Frances Elizabeth ("Fran") Allen,

Recipient of the ACM 2006 A.M. Turing Award

Delivering a Keynote Address

at the

2008 Grace Hopper Celebration

Introduced by Anne Condon, General Chair, of the conference

Ann Condon: It's an honor and a privilege to introduce today's plenary speaker, Frances E. Allen. Like many women of her generation, Fran Allen started out her career with a degree in education, in her case from Albany State Teacher's College back in 1957. She then went on to earn a Masters degree in mathematics at the University of Michigan, and at that point IBM had the good sense to recruit her to join their team.

At IBM, Fran quickly became a pioneer in the field of compilers and also in high-performance computing. She is widely recognized for her fundamental work in the theory of program optimization and also for her early leadership of the Parallel Translations project.

Fran has won many significant awards for her work over the years. Among them she is an IBM fellow, which is an extremely prestigious award at IBM. And in 2007, the Association of Computing Machinery announced that Fran Allen was the recipient of the Turing Award, which is the highest honor for a computer science researcher in our field. And she is the first woman to win this award.

Fran has also distinguished herself in service contributions to our community, and particularly in supporting women. She spent many years as a mentor through IBM's mentor program. In 2000, she herself became the first recipient of the Fran Allen Women in Technology Mentoring Award.

Please join me in giving a warm welcome to Fran Allen, our plenary speaker today.

Fran Allen: I'm so delighted to be here, and I assume the slides will come up. I have so much to tell you. We've got a lot of work to do this morning. I'm going to make quite a few suggestions, and I'm going to tell you about my own career. I need some technology help.

I'm going to talk about really what it means to me and has meant to me to win the Turing Award. A great deal. And also to comment on the fact that the times they are a-changing, for computer science and for the whole impact that computing has had on our field, and on every other field and on people around the world. So as I said, we've got work to do this morning. So let's get with it, if I get the technology worked out here.

So, change is happening. And as everyone knows, last year I won the Turing Award, and I just thought I would point out that I am the first woman since the award was initiated by ACM in

1966, and there have been a total of 54 people who have received it. Now, I am deeply honored by that, but also deeply concerned. And I think we all should be very deeply concerned that more of the women in our field are not receiving, in my opinion, the awards that they deserve.

Anyway, before I get into my rants on some of that, and I really will try not to do a rant, I do want to talk about a number of things. There's a list of what I'm going to talk about. But since Anne has given such a nice introduction about my background before I joined IBM, I'm going to start with how I joined IBM.

IBM had a recruiting brochure in 1957. I didn't see it at the time when I was recruited at the University of Michigan, finishing up my Masters degree in mathematics. But I came across this quite a few years later, and I'm certain, or it's likely, that I was recruited as a result of IBM's expanding and its needing many people at that time, and looking for women. It's always had a great history of hiring women, and other minorities, of course, going way way back.

And this was, I want to point out, well before computer science. Computer science didn't exist when I started in '57, and it didn't come into existence until at least ten years later. And I'm going to come back to that point at some stage.

So the first project after I joined IBM, since I'd been a high school math teacher for a while and joined as a programmer, my manager decided that I should teach the new FORTRAN language, which had come out a few months before I had joined. And they were going to issue an edict that everybody, all the scientists in IBM research were going to have to use this language and give up on using Assembly language. And they had to take a course. So I was the teacher of the course and I had to learn FORTRAN and teach this unhappy class of scientists how to use it.

Well, the net of it was that I was enamored with the whole idea of using a high-level language and adopted— And I didn't realize I'd done it at the time, but over the years I've realized I adopted the goals that John Backus had set in establishing FORTRAN, for user productivity and problem performance. So that became the goals of my entire career in high-performance computing and compilers and computer languages for high-performance computing.

So there's some wonderful stories that go with that. But I want to move on quickly through this and talk about my next project, which was Stretch, again at IBM. You're going to get some IBM history here, some early history that's not very well-known. And in fact, because it isn't very well known (certainly the Stretch is not well-known for some reason) I'll tell a little bit about that. The Computer History Museum two or three weeks ago had an event in which Fred Brooks and I… (Fred was associated with the Stretch project, and I was associated with the Stretch project) and one other person, Harwood Kolsky did a sort of town hall meeting on that particular project.

