The Computer Museum History Center
presents

"Building Computers in 1953: JOHNNIAC"

Willis Ware	Bill Gunning
JOHNNIAC Designer	JOHNNIAC Project Engineer
Paul Armer	Mort Bernstein
RAND Department Head	JOHNNIAC Software Developer

5:30 PM, Tuesday, Sept. 15, 1998
Computer Museum History Center
Building 126
Moffett Field
Mt. View

JOHNNIAC was one of an illustrious group of computers built in the early 1950's, all inspired by the IAS computer designed by John von Neumann at the Institute for Advanced Study in Princeton. Some of these machines were MANIAC (Los Alamos) and ILLIAC (Univ. of Illinois), as well as WEIZAC, AVIDAC, and ORDVAC. JOHNNIAC was built at RAND Corporation in Santa Monica, and named after John von Neumann himself (he "disapproved", but didn't complain too much). This talk will be given in front of JOHNNIAC itself, the actual machine, since it is now part of the Computer Museum History Center at Moffett Field, the world's largest collection of historical computer hardware.

JOHNNIAC ran for the first time in March, 1954. It pioneered the development of time shared operating systems with JOSS (the JOHNNIAC Open Shop System). JOSS could support several dozen users with drum swapping -- there were no disk drives then, so no disk swapping, but instead large rotating drums for magnetic storage. JOHNNIAC also inaugurated the commercial use of magnetic core memory, iron doughnuts on wires, which dominated computer memories for the next decade. Among other tasks, JOHNNIAC was also used to develop digitizing tablets for computer input.

Our speakers were all working on JOHNNIAC over 40 years ago. Willis Ware led the development of JOHNNIAC and received the IEEE Pioneer Award in 1994 for his work on JOHNNIAC. Bill Gunning was the computer engineer who built JOHNNIAC. Paul Armer managed the Numerical Analysis Dept. at Rand where JOHNNIAC was built, and later directed the Computation Center at Stanford. Paul hired Mort Bernstein to work on software for JOHNNIAC at RAND, and Mort is now working on a JOHNNIAC simulator, due to be completed before the year 2000.

PETER NURKSE:

I'm Peter Nurkse, and I represent a group called Bay Area Computer History Perspectives: that's me and Jeanie Treichel, also Sun Microsystems. The two of us have been organizing a history series of talks for almost six years now, and in the last two years we've been joined by The Computer Museum History Center, which as you can see looking around you, is the world's largest collection of historical computer hardware.

Speaking however of our first year of talks, back six years ago, we had a couple of talks on ERMA, which was the first computer built here in the Bay Area; it was built at SRI in Menlo Park as part of a project for Bank of America. Though ERMA back then, 1954, it was still a prototype stage, and after they finished the prototype they tore it down, and then built a totally different computer for the final commercial application, which was installed in 1959. So if you think back then, 45 years ago, 1954, what was going on in the Bay Area, there was some ERMA work, and what was going on down in L.A., well, JOHNNIAC here was up and running, and I think the Standards Western computer was running. That was probably the first computer built in California. Well, the balance of power in computing 45 years ago was rather down towards L.A. It's fortunate, we may say, that Silicon Valley asserted itself and reclaimed preeminence. So that's part of the benefits of history, looking back these relatively enormous periods of time in computing history, like 40-45-50 years, and seeing how things have changed.

The program tonight is on JOHNNIAC, which was built in Santa Monica, and first started running 45 years ago. So I'll turn the mike over to Gordon Bell, who will introduce the speakers. [Applause]

GORDON BELL:

Thank you, Peter. Actually, I was at the Manchester celebration in June, and it was the 50th. We're just 50 years into computing, 50 years ago the first stored program computer ran. What's particularly significant about the talk today is that first we have the JOHNNIAC here, and it's a member of the famed IAS family. It's a true von Neumann machine, whatever that meant -- whether it was named because von Neumann wrote the first report on [the] stored program computer, or, it was a parallel machine, and everybody else was building serial machines. I hope we'll find out the theory about how it ended up that way. [Audience comment: fon Neumann!] What? [It was ffon Neumann!] Oh. Well, thank you [laughter]. Anyway, it was a member of

those machines, the Princeton and IAS machines, which included the ILLIAC, SILLIAC, WEIZAC, MANIAC, and a few others, running in March ‘54.

Now I’d like to talk about the people who did that. I’ll start with Paul Armer, who was at RAND between ‘47 and 68 as, I guess, programmer and eventually head of the computer science. Sort of the project or overall department head at the time. He’s been at Stanford, at the computer center, and has been a member of the Babbage Institute, and doing various other things.

Willis Ware was the designer of the machine, spent ‘46-’51 at IAS, then came to RAND Corporation via North American Aviation, and has been there from ‘52 to ‘92, and is in fact still a member or a consultant at RAND. He’s a member of the National Academy of Engineering, a Fellow of the IEEE, and AAAS, and the ACM. He also wrote one of the first logic design books.

Bill Gunning was the JOHNNIAC Project Engineer, and was at RAND, and transferred to RAND from Douglas, and was there from ‘47 to ‘54. [He] eventually came to the Bay Area, and I ran into him a couple of times at Xerox PARC; he was part of the DEC/Intel/Xerox relationship that created Ethernet. He was one of the key people who made it work. He’s now a consultant at PARC.

Mort Bernstein, who was the JOHNNIAC software developer - have to get to that last. As Ken Olsen was fond of saying, “Good software comes from heaven when you have good hardware.” [Laughter] So we know - it wasn’t until a year or two ago that this explained how you get bad software. [Laughter] So, there may be some wisdom in all of that. Anyway, Mort did mathematics at the University of Pittsburgh, then was at RAND from ‘54 to ‘63; was at IBM for a while, and then went to the Systems Development Corporation. Was part of the programming of the FSQ-7, the SAGE system.

So let’s hear from the team now. [Applause]

I can’t help it, but for the record I need to say that when [the JOHNNIAC] was running the open-shop system JOSS, the machine that replaced it was a PDP-6 that I helped design. [Laughter]. [WILLIS WARE: Major mistake. PAUL ARMER: I’ll let you tell us why later, Willis.]

