. Right. It started this System Y in IBM research as a small group. Then- I forgotten what year it was. We built an experimental compiler, which before the machine- to inform the machine design.

Kossow: Was it a Fortran compiler or another?

Allen: It was language independent. The core of the compiler was language independent and machine independent because what we wanted to do was experiment with what the characteristics of the machine were- would be; like number of registers, depth of buffers, that kind of thing. It was basically Fortran-like but there was no intent to restrain it to Fortran. So, the core of the optimizing transforming program was designed to be independent of either the language or the target machine. And that's where the- that piece of work was the genesis for the piece of work that led to the optimizing compilers of that nature, which continue. And that form is used today very widely.

Kossow: Like the Fortran-H compiler? Any specific IBM compilers you could think of that directly evolve from that?

Allen: Yes. Well, it certainly had a history from Fortran, the original Fortran. But the IBM Product compilers today, they're called the XL Family of Compilers. And they support- this family has the core of the optimization analysis. It supports all of the IBM product languages from Cobalt to Fortran--

Kossow: So, this was the origin of PL/C?

Allen: Yeah.

Kossow: The Toronto Group?

Allen: Yes, the Toronto Group. Right. And all of IBM's platforms. Now some of the- there's a lot of special compilers also, but that's the core high-performance compiling system. It came from- and several incarnations before it got that far and it continues to evolve. So, that comes out of the ACS work basically. But it was also that which drew very kind of heavily on the work we had done at Stretch and very directly on the Fortran project originally. So there is a whole line of- the compiler work has a very

long line of where the current system is going back to Backus, basically, and through his work. So that was one of the pieces of work we did before it became ACS and moved to California. It was a conference, which brought together both the research people and some of the IBM people and a couple of people from universities at a workshop to move it from what we were doing and wanted to do in System Y. Move that technology to product. So the move to California and they renaming it ACS was a move to product. And that's when Amdahl joined- came in. He hadn't been in research. We moved out here. And the idea all along was to keep the core group very small, and be frankly, independent of IBM. We were subcontracted to Motorola, the chips. We subcontracted the compiler work that we had based upon what we had already done to Computer Sciences in Southern California. We just subcontracted a lot of parts of it. We did a lot of analysis; in addition we did some great work on branch prediction and on caching. That was where a lot of analysis was done on cache protocols and all of that. It was a single instruction stream machine, one instruction counter. We had a lot of kernels that we worked with to validate the effectiveness of the proposed machine designs via a simulator. By the way, that simulator, another great piece of work that John Cocke did for Stretch/Harvest, was a simulator. He and another person, it may have been Harwood Kolsky, wrote that. I've forgotten who was involved. But it was John and Harwood or if it was John and someone else.

Kossow: So, it would simulate the pipeline?

Allen: Yeah. It could simulate at the minor cycle level or major cycle level, the pipelines and everything.

Kossow: What were the major technologies that came out of that compiler work? You were doing flow analysis then?

Allen: Yeah. It was the control flow analysis and data flow analysis and all of the collection catalog of optimizing transfer informations, and the theory behind them. It was all done quite abstractly. But it did build on- the first thing on what we, on what IBM, worked on. The Backus work on Fortran. Because they had already identified the pieces of the compiler that still exist today as distinct parts. The parser, the register allocator, the control flow analyzer and the common sub-expression elimination as an example of one of the optimizations. And all those are their terms. So, a lot of had been done.

Kossow: That was the first thing you did at IBM, wasn't it? You were teaching Fortran?

Allen: Yeah. Teaching the scientists and research Fortran II. Because there was going to be an edict that they had to use it. I came in the summer- joined IBM in the summer of 1957 and the Fortran compiler had been made available for use in April of 1957. IBM Research management decided that their own scientists had to use it.

Kossow: Have you worked with computers at all at U of M? Had you had experience with computers?

Allen: Yes, yes. I had taken a course in computing and used a 650. Yeah.

Kossow: Going to IBM and then being thrown into the insides of Fortran compiler must have been daunting.

Allen: Well, it wasn't so bad because of the way I had learned to program on the 650. We first had the usual ways it was being done at that time. You coded at the hardware level almost and then you had to code it at a higher level. Then there was SOAP, which was considered the highest level at that time.

Kossow: Did you spend much time with the people that wrote Fortran? I am trying to understand how you got to know the inside of the original Fortran compiler; how you ended up in programming languages.