Its goal, and this was set by the President of the company, was to be 100 times faster than anything that existed. Actually, the first one was built for Los Alamos. And computing at the time was still extremely primitive, and computers were primitive. And "stretch" was absolutely the right word for it.

I point out on the second line a problem that the designers recognized actually in 1955, that memory access time was going to be their biggest challenge. It's still our biggest challenge in computing. We don't know how to get the data to the computation, and it's still… Caches, all kinds of things, since then and it's still a huge, huge problem.

So the total project involved an extraordinary piece of hardware, and some extraordinary compiler to go with it, an I was part of the compiler activity. But in addition to the Stretch part there was another attachment, bigger than Stretch, and a totally different machine for the National Security Agency, and I was involved with that also. Not with the hardware at all, but with building the compiler for it and a language for it.

That still would be, outside of base technology, an amazing machine today. It was a machine hosted by Stretch. It had a streaming data computational model. It was one of a kind. It had eight instructions with unbounded execution times. And of course the basic purpose of the machine was to do code-breaking on data collected up on listening stations around the world. And there was another part to it I don't mention here where the data was stored, which was an integral part of the computational units. And it was the only system that has ever been built that had perfectly-balanced I/O, memory, and computational speeds. So hundreds of thousands of bytes of data could flow through and be analyzed— Looking for patterns, of course that's what code-breaking is largely about. Looking for patterns and accumulating data as a result of the Harvest section.

One of the great designers of that, and Fred and Harwood and I agreed at Computer History Museum event that the greatest designer on the whole system of Stretch and Harvest was Jim Pomerene, who is still alive. He came to IBM after working with John von Neumann on John von Neumann's computer. He was the lead engineer there. And he has not been properly recognized over the years. And in thinking about Jim, I realize that there are many many others that have not been appropriately recognized over the years for their work. Now, in his case a lot of his work was top secret. Some of it was very very early, but he still to this day is working with people on various problems related to these kinds of systems.

My role was again the compiler (I'll show you a picture of it in a minute) but also a language called the Alpha language for the code-breakers to use, and it was a perfect match for this very special-purpose machine in that the code-breakers could write their code very succinctly, their algorithms, and have it map to this very special machine in a very short few lines of code.
I recently looked to see how many lines of code would it take to do a DNA mapping, if one had the system and had the Alpha language, and it was about fifteen to do a total DNA mapping.

Now, we don't have languages like that anymore, or not many of them, that are so high-level and so very special-purpose. As we move forward from where we are and what our next steps have to be (and I'll talk about that in a little bit) we're going to have to be rethinking a lot of how we do computing these days. And we're at a tipping point in the computing field, and a lot of changes are going to be happening. I'll get to that. But going back to looking at something like this, it's clear we can rethink some of the lessons that were learned at that time, and in this particularly bold project.

Here's the compiler we built then, and it's still… The pattern of this compiler being able to take multiple source languages, very different source languages. FORTRAN, a business language COBOL was popular (It was sort of like COBOL) and the Alpha language, which was the language that the agency wanted. And then doing a formal analysis going down to an intermediate form, going through optimization and transformations, and then on an intermediate form that could represent all of the source languages, and then mapping those to the two different target machines… IBM's product compiler today, the family of XL compilers, looks like this in some sense, and it can take all of the product languages that IBM has for its customers to all of the machines that IBM has towards customers, though a system that looks sort of like this, in intermediate forms.

So what were the outcomes of this project? Well, the Stretch didn't reach its 100 times goal. (I'm having to look at my own slides because I thought I'd have my own computer here. They took it away.) But the President of the company, Tom Watson, had announced a 100 times goal on this project, and then he had to apologize to the world. And in fact the FORTRAN compiler, which should've been the simplest part of it, also failed in terms of how for another customer, (this was for weather forecasting), it took 18 hours to do a 24-hour forecast. It was both the machine was not as fast as it was supposed to be, and the compiler was not as fast—it didn't produce as good code as it was supposed to.

But we were inventing everything, and it was a group of very young people. And no experience. And there wasn't any experience to be had anyway. We wouldn't have taken on this project, I think, if we knew what we didn't know.