PAUL ARMER:

I’m supposed to say a little bit about the early history, particularly to explain to some of you who may not know it, the origins of the RAND Corporation, which I want to talk about briefly. But actually, I had essentially very little to do with the JOHNNIAC in the very early days; it was those other folks over there, and not us software types, who were worrying about that machine. The one aspect that I’m concerned about covering is that a number of people, Bill for one, George Brown, John Williams, Willis, made a trip to the east coast when we were considering what did we want to do about meeting the incredible problems we saw, in that we needed a lot more, and a lot more reliable, computing than we had available at the time. One of the things

which happened on that trip -- well, there was a notion that maybe we could get IBM to build a machine for us.
They told us that no, they weren't going to build one. And so we came back and decided we were going to have to build it ourselves. I was off in another group at this time so I don't really know much about that trip and what went on, but maybe some of the others in the group can say more about it.

The RAND Corporation got started at the end of World War II when General Arnold, then Chief of Staff of the Army Air Corps -- and if you remember the history of our armed services, the Air Force is a very late comer to the armed forces, and during World War II it was the Army Air Corps, and then there was the Navy as the second arm. And it was only after World War II that the Air Force became a third member of the armed services. But anyway, towards the end of World War II, General "Hap" Arnold, who was the Chief of Staff of the Army Air Corps, got concerned about the fact that they had all kinds of scientists from the academic world working for them on their research problems, and these people were all dead intent on returning to the academic world as soon as the war was over. So he began worrying about what he was going to do to essentially attract this kind of talent to the problems of the then Army Air Corps.

He apparently talked with a number of executives from industry, and with a number of the academics who were doing this work, and out of this came a couple of thoughts. One, that a military installation was not the right model, that the model that was much more appropriate was an academic model. And also the fact that it would be very important that the scientists themselves have very much of a say about what they were working on.

Out of this came the notion, "All right, we need a nonprofit organization, and we'll hire some of these people, and that will work." The thing that didn't work about that model -- oh, eventually a contract was given to the Douglas Aircraft Corporation to do what was called project RAND. There were two notions about how they came up with the letters R,A,N,D, one of which was Research ANd Development, but the more popular one was Research And No Development. [Laughter] But anyway, this contract was given to Douglas, and that didn't work at all, because the people on Project RAND essentially needed to know what all the aircraft companies and other organizations like that were doing in the way of work. And a Boeing employee was not about to tell an employee of the Douglas Corporation what the hell he was working on. So quickly the plan was changed such that it's got to be a nonprofit organization, with no connections with any commercial company. The Ford Foundation was conned into putting up a big chunk of money, and the RAND Corporation came into being.

I want to say one quick word about that history. I wasn't there at the time, and consequently what I've just said is very much a personal impression of the world at that time, and might be quite wrong.

So the contract was moved from Douglas to the newly formed nonprofit. I might say a bit about some of the things that changed at that time. The notion was that you have to, in this organization, have good salaries and good fringe benefits, and our early fringe benefits were that we had four weeks of vacation, and the first year you got four weeks -- it was not anything that

built up over time. Further, we had from the academic world the TIA-CREF retirement plan, and in the very early days of RAND, when we flew, we flew first class. So, the fringe benefits were nice.

Further in relation to my comment about the individuals having something to say about what they were working on, was that when a senior scientist was hired, in essence he was told, "We've just bought your time for N years; go off and spend it in the best interests of the Air Force." Or, at that time, the Army Air Corps. I think this was just incredibly important in the early history of RAND, and in the fact that it worked out well. I used to often describe the way RAND worked was that each year the Air Force said to RAND, because it eventually became the Air Force, "Here's a bag of money. Go off and spend it in our best interests." And then the RAND management divided this money into N bags, where N was somewhere between 6 and 12, and said, "Go off and spend this in the best interests of the Air Force," and I think added, at least in the way we operated, "in the best interests of the Air Force and science." I think for a number of the things that we did, that was a very key point. And indeed, if there was a proposal that we work on something that maybe its immediate advantage to the Air Force was not particularly obvious, but if it was good for science, we felt we could do it.

Now I want to turn a little bit to the early days of computing at RAND. I was hired by Cecil Hastings, in the summer of 1947, to operate a desk calculator. There were lots of people around RAND in that position. The president of RAND at that time was Frank Collbohm, the number two man was J. D. Goldstein ("Goldie"), and John D. Williams was head of the Math Department. Reporting to him was George Brown, who headed the Computer Science Department. And there was also a Math Department. [Comment: Numerical Analysis.] Numerical Analysis -- did I say Computer Science? Sorry about that. And George Brown has been on my back about that, about four or five times in the past. I'm a slow learner.

Cecil and his group of desk calculator operators reported to George Brown. George Brown left RAND in 1952, and I succeeded him. Willis Ware became Associate Department Head. And then we did, Willis and I, something that around RAND was known as the hat trick, because we traded hats. So, Willis has also been head of the Computer Science Department for many years, although the Department doesn't exist any more.

The biggest problem with desk calculator operation is that it was not only very slow, but that it was incredibly error-prone. If you think about someone sitting at a desk calculator: he gets a result, he copies it onto his worksheet, and then sometime later he's got to punch it back into the keyboard. Every one of those times, something can go wrong, and an error introduced. Of course, you can come up with some examples of being able to do this well, with meticulous care. I mean, for example, at least those of us who were operating desk calculators in those days were well acquainted with mathematical tables that had been produced by, of all people, the Works Progress Administration -- some of you probably don't even know the initials, WPA, but it was essentially a government way of trying to create jobs during the Depression. But they produced a number of mathematical tables that, as far as I know, nobody has ever found an error in. I suppose there are some, but ....

Punched card computing was a lot better, and actually when I arrived at RAND, the Department was already involved in punched card computing, with Douglas equipment. And shortly thereafter, we ordered a lot of punched card equipment. I guess the first calculator we got was a 602, and this was just an incredibly better way of getting computing done. I eventually got out of the desk calculator business, by going to Cecil Hastings to say, "Hey, when we open our own installation I want to work with that stuff."

One of the serious shortcomings of this kind of gear was that essentially the only way you could really do it was in a parallel fashion. Think of payroll. You've got a card for every employee, and first of all for every employee you multiply hours times rate. And then eventually you do the calculations for each employee, calculations of taxes due and other deductions, and finally you print out a check. But, until you start printing checks, you haven't completed the work for any one of the employees. Now, this doesn't bother you worth a damn when you're doing payroll. But suppose that you're trying to maximize a function of five or six variables. What do you do when you have this kind of a situation? Well, you pick out a number of combinations of these five variables, and you start calculating. Until you finish the last step, you've essentially got nothing on any of the cases. And when you do this, say trying to maximize a function like that, when you get through, 90% of the results you've got, you can't possibly be interested in. You know, it was off in parameter space that didn't bring in the results that you wanted at all. So you wasted a lot of time. Not only was it slow, but you wasted a lot of time.