Allen: The listings were easily available. Though John Backus and his group were not IBM Research at that time, or they may have been at IBM, but in a different place. But one of the things that I remember running- discovering what I thought was an error in the compiler because it had collapsed three loops into one. I thought, "Ah, that's not right." And we were looking at the assembly code from the original source and it turned out that they had an optimization in their compiler that has almost never been done since. It is they recognize the way the data was stored, it was a three dimensional matrix. The fact that they knew the size of each component of the matrix, that they could just go through storage with one loop.

Kossow: It was nested three deep, so you could just flatten it.

Allen: Flatten it. Yeah. It's the storage layout and knowing that exactly- that it looked at all the data. So, that was- then from seeing that and looking back into it-- Also people were just reading code at that time. Another project that I was involved, right afterwards I think, was reading Roy Nutt's code for an assembler on the 704?

Kossow: FAP?

Allen: Yeah, FAP. That's one of the ways I often learned programming was to really look at great code. That's the way I liked it and always liked to do it.

Kossow: People still do that today.

Allen: Yeah. Find a great coder and borrow his code.

Kossow: So here we are in the middle of ACS.

Allen: Yes. And then the project got cancelled as is well known, which was really distressing But in retrospect, I don't see why they let us go as long as we did. In part, because the company had poured all its resources on to 360. Of course, it came out in 1964. And then everything was focused on that, and it had to be. And we were running very counter to them, to the 360, in every way. It was not instruction set compatible whatsoever. We were going outside to get the technologies. It was going to be very costly,

as any high performance system is. It is very hard to get payback in that marketplace and IBM over the years had sometimes gone in and out of being interested in high performance computing. Fortunately they are in it big-time now.

Kossow: But then they had the problem in the early 70's that they had no competitive machines.

Allen: Yeah. That's right. That's right. So, it was a tremendous blow to all of us. It was much less of one for me because first of all, I was in the software. I was just happy to go back to New York. But I used that little period there to write the compiler work and that served everybody well. John did too. Well, John never wrote so we took care of that, getting some of his ideas written up.

Kossow: Then you come back in '67, '68?

Allen: It was- I actually came back-- I actually worked for Gene Amdahl for a year. Gene then got the ACS-360 going. And I was reluctant to let the compiler work die. So, I got the head of compiler work in IBM, who was located in New York at Time/Life Building at that time, and convinced him that the work should go on and that we could work on it on the ACS-360 project and stay out here. But, that all kind of cratered after awhile. But that was a nice period because again, it was a period of trying to take what we had, what we knew as good. We had a lot of timings to prove it. And it was a lot of new technologies in those compilers, and the work to trying to move it again to product. What I have always liked is working on both sides on doing the kind of theoretical work but proving it in the product. Getting it to product. Repeatedly, it has been that aspect of the work that I enjoyed most.

Kossow: How did you finally push that technology into the product compilers? You were in California and all the compiler work was being done in the east. Did that happen when you moved east again?

Allen: Yes. But I took a sabbatical then; the first of two sabbaticals I've taken. I went to NYU. There was a lot of compiler work going on there. And then returned to IBM research. I decided that I wanted to use the compiler technologies to make building compilers simpler. It has always been an intriguing question since the mid '50s about compiler compilers and that kind of thing and so I had a project on that. I also called it the experimental compiler but it was a compiler compiler and the target was going to be PL/I. Unfortunately, if I had something, a simpler language, it probably could have been quite successful. But it is still referenced today. The work that the people in the group did is still held up as advancing. I could clearly see that early on that PL/I was going to be too difficult to build a compiler for PL/I out of using PL/I in a system that actually constructed itself out of basic definitions of right down it. What is the floating point operation doing? So, you put together information- that kind of thing. But there was some good theoretical people on that; Larry Carter, Bill Harrison, that have pushed a lot of that. Much further than what I was willing to at the time because I couldn't see that we were going to be in trouble with the real practical issues. So then I went from there to a some other work. All of it related to compilers and mostly related to getting compilers into IBM products. I had a PL/I project going with a company doing a PL/I optimizing compiler for the laboratory out here. Working with Intermetrics in Boston on it. Then doing some vectorization work. So moving around and getting--

Kossow: Did you have any involvement with Intermetrics and Hal-S or the Space Shuttle work?