I might say, as far as the compiler was concerned and the project itself, there were many women involved. Four of us were first-line managers on the compiler, and three of us were women. And this is in late 50s. But computer science didn't exist yet. That's not computer science's fault per se, but I did make a link later on which had to do with how the professionalism of the field evolved over the years as to why we've had some problems with that.

So you see a comment there by Dag Spicer from the Computer History Museum. They have one of the machines, the Stretch. The Harvest disappeared into Fort Meade. I was at Fort Meade for a year and actually was there in the basement during the Cuban Missile Crisis. It was a marvelous and very interesting, very scary, experience. And I thought I would be spending the rest of my career there because I was supposed to write the acceptance test code, and it was to do automatic abstraction of articles, and I didn't. So I wrote up in Alpha an automatic abstracter and I never thought it'd be able to abstract *Time Magazine* articles, but it did the second time, and I got to go home. But I was amazed about that.

Okay, just going quickly, the next project that I was involved with (though /360 came along in here, too) as far as IBM history is concerned was to build…those of us that were disappointed we didn't really succeed on Stretch, that we were to build the fastest machine in the world, again. So we did that. Joined a project to do that, we didn't do it. It was cancelled.

And this time we had moved to California. We were in Silicon Valley, before Silicon Valley was there. And we moved there in order to get away from the Armonk headquarters of IBM. They were distracted by what was going with another big project, the 360 system.

So this, we got a long ways, and did build an experimental compiler, because we didn't know what the machine would look like. So we built the experimental compiler to be language-independent and machine-independent. It was a carry-forward of the very first piece of work we had. I had John Cocke's picture up there. He was an earlier Turing Award winner, and this was his favorite project, and he was one of the drivers of the engineering; just absolutely amazing man. I've had the good luck all through my career of working with some really truly great people, and that's been very fortunate.

But there weren't really any women on that project. It was interesting that seemed to, in that period of the 60s, start to fall off.

When this collapsed in Silicon Valley everybody left to go every which way, I actually stayed on for a while and tried to get the compiler embedded in some work that Gene Amdahl was doing (I worked for Gene Amdahl, if you know him) but that didn't pan out.

So I took a sabbatical at NYU, and my career has always had a circle of research, going to product and, moving often with the research out to product, and then often with doing a sabbatical or some period of time where I look again at what the research issues are. And going to product is a great way to find out what the research issues should be. You spend all your time not solving interesting problems, but trying to meet a deadline, and stack up new interesting areas to explore.

Then I spent a period kind of getting my things back together again, doing a lot of product work, compiler work, adding on, writing some papers. And then IBM, which was very late in getting into parallelism, asked me to put together a compiler group to look at automatic parallelism. And that turned to be one of the most fun recent projects I had. I went to visit the University of Illinois, where great parallel work was going on, and NYU and some other places, and hired a lot of young people and we put together a team that for 10 or 15 years just poured out papers and compiler technologies and built pieces of it and everything. I really really really enjoyed that whole era.

Now, I skipped over what happened when I first came back to research after this kind of round-trip to the West Coast, and I found a glass ceiling. It's taken me quite a long time to understand why the environment had changed. And I am convinced, and everyone I speak to guesses that it's the same reason… What I believe is that computer science emerged as a science, as a profession, with all the requirements on what professional standards and requirements of what one needed to know to get a job in the field.

Most of the computer science departments in the mid-60s emerged out of the engineering schools, which are almost all men. And the engineering schools had courses that the students had to take, and the companies then were hiring people that satisfied criteria that didn't exist early on when computer science didn't exist. Earlier, when anyone who had good marks and was bright and eager, I guess…bright-eyed and bushy-tailed, could get a job. And they didn't have to be

mathematicians or scientists. They could be English majors, or whatever, because the field was growing and no credentials had been established.

In that period, then, credentials were established, and by the early 70s things had really changed for women, at least in my environment, and most other groups that I've talked to about this theory absolutely agree that that was where there was a significant shift.

Okay, moving forward.