This situation really turned around when IBM came up with the Card Programmed Calculator, in which the punched cards ran through the drive of a tabulator. And in each card was one program step. That meant that you essentially, in this instance of trying to maximize a function, when you started a given case, the machine worked entirely on that case until you were finished with it. So, when you began to learn something about the results from this, you don't calculate all these cases that you threw out the last time. So, literally, that was a marvelous change.

Sort of a brief side story on this: one of the first programs that we did on the CPC involved computing missile trajectories. We were given five cases which had been done, and supposedly checked, on desk calculator equipment. When we ran them, it turned out that not a single one of those cases had been done correctly! They weren't drastically wrong, and people could, I suppose, see that something has gone wrong, and back up a few steps, but none of them was correct.

Now, since I'm trying to tell you a little bit about what was going on in the world way back then, we were all essentially feeling an incredible demand to expand our capability. Not once or twice, but orders of magnitude. Further, we wanted it to be a good deal more reliable. I'd like to quote some figures that Bill Gunning came up with for me. In mid-1953, RAND programmers were using the Standards Western Automatic Computer, at UCLA. That's Standards, as in Bureau of Standards. The machine was called the SWAC. Our programmers did a memory dump, so that they could start over if necessary, every minute. Each minute on SWAC was equivalent to about eight hours on the Card Programmed Calculator. The Card Programmed Calculator had a mean time between errors of about eight hours. Later, SWAC improved to about 10 minutes mean time

between failures. So one of the goals of JOHNNIAC was that it be reliable, and by early 1956 its mean time between failures was greater than 100 hours.

By this time -- by the time that JOHNNIAC began to operate -- the ENIAC effort in the east, and of course the work of von Neumann and all at Princeton, which had such an impact on us that we called the machine the JOHNNIAC, was one of the things that was going on in the world. There was SWAC, as I mentioned earlier, at UCLA. Incidentally, von Neumann was a RAND consultant, and influenced us a great deal. Bill and Willis will talk more about the RAND-Princeton interaction. And with that...

WILLIS WARE:

I'd like to do the "Roots" part of this discussion, and look back at the pre-history of JOHNNIAC, and in particular, the work at Princeton's Institute for Advanced Study, where a great deal of the foundation for the work of the machine behind us was laid. In 1946 -- again, just at the end of World War II -- von Neumann, who had been familiar with the ENIAC during the war, assembled a group at Princeton -- at IAS, excuse me -- to build a machine for science and engineering applications. Burks, von Neumann, and Goldstine had been together at the University of Pennsylvania during the war, and they jointly wrote a document called "The Logical Design of a Digital Computing Instrument." Now I think instrument is the right word. I meant to look it up, but I didn't. And that logical design, spelled out in two volumes, was to be the foundation for the machine that von Neumann wanted designed.

The important thing about that project at IAS was that it was oriented toward science and technology issues. That was natural, given von Neumann's interests. It was to be a parallel machine. It was to be as fast as the electronic art of the day could sustain. And on the other side of the street were the companies like Eckert and Mauchly, who were building character-oriented machines, serial, aimed at a quite different set of applications -- what today we would call transaction calculations, or data processing.

So naturally the IAS effort came to the attention of all the groups in the country that were interested in fast S&E (science and engineering) work. So we had a steady stream of visitors through the project, looking over our shoulders, wanting to mimic for themselves what we had done. And among them was the University of Illinois, Los Alamos, Argonne Laboratory at Chicago, and RAND. So I got to know the RAND people during their visit, and in looking back I suspect the only one I really ever met in any depth would have been Bill Gunning. I don't think I met the others that would have come through. But they looked over our shoulder, all the visiting groups as they came through, would collect the latest information, the latest things that we had done, the latest set of drawings, and march out, and it's a mystery, I guess, that they didn't beat us, because they were that close behind.

Then, finally, when I finished my graduate work at Princeton, I came west, in 1951, first to North American Aviation in Downey, where I learned a little something about how to make airplanes on their production line, but at that time there was an early predecessor of the current IEEE

Computer Society holding meetings in one of the temporary buildings on the UCLA campus. It was the Institute for Numerical Analysis building, where Harry Huskey and others -- Bill -- participated, built the SWAC, and there I renewed my acquaintance with Bill, and met a lot of other people, like Harry Huskey and Ragnar Thorenson, and all the famous names.

By that time the RAND effort had started, a momentous event happened to occur: the Gunnings went skiing, and Bill came back with a broken leg. The RAND management got immediately concerned that here was the man in charge of their effort, and if he were to sustain a more serious event than a broken leg, the project would be in deep trouble. So it was a natural and easy transition for me to move from the east side to the west side of town down there, and to join RAND in the spring of 1952, where I've been ever since. I had grown restless at North American anyway, so it was just as well that that wonderful opportunity came along.

I'd like to comment about the IAS effort, and what it exported to the technical world of the time. Now keep in mind that the commercial electronics art was still struggling, or really hadn't materialized, but it was struggling from the aftermath of World War II. So one just didn't go out and buy anything that one might want by the way of components. So we built a machine out of the war surplus that we could get from the Army. There were regular vendors that would come through who dealt in war surplus, and we'd have a shopping list, and they'd find aluminum, and tubes, and one thing or another for us, and that was what we had to work from.

As I've commented, Burks, von Neumann and Goldstine produced the logical design. We, as an engineering group -- the engineering group varied a little bit, it was five, six, sometimes four, but six, give or take one -- the engineering group produced a prototype design of an all-parallel machine, and it documented it quite well in terms of drawings and progress reports. But we also exported a design philosophy. All of us who were in that engineering group had lived through the war with the pulse technology of radars, IFF equipment, radar beacons, and that was the technical foundation from which we went. So we built all our own test equipment based on that foundation. But for the most part, those devices that I just mentioned beat the electronics unmercifully. The radars typically beat their vacuum tubes with tremendous wallops of energy, and that was not the way to build a computer. The IAS attitude was to treat the electronics as kindly as possible, and electronics meant vacuum tubes at that time.