Allen: The Hal-S had happened before. I was up there at the time when they had the problem with it. Wasn't that the one with the problem? The Space Shuttle that had a computer problem in flight? I remember where the three computers on board-

Kossow: Where they disagreed.

Allen: Yeah, disagreed. <Laughs>

Kossow: That was fairly early on.

Allen: Yeah. I remember a lot of discussions on that. Yeah. They was a good group.

Kossow: We're sort of in the Future Systems timeframe now. Did you have some work in Future Systems?

Allen: Ah ha! Now there's a topic. <Laughs> That's true. I've kind of flipped over Future Systems. <Laughs> Yes. I was involved with future systems for a short time. That's right. That was the first thing I think I did when I got back from NYU was Future Systems; was got assigned to- was got into the Future Systems group. And realized very quickly that it wasn't going to work. And wrote a letter..

Kossow: What part of it didn't you think would work?

Allen: In order to be able to access something from memory, it took two or three accesses because the interface with the machine was essentially kind of an abstract machine of some sort. So if you had- we're talk about A(I). Then you had to access I. And then use that value to access A(I). So, it was two round trips to memory to storage in order to be able to get a datum that you wanted, at the minimum, unless it was constant or something. So, that's where the bottleneck is. It's always been there in high performance systems. I discovered recently that that bottleneck was known to Turing. And so I wrote a memo saying that it wasn't going to work. I already had talked to one of the managers, actually he was the head architect on it, who kept the architecture on sheets of yellow paper in his bottom drawer of his desk for months. And whenever you found a problem, say, "Well, this won't work for COBOL." he would come up with a different answer. It's was all incompatible. It was a mess. So I finally wrote him a letter saying to the overall person saying it's not going to work. And cited the particular memory bandwidth problem that was just being exacerbated by the…

Kossow: He had no performance analysis people working for him?

Allen: Yeah. <Laughs>

Kossow: And this was going to be the future after the 360?

Allen: <Laughs> And it tied up IBMers around the world. It wasn't this small project. Nobody knows how much money it wasted but it certainly wasted a lot of opportunities in that period. So, I got shelved. I didn't get any raises for a couple years.

Kossow: Meanwhile, John [Cocke] is off and building 801?

Allen: I don't know what the timing there was.

Kossow: 801 was mid 70s?

Allen: Yes. But John was on a couple of other projects in there. One I haven't been able to find out much about. Some people out here know about it. It had to do with telephone switching. John was always interested in that among as many things. Signal processing systems was another thing. Anyway, what John took on was another goal. Instead of the fastest machines in the world- being working on it. His goal became the best cost/performance machine instead of the highest performance; just cost/performance. And that's what drove along with his experience in the early ones- what led to the PowerPC. Again, it was also tied- they built the compiler first. It was very, very nicely done. I think if they had also designed the language, a cut down variant of PL/I. It was called PL.8. That that language, solved some of the pointer problems in PL/I and interrupt handling problems. And I think if IBM had published that language, made it public.. It didn't, which was a terrible mistake. We might not have C today, which IS a terrible mistake. <Laughs> So, I don't know. One doesn't know. But it was very unfortunate that that language was not published. it was not available. For one thing it was could constrain storage management problems to a fixed space, for one thing. In PL/I, you could just wander all over the storage with pointers. With this one, you had to specify if you were using a pointer based data structure, you had to allocate an area in which it would be used.

Kossow: But it wasn't as strictly typed as Modula or Mesa or..?

Allen: No. No. But it was good. It was very good. And very compilable; could produce very, very good code for it.

Kossow: Were you involved in any of the 801 project or the spin-offs?

Allen: No. No. That was the time when I was more interested in applying the compiler technologies in a different way. That project was oriented towards a machine system. I was more interested in moving the compiler technologies towards making- towards better tools and compiler/compiler systems and that kind of thing.

Kossow: When does the Toronto [Compiler] Group get started?

Allen: They got started as a result of moving the work on PL.8 to product to support the PowerPC, which was built in Austin. A bunch of people went down to Austin and another group worked very closely with the Toronto group.

Kossow: Now we are moving into when you really got interested in working on parallelism.