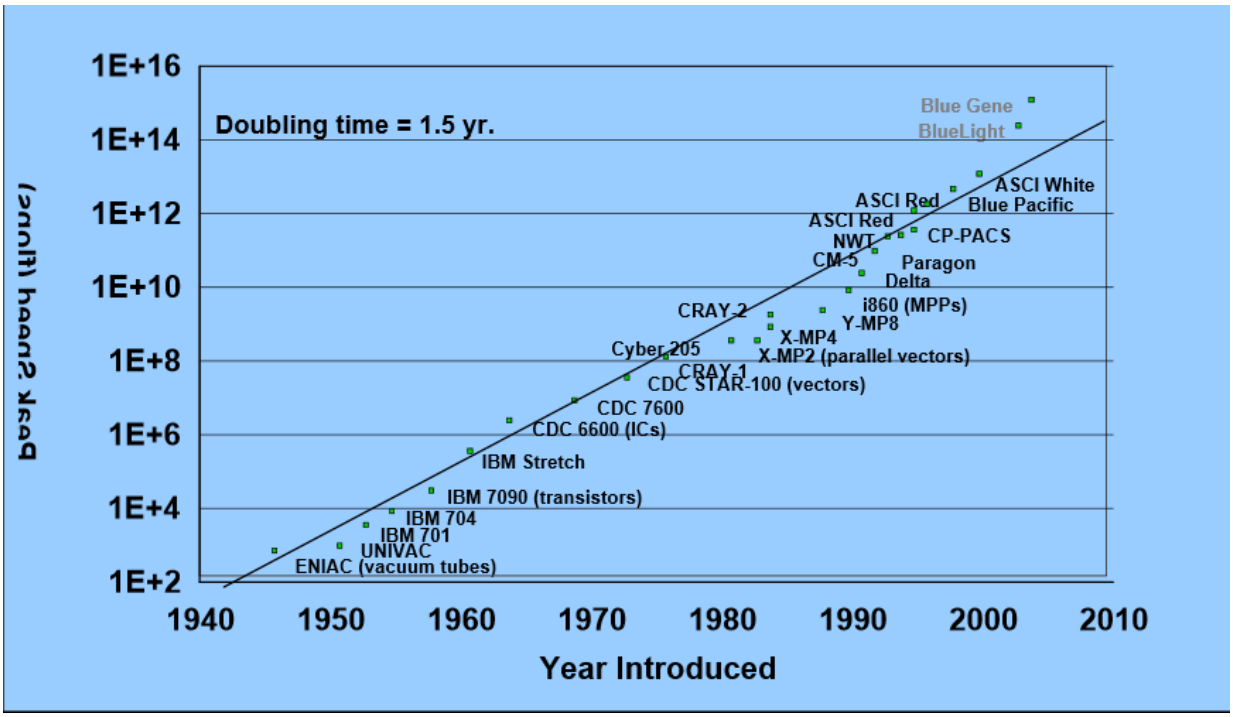
I retired, by the way. I didn't mention that. I did it after 45 years. And then after 50 years of being associated with the field, I received the Turing Award and my life changed.

And I want to tell you what it was like to get a phone call from Ruzena Bajcsy, actually, she was chair of the Turing Award committee—this was in January or February of 2007—to tell me that I was receiving the award, which was stunning. And then I of course got calls from every other person on the committee. But I was warned that I couldn't talk about it for two or three months. ACM and IBM needed to get the PR together. And I was kind of happy that there was a delay in that time, because I had time to think about what this meant to me, and in the first hour of hearing the news… I was home alone and I just kind of paced around and talked to my cat. And realized that one of the things… There were two things that came up. What in the world am I going to talk about? Because Turing Award people and Jim Gray, who was a former winner and he talked with me that week. That following weekend was the weekend that he disappeared, very sadly. But he said, "Well, you're going to be on everybody's A list, so you better get started thinking about what you're going to say."

But the other thing that I kept coming back to was the fact that so many women that I knew, that I'd worked with, and were deserving of so much more recognition than they had ever gotten in their careers… And that's true of men too, of course, like Jim Pomerene. But it really made me feel that part of what I wanted to do was to try and change that, change the way we recognize people and increase the recognition. I now am on a committee in Anita Borg that Katy Dickinson heads, trying to focus on awards for women. I spend quite a lot of time not so much on that committee but thinking about oh, we've got to nominate so-and-so, do something about this, and stimulate those awards. And I think that, hope, just like mentoring this is an extremely important task for all of us. And it's for both men and women. Equity is important here. But I think there's been some equity lacking for some people.