And so, all our vacuum tubes, we decided on a series of things, to run the vacuum tubes all at derated heater voltages; instead of the nominal 6.3, as they all were in those days, we ran them at 6.0 or maybe 5.9 or thereabouts. We kept the heat dissipation in all the vacuum tubes at half of what the handbooks would specify. We derated all resistors likewise, to half dissipation. We derated -- marked down, so to speak -- the voltage ratings on capacitors. We avoided thermal shock like the plague, so we turned all our vacuum tubes on over a period of several minutes, with a huge variac arrangement. And we designed circuits that had to function properly, even though every component would drift by 10% in the combination of worst directions.

We also exported some important logical principles: all the circuits were to be direct-coupled. No capacitors were to be allowed in signal paths. So circuits therefore would function at their inherent time constants, not as determined by some artificial time scale set by an internal clock.

In the lingo, the machine was to be fully asynchronous. No internal clock, no rigid time-line. So in principle, one could arbitrarily slow down any event in the machine, and it would successfully conclude the proper sequence, although of course somewhat slower. Another design principle was that capacitors would never be used for the temporary storage of information, so when shift registers were designed, they weren't lateral, as the typical design of the day were, they were ratcheting in nature: from here to here, and diagonally back. So every movement of information in the machine was subject to the proviso: it would never be destroyed at its source until it was known to be safely secured at its destination.

The IAS machine was some 1,200 or 1,500 vacuum tubes, all direct-coupled; that must be the biggest direct-coupled DC coupled device that was ever done. There was of course one place where we had to deviate from those rigid principles, where the physics dictated to the contrary, and that was notably in electrostatic memory, where the physics of the charge behavior on the phosphors of the CRT just had to tie you to a time scale.

Paul has given you the beginning of the RAND scene. I've sketched what the pre-history of the JOHNNIAC was and what the technology was that was exported to several places, including RAND, where Bill was in charge of the project. So the next move in this show is for Bill to talk about hardware.

BILL GUNNING:

Thank you, Willis. One of the things I was going to talk about was what we did about vacuum tubes to try to make life more bearable and longer, and he's mentioned both regulating the heater voltage and also turning the tubes -- the heater power -- on slowly, and turning it off slowly. And the reason for that was that a common failure in vacuum tubes was so-called heater/cathode short. For a tube to work, you had to get this cathode, which is a little cylinder, up to a red-hot temperature, and to do that there was a tungsten wire in there called the heater, and it had to be insulated from the cathode. If you slammed the power on to the heater, it will expand in how long it takes for -- well, these are not incandescent lights, but -- in how long it takes for a light to come on: a fraction of a second. But the sleeve, which was the cathode, didn't heat up for many seconds, as you know if you turn on a vacuum tube amplifier. The problem was the differential expansion between the heater wires inside of the sleeve of the cathode. There was abrasion there, and the insulation would wear out, and you have a heater/cathode short.

So, we designed -- I don't know if the transformers are here [in the room] or not -- but we had a special transformer designed that allowed us to put the tubes in groups of no more than 12, and we were able to test the heater cathode leakage. Because as you watch the leakage you get some idea of whether or not you're about to have a failed tube. I don't know where the selector is on the machine any more; it doesn't seem to be where I remember it. But it was possible to go through this kind of analysis while the machine was running. That saved a great deal of time in dealing with the tube problems.

Let me talk a little bit about some of the other things that were copied from, and some deviations from, the IAS machine that Willis talked about. One was that in the IAS machine there was not to be any connectors; when you put two pieces of wire together, they were soldered. And we took the bold step of using lots of gold plated connectors, to be sure. But that made it possible for us to get in and reach a tube that was about to fail, or already had failed. And that helped us a great deal.

A bold step that we took -- we thought it was bold -- was to try to get the indication on the operator's console of what was the state of any of the flip-flops in the machine, and to be able to set them either to 1 or a 0. This meant bringing a wire out from each one of those flip-flops -- there are typically 10 in a bay -- well, in fact there are 20, because there are the two ranks that Willis talked about, in a shift register. You could shift up or shift down diagonally. Yeah, here are all these wires coming out, there's a big bundle, something like 3,000 wires came out of the frame, and went over to the console, which was -- made the wires maybe be 15 feet long.

And the first test on the machine in those days was the prime number test. You wanted to compute prime numbers. It was easy to program, apparently, and check. And it would run up to a certain point and fail. You do it again and it would fail at the same place. Finally we tracked this down to what we called the 30,000-ohm, 30K, effect. Not Y2K. Because that was the resistor that was in series with each of these wires, to try to isolate the capacitance from the flip-flop. And it didn't do a good enough job. What happened then was equivalent to cross-talk in a PC board that you're familiar with in a modern machine, where it's not digital, it's really sort of analog, and that you still have to deal with it. The solution was to stick a little neon tube in place of the 30,000 ohm resistor. A neon tube, up to about 50 volts, is pretty much an open circuit, and then it breaks down and allows the current to flow. And so that's the way the card machinery was coupled to the innards of the machine and the way the console was coupled. Mort will have a story about that later on.

I'll go on to the fact that we had an enormous -- well, like this [gestures wide] motor generator in a room by itself because it was noisy as hell. That was an AC line filter; it had some extra inertia on the shaft, and the power in Santa Monica wasn't all that wonderful, and so that helped a great deal.

The last thing I want to mention is, if you look around on the end here you'll funny little things on a panel. Those are grasshopper fuses, which are sort of a three-terminal fuse that makes a circuit if the fuse blows, and allows an alarm to come on, and they're common in relay telephone switching offices, and we built a lot of those in here.

Let me say just a little bit about, now going to memory, about the Williams tube memory, which was the choice of most of the Institute for Advanced Study copies, and the Selectron memory. Mort brought a copy of the Selectron here, that we can take a look at later. I'll try to go through this fast. The Williams tube memory used a standard cathode ray tube, and stuck a screen on the outside of the tube, at the phosphor end, and ran a wire into the signal amplifier. The way it would store information was to have the deflection set so that it could be placed at any of, if you were greedy, a thousand spots. What it ended up with was usually 256 spots, 16 by 16.

Depending on what was done, the last time the beam was at that particular spot, whether it wrote a second spot or just wrote the single spot, would determine the charge left in that spot. And so when you came back with the beam, you could get either about half a millivolt of signal for a few microseconds, or a much smaller signal. And that means that that was a very, very fragile arrangement. Typically, if you'd come into the room with a 701, which used Williams memory, or the SWAC, or the other machines, and flipped the room lights on, you had a pretty good chance of producing a memory error.