Allen: Yes. I don't know which what came first. But there was a project at NYU, the Ultra Computer Project, which was 256 processors, single instruction multi data machine. But separate processors supported by DARPA. Very fascinating early SIMD parallel. And IBM was involved with building it but it got cut back a lot. Then it kept dragging on for quite a while. The processors themselves went from 256 to 64, so that way we didn't have to build a big combing switch in the network that supported it. And the processors became somewhat obsolete. But I was involved in the compiler for that.

Kossow: Originally, it was supposed to be a full crossbar?

Allen: Yes. More than a crossbar. Because at 256, they really needed to have- I don't know how to describe it, but an intermediate staging area so that it didn't go through the full crossbar. Yeah, probably in their combining switch in the middle of that. And my group did the compiler for that machine. That was a fun project. And that we worked with the Toronto people. That didn't really go anywhere but we learned a lot about the state of the art and parallelism. But also the other thing that happened is that one of our great visionaries, Irving Wladawsky-Berger he really has just been able to nail what the next big thing is. He was heading computer science at IBM Research Center at the time and he realized that IBM was nowhere in parallel computing; in parallelism. We were very, very late to the game. We started of course, in my mind in Illinois. ILLIAC IV and the great work of Dave Kuck and his students. They put that whole base of ideas in terms of practical ideas came out of there. Of course, there's been a lot of literature on it earlier and good literature. So Irving asked me to start a group on parallelism, compiling for parallels. And I'd already done some work with vectorization.

Kossow: Had the vector feature been released on the XA by then?

Allen: I think it probably had. Yeah.

Kossow: Because that was about the only thing that IBM had was the vector feature.

Allen: Yes, for a long time, and we were late in that too. Cray was way ahead of us. So we did a lot of work on that. I hired a group of- just an absolutely fabulous group. We got the name PTRAN. Parallel Translation. It was really, again, a big kind of module. We built an experimental compiler; followed actually the model that Dave Kuck had of being able to have an ongoing compiling system that one could measure the effectiveness of applying all kinds of parallelization and analysis techniques to programs in order to understand the value and how they interacted with each other. So, we had Cucke's Paraphrase compiler. That was what he had used for his students. Just a marvelous one. I hired a couple of his students. <Laughs> I hired some great people and it just took off. The way we worked on PTRAN was we turned out huge in number of algorithms and new solutions to things on everything from run time

scheduling in parallel environment to debugging in a parallel environment. I got somebody out of Wisconsin for that. And lots of program transformations and analysis techniques. There was a last cycle through on that which was some of the core technologies that we designed and built made there way into the Toronto system through a laboratory out here. <Laughs> So, there was some really good work there.

Kossow: This is across SIMD and symmetric processors?

Allen: No. The last piece of work, which was done by basically a guy named Vivek Sakar, who graduated from Stanford and worked for me, then took a sabbatical at MIT, came back and worked for me in Research. Then is now at Rice University. He kind of has an uncanny ability to pull together nice algorithms and things into good industrial-strength systems. Works a lot of the people together. So, what that system can do, is find, to the extent an analysis system can, find all the maximal parallelism that exists in a program. To the extent it can deduce it. And then from that, compose the most cost-effective configuration of tasks, which takes into account pipelines, the overlaps that happen and just plain old instruction streams. As well as whatever other model you have.

Kossow: Synchronization, decomposition.

Allen: Yeah.

Kossow: We have about five minutes left. Is there just anything else you'd like to talk about. We've gone full circle from parallelism all the way through your career and then back to parallelism again.

Allen: That's right. Yeah. Well, I think there's one last thing. I've always worked on the same problem. And it was a problem that I was inspired to work on by John Backus. It is that compilers should make users more productive and systems better performers. And we've often worked on the performance side but not on the productivity side. Backus had that as far back as Fortran. That was his goal, his two goals actually for Fortran I. That's what we need today. We've given up in lots of ways on the productivity side. In order- because we had to focus on the performance side.

Kossow: Now we have these gigantic systems that have more processing power than we can possibly use.

Allen: And that's why I am excited. One of the main reasons I am excited about this new multi cores is that we are going to have to rethink some of that. Users cannot get the performance out of that and neither can the software systems of today. We're going to be dead on performance. We're just going to trickle off or just flatten out at best. And it's time to rethink both what we're doing with languages and how we express problems and how we then map the myriad of different forms of solutions to the capabilities of the machine.

Kossow: All right. Thank you very much.

Fran Allen: Yeah. Good.

END OF INTERVIEW