So what I'm going to do now is talk about, and I'll be quick about the next subject, is what did I decide I was going to talk about. I have a little something to say about whither computer science, and the last one is where of course have all the women gone? Well…they're here. And that is great. This is just a really uplifting feeling.

So I'm going to go very quickly through an exciting new problem, which is both an opportunity for a lot of people, and a lot of women have gotten involved with it. And it's new. Well, let me just tell you what it is.



The performance of computing is not going to stay on the curve that is shown up there. Each dot on there is the fastest computer of that year, starting with ENIAC way down in the corner, 1944 I guess, and going up to the Blue Gene. But there's that curve, which is Moore's Law, which has been very steady and doing well in the high-performance field. But what is happening is that the drivers of this, the basic core technologies, are in trouble. This is not my specialty, but some people here would know a lot more about it. But it's in trouble.

The performance futures have fallen off, and fell off about 2002 or 2003. I was talking with Greg Papadopoulos this morning and trying to pin him down on just when it had happened. Any who, close enough. So that is creating a serious gap in the expectations of this area. And also there are problems with heat and energy. That's a separate problem but due to the same root cause of miniaturization.

So now what's the solution? Well, the solution is to move to have explicit parallelism, back to a field I know a bit about, and do it in software. And have what's on the chip, the actual technology…the parts can be much simpler, much smaller, much cooler. And put all the work that they used to do in terms of having capabilities on each of those chips, lots of heavy-duty capabilities in the transistors on these chips, onto the code that's produced by software.

What they've done is given us a tool which, with a lot of parallelism, but it's up to us in software to figure how to use it. And this is a comment on the parallelism, and in fact, we cannot, without a huge amount more work, we're not in a position to either in software or to ask the users to do it. The architectures upon which all these gadgets are based and which are the target of all the software that runs, is changing as rapidly as I have ever, ever seen it, and it's just getting started. This is going to affect everything from the very high-performance computing, maybe even less up there, but it's going to effect the handhelds and everything.

So we're at a fairly fragile crux point in computing for basic assumptions we've been making about computing for a long time, for 60 years. And the software's not in a position to help, to do it, and neither can the users grapple with all of these myriad architectures that are going to emerge at a very steady pace.

So John Hennessy, the President of Stanford and of the Hennessy-Patterson books has said that it's the biggest problem computer science has ever faced, and I absolutely agree with him. I'm not sure that there's any disagreement on that statement. I see it as also the biggest opportunity we're going to get to make some very nice advances in compilers and languages and parallelism. But it's going to mean new languages, new compilers, new systems, and it's not going to happen overnight. Fortunately, the community recognizes all of that, and the community I think itself, from various companies and certainly universities, are all coming together and cooperating on different ideas and aspects of this issue. It's really a time of change for our field, and a time when I think since there are already quite a few women working on aspects of this problem, that it's going to be an opportunity for the women to play major roles. Partly because they're such good community-builders and communicators. I won't go down that…why, but it really is happening in every corner of our field.

As I said, the change is happening. This is a little sidebar. I really have been unhappy about computer science, but I'm really not in the middle of the issues of education in computer science, of the curriculum. But it distresses me that I see less and less interesting work being done in my own field. Piles of papers and conference after conference, and I don't find much new happening. So I've been distressed about the field itself, and I heard Dick Carp, who's one of the great theorists in the field (also a Turing Award winner; he's at Berkeley) say in a talk a couple of weeks ago that computer science "is placing itself at the center of scientific discourse and exchange of ideas, and this is only the beginning." He also says the algorithmic world, which is part of our computer science is changing mathematical, natural, social, and life sciences. So I think that this is absolutely great news from one of our great thinkers. It means change is happening in a very fundamental way.

Switching now to more of where we're going in women's issues. This is first of all my hopes for the future. I'd like to see a new generation of women experience the excitement I feel for the whole of my field. I would really like women to stay in technology, stay in the science side. It's terrific when they become great leaders, but not all women are destined to do that. It's great when you can, but there's a happiness, a joy that I find every day, in the field that I specialized in for the last 50 years. I get excited by new ideas and the new opportunities. And discouraged sometimes, of course. But it's a great feeling to be part— And it'll go on the rest of my life, I'm sure. As long as I have my mind. One always starts to worry.