Another problem was that cathode ray tubes were meant to be looked at, not to read the secondary emission ratio off of every spot. If there were little defects in the phosphor, it could change the secondary emission ratio, and make that spot a "bad spot." People at IAS got something like two percent yield of defect-free cathode ray tubes. IBM, by spending, it is alleged, a million dollars to develop a superior processing facility, got that yield up to 60 percent, a great achievement. But it still was -- the room light phenomena was an indication of how tricky this was. Often you would adjust the position of the raster to straddle a defect in this tube, but since they were all in parallel that didn't do you -- that was a loser too. [Comment: Flashes from news cameras were pretty bad, too.] I'll bet! Any kind of electromagnetic flash. And the signal was right in the middle of the broadcast band, too, so if you were close to [radio station] KFI, boy, that was no good.

So what we did, since as Willis said, we were really going for reliability, was to pick up on what was designed for the original IAS machine, the Selectron tube, which has digital addressing, so you don't have any analog stuff to come back and find the spot at which the information is stored. And that was intended for the original IAS machine at a thousand bits. By the time we decided to go with this tube, it was dropped down to 256 bits. We designed it for two banks, so we could get 512 words, if we put in 80 tubes, 80 of these tubes. Here it is. The way it works is, in the center of the tube are eight cathodes. There are metal bars that separate those cathodes, nine metal bars. And there are also metal bars that are going horizontally across the cathode, so by putting appropriate voltages on those bars, you can select one out of the 256 bits.

The information was stored by means of secondary emission; there were little eyelets in here that are insulated -- just electrically floating. But there are two stable states in a secondary emission thing, a high state and a low state. By selecting a particular eyelet and illuminating it with electrons, you can get current flowing through it, and it flowed out into this box, where there is a shielded lead, a coax lead that comes through the glass, and then it went through coax into our sense amplifier. So that meant that we got something on the order of a thousand times as much energy for a readout as a Williams tube. It was a very successful gadget except that it cost $800 each, and RCA decided that they didn't want to make them any more because Jan Rajchman, who -- and George Brown, were co-inventors -- could see core memory coming, and this was doomed. So we went ahead and got enough of them to use them for about a year and a half or two years.

Let me tell you one other thing about it. We went to visit the line on which these things were built, at RCA, somewhere in Pennsylvania. [Inaudible comment] Where? Lancaster! They had this tube sitting in a socket, sort of like this, at the end of the production line, I guess there were

several of them there. And there was this Tesla coil, like a cattle prodder, 10,000 volts. They had it sitting there so that it was -- sparks would flash around on the inside. "What's that for?" Well, they said -- very often clean rooms were not what they are today -- there would be some lint or other foreign matter inside of this enormous vacuum tube, and that would short out an eyelet. But by sparking them with this high voltage, they could convert it into an okay tube, and put it into a box and ship it to us. So, that was something that we learned as a way of extending the life of these tubes, substantially.

Let me just say one other thing. This is a 40-bit machine; each of four bays has 10 bits in it. We built a 10-bit machine -- a 10-bit slice of this machine -- and were able to test the design of the control, and the memory with Selectrons, and that thing we called "Junior". It was later cannibalized and the registers were put into this frame. I think I'd better -- that's the end of the hardware part.

MORT BERNSTEIN:

Well, I want you to observe that software always comes last. [Laughter] It's the last thing anybody thinks of.

The JOHNNIAC was probably, outside of the SAGE system, one of the longest lived computers around. It lasted 13 years before it was decommissioned, and only because it had reached the state where it was not maintainable any more. Spare parts were not available, and it was time to go.

It was an extremely rich software environment, for one very delightful reason: unlike the IBM machines that RAND began to rent, beginning with the 701 sometime late in 1953, the JOHNNIAC was a free good. It had been paid for. The only cost was maintenance. Nobody ever figured out what that was. [Laughter] Nobody ever stated what it was, at least. So, essentially, it was a free good, so unlike trying to run a problem for someone on the 701 or the 704, where you had to worry about 300 bucks an hour for compute time, including check-out -- and I have to remark that the Numerical Analysis Department, when I got to RAND, was unlike all of the other departments at the RAND Corporation, where the management took the big pot of Air Force money and divided it up into little pots, and gave each rice bowl some rice, Numerical Analysis never got a rice bowl. Our rice bowl got filled because all of the other departments took their bag of money and divided it down, and said, "Here's your little piece. We're going to need some computing this year." And if they concluded that there wasn't any computing to be done this year, there wasn't going to be any rice in the bowl.

Well, that meant that rented machines were less desirable in some sense than free goods. But, there's a downside to that. The free good, unfortunately, wasn't compatible with anything else in the world, and the world was beginning to move toward the world of IBM essentially. And the Air Force was beginning to acquire computers made by IBM, of course, and other government agencies were acquiring computers made by IBM. If you were going to do a job for an agency under contract, and give them the results including the software, they weren't going to be able to

do anything with JOHNNIAC software, so there was this wonderful schizophrenia around the place. So projects that were internal to RAND, that had no external connection where you had to deliver the software, could easily be done on the JOHNNIAC. And so the programming environment was a very diverse one, because there were people who were interested in the machine because it was unique. There were people who were willing to devote their own personal time to doing things on the JOHNNIAC because there was no real accounting ever done on the JOHNNIAC. If you wanted to come in at two o'clock in the morning and run the machine, nobody cared. If there was production to be run, of course you couldn't interfere with it, but if the machine was idle, it was available. And so there was an awful lot of bootleg software research done.

And as a result, when the core memory got stuck on the machine -- I have no information or memory about how the machine ran, what programs were used, or even how programs got assembled on the Selectron memory machine. There seems to be no information in anybody's head anywhere about -- except for the first test program maybe being a prime number program. What was the first application run on this machine in 1953? Nobody knows. [Comment: prime numbers.] No, no, no, no [laughter] -- application! Was there a customer who came along and wanted a program done on the JOHNNIAC? I have no idea. And I haven't been able to find anybody who does. I'm sure there's somebody out there who must remember, but I was not around, and I don't. But when I got there, the machine had a very empty head, there was no memory in it at all, and very shortly after I got there, the core memory was installed. They beat it up for a while, and it eventually passed the acceptance test. The acceptance test was an interesting one, and indicated the general reliability of the machine. The acceptance test was: it had to run without a fault for eight hours. That's that the mainframe had to run without a fault for eight hours, which it did. It was a very, solid machine, and compared to the 701, which was in the room next door. The IBM 701 had a mean time between failures of approximately 30 minutes. It may have been less than that, in truth.

And so, if you run a program for the 701 and your estimate was that it was going to run for more than half an hour, every 15 minutes of your estimated compute time you did a checkpoint/restart, so that the operator could remount that tape and restart the program from the point that you had last done a checkpoint. We never did that on the JOHNNIAC. Now, the JOHNNIAC didn't have any tapes to do that with [laughter], but that was irrelevant and immaterial. It did have a reasonably sized drum.

The machine went through a number of -- an evolution, almost a continuous state of evolution until about 1963. The first part of the evolution was, after the memory got installed, they installed a 12,000 word drum. Very shortly thereafter, transistors became THE circuit of choice, and so the machine was cannibalized and the first thing that was changed that I remember, the adders went from analog tube adders to literally, digital discrete transistor adders. And I can remember, there was a great deal of worry about, was the cold air that was passing through the machine cold enough so the germanium transistors would last? Well, they decided that 60 degrees wasn't good enough, so they knocked it down to 55 degrees. From that moment on, you daren't be in the room with any of these glass doors open unless you had a parka on or you'd freeze to death.

[Laughter] That's not a joke. It really was like a meat locker in there with the doors open. The germanium transistors lasted very well under those circumstances.

Slowly but surely, about half of the logic in the machine -- the shift registers, the multiplier control -- all became transistorized. But at that point, attention got diverted in other directions to what to do with the machine. But I have to tell you a story now about -- that Bill said when they put the neons in, as active elements in the circuit. The peripheral equipment on the machine was a high-speed Analex printer which is an earlier version of, a 40-column version of a numeric-only; it's sitting back here in the corner. The 600 line a minute, 56 character, 144-character-wide printer must be somewhere in the [History Center] warehouse, but God knows where it is at the moment. The card reader was an IBM collator, and both feeds were active. The punched card output was a summary punch which had been modified with beefed-up magnets, so that you could punch binary cards. IBM had learned to do that a long time ago for the 701.

I don't remember whether it was a unique feature on this machine, or IBM also did it on the 701, but there was something called echo checking on the punch. And so you would punch a card, and then it would go through a read station after it went through the punch station, and row by row the bits got read back into the machine, and they could be, if you asked it to, and then they could be compared with what you had punched, so that you would verify that what you sent was what you got back. Okay.

Well, very early on the JOHNNIAC was used to create -- to do the payroll for RAND, at least the first stage of the payroll. I don't remember what all was involved, but it was multiplying hours by rate, and other simple-minded things that probably were transferred over from either the CPC or even a 604. The rest of the payroll ended up on the IBM 704, and I have to divert and tell you a funny 704 story at the moment. The payroll was done on the 704, and it was a relatively small machine. I think we had 8K of memory initially. And so the payroll was designed to run in a minimum machine, and so the data fields were minimized to the point where you could squeeze as much data in main memory as possible. The number of dependents, when computing your taxes, somebody figured nobody is going to have more than seven. Right? Well, a guy named Lester Ford, who was a mathematician, had his eighth dependent. And the payroll went through. The payroll people went through upgrading the entry, and that got read into the machine, and this three-bit field that was left for the number of dependents -- the updating took place by adding, and so it overflowed into the next field, which created an illegitimate value. And the payroll came to a screeching halt that week. Well, RAND paid twice a month, and they liked to make ....

[Problem with video camera: about 18 seconds missing]

... the first thing we did that morning, after we got the rest of the machine checked out, was run the punch diagnostic. We must have put 2,000 cards through the punch, without one single error. Didn't make sense. So I sent Harriet back into it, and I said, "Whatever it was, was transitory. Don't worry about it. Punch works fine."

Two weeks later I get a note from Harriet, [which] said "I had a tremendous number of echo check errors last night. It took me all night to get the payroll finished." Well, I have to back up

and tell you that one of the modifications we made to the machine was: we put a bell on the wall, outside the machine room, which rang when the machine halted. That way Harriet could put a stack of cards in the collator, and some blank cards in the punch, start the program, and go out and do whatever other things she was doing in the machine room. And when the bell went off, she knew she had to feed more cards, because it normally would stop when it ran out of cards, either in the input or the output.

Well, that bell rang frequently. Finally, we had to convince Harriet -- by the way, this lady worked graveyard shift, midnight to eight in the morning -- to stay over a little extra, and we would meet with her, and find out what was going on. We asked Harriet to please go through, in gory detail, exactly what had happened when the payroll failed. By the way, she couldn't give me the data, or anybody else, because Harriet Pierson, besides top management, was the only [person] allowed to look at payroll data, so -- it was worse than top secret. It was compartmented like you don't want to know. Anyway, Harriet stayed over, and came in, and went through this exercise, and for the exercise she had created dummy data, just in case we might look at it.

Well, nothing failed, and she couldn't believe it. So we said, "Tell us exactly what you did." So we backed up, and she went through the whole thing, and I said, "Just go back out into the machine room like you normally would, after you loaded the machine." And she did. The first thing she did is, she walked by the door, is turn the lights off. But the drapes were open, and there was lots of light in the room, and so it didn't have any effect as far as we could tell. But, when we sat down and thought about all this, we said, "Gee, maybe you'd better simulate the whole situation." So we closed the drapes, and ran the payroll program. And sure enough, after about 15 or 20 cards, we got an echo check error. The damn machine was afraid of the dark. Open the drapes [laughter], turn the lights on, and the machine ran fine! [Laughter] Makes no sense. Until Dick Stahl, one of the technicians on the machine, remembered that the neons were an active part of the circuit, and apparently by running a little test he determined that without any sunlight coming through in the windows, or fluorescent light from the overheads, which provided just enough ionization to keep them active, they deionized to the point where they would no longer conduct. [Laughter] Now, the question was, how do we fix this? There were something like two hundred and some odd -- how many neons were there? Well over 200. And nobody wanted to get in there and unsolder and resolder 200 -- the machine would have been down for a week at that point.

Then somebody had a brilliant idea. Down here, where the air ducts for the return air from the air conditioner were, they put a bank of fluorescent lights on each side of the machine. If the filaments were on, they were on, the machine never had to run in the dark again. It never did. [Laughter] It never was afraid of the dark again.

That was one of the more delightful aspects of the machine. There were other fun things that happened with the JOHNNIAC. Before I tell you about those, as I said, it had a very, very rich software environment. All in all there were four assemblers written for the machine. The last one was written around 1961-62, and we almost didn't get the write-up published. I called it JACASS. Well, the JOHNNIAC had been abbreviated J'AC for years and years, and what would

you call JOHNNIAC assembler, if you're going to abbreviate it? [Laughter] The people in Publications were very upset at the word JACASS. It survived. I have a write-up to prove it.