And I'd like to see women creating the workplace that meets their individual needs. We're at a point where we have so much wonderful technology for communicating and sharing, and it's global. And a global company like IBM has projects where the individuals or people on the projects work on components of the project around the world. If there's an expert in Bangladesh, that person is part of the project and so forth. And well, you know the whole globalization issue, it's adding a great deal to the diversity of our ideas and to the benefit that we derive from it.

And the third one is I'd like to see computer science become a core science. This was inspired by Dick's remarks at the British Computer Society workshop on visions in computing last week.

And I'd also like to see Anita's goal happen. She had a goal for us to have 50% women in the workplace by 2020. Anita had been a student of mine when I had a compiler course at NYU, and I was the only woman and was a professor of hers in graduate school. We kept up a relationship over the years, and I remember one very rainy night I was at work late and I was in a terribly grumpy mood about a lot of things. The phone rings and it's Anita, and she said, "What do you think about $^{50}/_{50}$ in 2020?"

I said, "Oh Anita, we can't possibly do it. It's 2000 something-or-other. It's impossible, it's impossible. Look at what's going on in the graduate schools and everywhere. We can't do that." So I really poured cold water on it.

And then she said, "Oh, well." She said goodbye.

So I really started to think about it, and about half an hour later I called her up and I said, "Yes."

She was great for putting out grand challenges for all of us, even those of us who would get cold feet once in a while.

And I certainly want to have many many more women, and men, but especially women, have the experience of winning an award as I have done.

And I'd like to spend just a little bit on remembering a few great women. There are many many many many, and I would be in real trouble if I tried to make a list. But Anita is certainly one of them.

When I was in England a couple weeks ago, I visited Bletchley Park, which was the code-breaking organization or site for the British during World War II. It's a museum that's in desperate need of funding, and for a while I've been a long-distance member of the museum because of my NSA interest. But I had not ever visited it. Go there, if you possibly can. And look at their web site. This is where Turing worked and lived and did marvelous mathematics and insights.

Also, I discovered, there were thousands of women there ("wrens," they were called) who were in the service and had been drafted to monitor the German communications. Of course there was the Enigma, if you've heard of that machine. It was one of the German machines, and the British had their own.

Then there's the ENIAC computer, which was being built here in the United States. And by the way, the one at Bletchley Park, though it's been secret for a long time, was the first stored-program machine. ENIAC came a year later. So technically it was not first, but it doesn't matter. It was essentially built kinda simultaneously.

Jean Bartik, who was one of the computers on that project, is being recognized by the Computer History Museum next week as a Fellow of the Computer History Museum. She's still alive and feisty. The programmers were called "computers" at that time, or on that project at least.

I'd also like to think, as I did when I received the award, about all the many people… One was particularly a friend, Betty McDonough, who was on the early days of Stretch and built the very first multi-tasking interface machine as part of the memory latency problem, solving it. And the work was just stolen from her. Terrible. But that's the way it happens.

And this is the picture of the Wrens working on the Colossus machine. That's one of the machines that they had at Bletchley Park. For years, three shifts a day, these people would work and never was the information leaked. It was just amazing. Unbelievable.

I want to thank all of you. And I would like as my final word, I guess, is that women's work should not be top secret like the Wrens' work and all of Jim Pomerene's work. And we're ready to celebrate. We have a reason to celebrate. Because we have more and more women, great women, entering our field now. I don't know where they're coming from, because we do have a problem in computer science and computer science is changing, but I really am excited about these times, and it's time for a celebration.

And here is a great picture of a watercolor that Maria Klawe did while she was attending a meeting.

So thank you very much, and goodbye.

Anne Condon: Thanks, Fran. That was a really insightful and fascinating look back to the prehistory of computer science all the way to future, both on the technical and to some degree the cultural front. I particularly enjoy the way you look at the performance challenge as an opportunity rather than a problem, and I think we're all looking at the issues of representation of women in computer science in exactly the same way.

So everybody, let's thank Fran again, and enjoy your coffee break.