The first fully symbolic assembler was written by a guy named Jules Schwartz, who some of you may remember, from other places, SAGE in particular, and JOVIAL. The second one was written about a year later by Cliff Shaw; most of you will recognize the name. He called the thing, the program, or -- he didn't believe it was an assembler. If you look at the write-up it says it's a loader, and it loads his version of symbolic cards. It was referred to as Easy Fox, because by the time he wrote it, we had four card styles for the machine, with definite formats. They were labeled A, B, C and D. Now, Cliff had invented two new ones for his stuff, and so they were called E and F, and in the military alphabet of that day they were Easy and Fox, so it was called Easy Fox. By the way, JOSS was written in Easy Fox.

Last but not least, around 1957 or thereabouts, I wrote a music assembler. Because the JOHNNIAC, as far as I knew, and these guys can't tell me otherwise at the moment, had an instruction called "Hoot". We had three-letter mnemonics, so it was abbreviated HUT, but the real instruction, if you look at the instruction list, is spelled "h-o-o-t", hoot. Its purpose was to flip a toggle which went into an amplifier, and out to a speaker which was buried under the keyboard there. And so you could make music with the machine.

Well, it got generalized to the point where any toggle that got flipped -- in fact, the whole order decode could be passed through that amplifier, so there's a switch in the inside on the left panel which is the hoot control switch, which says you could make the toggle flip every time any instruction went through, or you could pick any one of the 11 or 12 subsets of instructions, or you could pick "hoot" all by itself. I got inspired by a story by a guy named Stu Dreyfus, who some of you may know, who is over at Berkeley, whose brother Hubert is maybe somewhat more infamous or famous, related to the world of AI, but he ran the IBM 701 at 590 Madison for GE for a year or so, when he worked for Herb Grosch. And since they didn't have jobs that took up the entire shift every night, he was left with an idle machine that IBM didn't want back. So he got one of the CE's to show him how to connect a toggle or a wire to one of the overflow bits -- now the IBM 7000 series machines were unique: they didn't have one overflow, they had two, they were called P and Q -- I don't remember whether he told me he used the P bit or the Q bit. Anyway, he was able to get, with a radio in the right place and a wire as an antenna or something, he could get music out. And he learned to play the recorder by programming music for the 701.

That inspired me to write an assembler. I mean, one or two songs wasn't enough to hand-code, so I wrote it in assembler. The first thing I programmed for the assembler was "The Flight of the Bumblebee", which, by the way, it had about a three octave range, which was all I could squeeze out of the machine, because it wasn't all that fast. We were having some visitors at the house one night, and a lady I've known for years says, "And what kind of silly things are you working on now, Mort?" And I said, "I'm teaching the machine to play music." She said, "Aw, come on." I said, "I'll prove it." I went to the phone, I called the night operator, and I said, "Is there anything running on the JOHNNIAC?" He said, "No." I said, "Put the Bumblebee deck on, and put the phone over the speaker on the operator's console." And once it started to play, I handed the phone to her. Well, it could be set up -- and I told them how to set the switches -- it could be set

up so that it continuously repeated. And she sat there and after about a few seconds she said, "That's just somebody playing a clarinet," because it had a very reedy sound. Then she listened some more, and she listened some more, and she listened some more, and she finally turned to me. She said, "It's gotta be a machine -- he hasn't taken a breath yet!" [Laughter]

Sometime later the music assembler became generally available around the Corporation. Sometime later somebody in one of the engineering departments was going to give a briefing and a tour of the RAND installation to some relatively senior Air Force officers. And he decided that the "Bumblebee" and some of the other songs that people had programmed weren't appropriate for a demonstration for the Air Force, and he was going to program the Air Force song. Well, unfortunately, he didn't read the instructions very well, because I'm not much of a musician, and so you had to transform anything you wrote into the key of C, because that's all it knew. And I believe the Air Force song is not in the key of C. To compound the felony, the guy was tone-deaf. [Laughter] He never let anybody know he was doing this. We ended up with about a half a dozen very irritated Air Force officers after he played his version of the Air Force song for them. And not very long thereafter, an edict went out that music was not to be played for visitors unless it had been prescreened by somebody who could understand what was going on. [Laughter] It was a fun place!

Very early after we got the core memory on, linear programming had been programmed for the JOHNNIAC so that they could run small problems and again, it was a matter of money. If you ran something that didn't get -- it was going to cost a lot of money, the Economics Department and people like that -- logistics people -- weren't terribly happy, but if you could run the thing essentially for free on the JOHNNIAC, that was great. So, a number of relatively small but interesting LP [linear programming] problems were run on the machine. At some point in time, one of them failed. Now, there had never been a failure up to that point. And it just came to a screeching halt, after running for about two and a half hours. So, they couldn't find a bug in the program, they couldn't find a bug in the code, and the assumption was, there's a bug in the hardware, and we couldn't find a bug in the hardware. Well, Dick Stahl, whom I've mentioned before, decided he -- in the back of his head I'm sure he had some idea of what might be wrong. But he asked for a copy of the deck. And from time to time, after we'd made some change to the machine, he'd run this thing. It would still fail. Well, about 1961-62, when the decision was made that it was going to become a JOSS machine, Cliff would have been delighted if we could have increased the amount of drum storage by a factor of two, and it was decided that they ought to try to double the density of the drum. Which they ended up doing, and I ended up modifying the diagnostics, and it seemed to run fine until Dick Stahl came to me and said, "There's one thing we don't test in the diagnostics," and that is, the drum was able to start at any address and read to any other address, up to one full track, which was originally 1,024 and now 2.048 words. So that you could start at a high address, read over the gap, and it would run into sequential locations in memory. But we never tested whether or not reading over the gap worked properly. So he said, "Write a diagnostic where I can set the initial address, and we can sort of creep up on the gap, and see what happens to the data."

That was okay. Now remember, it was still 2,048 per band, drum at this moment. And so he started about 10 or 12 words from the end of the block, and worked his way up, word by word,

until he got to within three words of the end of the block. The first two words at the beginning of the block were read in as garbage. Well, he had attributed that to the fact that we had doubled the density. I think probably because it was two words that were clobbered. So we put it back to single density, and then we decided, "Well, we have these new diagnostics -- let's run it!" Guess what. That bug had been in the drum from the very beginning. It failed the same way. They went back, looked at what the problem was, fixed the circuitry. Two and a half years later, that LP problem finally ran to completion. [Laughter]

The machine was a research vehicle in many, many ways. We put lots of goodies on it. There was a 30-inch flatbed plotter, which was very material in solving a problem for me. Back around late '57 or early '58, a guy named George Clement, who was head of the Engineering Division, was asked to give a paper at the Franklin Institute on how you would get an instrument package onto the moon. So he wanted to know, how do you get to the moon? What's the trajectory? How long is it going to take, what kind of boosters do you need, what kind of velocity do you need to ... you know, he wanted all the parameters. We said, "Fine, go look it up somewhere." Well, it didn't exist, apparently, at the time. So we were commissioned -- a lady named Nancy Brooks and I were commissioned to write a program to integrate the three-body problem. Okay, that's simple. So we decided on Runge-Kutta, and we demonstrated clearly that it was stable for roughly the five day transit time. Somebody up in the Engineering Department gave us the initial conditions, namely: we assumed that the vehicle was in a 100-mile orbit, and what you had to select was the insertion angle. That is, where the vehicle was in the orbit, and the insertion velocity. And we were told, somewhere in the neighborhood of 25,000 feet per second, and I don't remember what the angle was. So we decided to make our first run. We missed the moon by 10,000 miles or more! That can't be!

Well, we decided that they had given us some not very good numbers, and slowly but surely increased the velocity and changed the thing until we hit the moon. Insertion velocity was 50,000 feet per second, transit time was two days, and we hit the moon at I don't remember how many hundred thousand miles per hour. [Laughter] Nothing would have survived, and we said "There's something screwy going on here. This can't be right." And the plotter had just come on-line, and what we were getting out of the program was, and we still had, I believe, the 40-column numeric-only printer, was tons of numbers, but we couldn't translate them into anything that meant anything inside of our heads, so I said, "Okay, I'm going to modify the program, where we're going to plot everything out as it goes." So we plotted the earth, and then the moon, and then the vehicle. At about every five integration steps we would plot where the vehicle was versus the moon. Within about five minutes it was quite obvious what was wrong. The vehicle was going east, the moon was going west. We had a retrograde moon! [Laughter] One lousy minus sign strikes again! [Laughter] It wasn't very long thereafter that one of those same kind of minus signs ended up in the destruction of a missile being launched out of Cape Canaveral.

Let me tell you one more story, and then we'll open it up for questions. The RAND tablet was invented by a guy named Tom Ellis, and all of the development software, and proof of concept, etc., was done on the JOHNNIAC. So there was this wonderful new input device -- oh, and we had a five-inch scope, which we could feed back through the mainframe, to show where the pen was. And it was sitting there, and it was an irresistible temptation, and so I sat down and wrote a

little character recognizer, that recognized I think it was 15 characters: 10 digits, the four arithmetic operators, and the equal sign. So if you put digit-operator-digit-equals, as soon as it saw the equals, it produced the answer on the scope. Okay. Well, most of you must remember who J. C. R. Licklider was, he was the first head of the Information Processing Techniques Office at ARPA, and he was coming for a visit to RAND, and we were going to try and convince him to throw lots of money to all the kinds of fun research projects we wanted to do. And one of the things we were going to demonstrate was this wonderful new tablet, which was a new device in the world, and the demonstration was going to be the character recognizor.

I sat down and I showed him -- this is in the middle of August, by the way -- how this thing worked, and he said, "Can I try it?" I said, "Of course, that's the intent of this whole thing." He sits down in the chair -- now, the program, in order to reinitialize it and reset it to its initial state, you touch the pen down at the 0,0 coordinate position. Lick takes the pen, touches it down at 0,0, the machine crashes, the lights go out, and the world came to [an] end. And he looks up at me, and he said, "Did I do that?" [Laughter] Fortunately, he hadn't, and it wasn't my program; it turned out that Con Edison had dropped the whole of Santa Monica. [Laughter]

The last modification that I know that was made to the machine itself was, we had argued for years about how to improve the machine. Do we put an index register into it? Well, that would have caused all kinds of problems, and reprogramming practically everything. And finally I convinced Tom Ellis that there was one more thing you could fix the machine, that would make at least future programs somewhat more efficient, and that was indirect addressing. And he wanted to know, was that really feasible? And I said, "Give me the logic diagrams, and I'll figure it out for you." So I went over it, and I said, "Yeah, I think it's very easy. All you have to do is this." And he finally agreed, and so indirect addressing was added to the machine. We checked it out, and everything that I had, ran. And then Cliff Shaw ran Easy Fox, and tried to load a program in it, brought it to its knees. What we had done, to get the indirect addressing: [a] word was 40 bits long, instructions were 19 bits; the two middle bits weren't used in any instruction. So we programmed -- we fixed the machine so it would look at those two middle bits, and if they were set, then you had one level indirect addressing. The right bit was for the right address, and the left bit was for the left address. Cliff had used those bits as flags in order to save words in the machine. None of his software ran [laughter]. And so, as a result, there is a switch on the console which disables and enables indirect addressing [laughter]. Thank you all. [Applause]

GORDON BELL:

Undoubtedly there are lots of questions, and we'll take a few minutes of questions.

Q. I had the great misfortune of programming two of these machines. One of the questions [inaudible] ILLIAC and TRASK [?]. One of the questions I always had is, why 40 bits? Why was that chosen?

GORDON BELL: That was von Neumann. [Comment: Willis worked with him.]

WILLIS WARE: That's right. Those documents that I referenced, "The Logical Design" etc. said 40 bits.

Q. What was the logic behind that choice? Was it just like, the range of numbers, or ...

WILLIS WARE: I'm afraid you'd have to ask Johnny. [Laughter]

GORDON BELL: I think it's described in the paper; in fact, Bell and Newell published the main one of those. As I recall, it had to do with 10 digits -- you needed 10 digits to do the things ...

MORT BERNSTEIN: Well, you also have to remember this was fixed point, there was no floating point, okay? And so you needed a reasonable range if you were going to do any kind of fixed kind of arithmetic on the machine.

Q. You mentioned the variac, and bringing up the filaments gradually. Did you ever shut the machine off when it was working normally, or did it run around the clock, or did you hazard ...

WILLIS WARE: No, we would power it down, slowly.

BILL GUNNING: But the DC would go off.

WILLIS WARE: Yeah.

MORT BERNSTEIN: But not the filament voltage. [BILL GUNNING: That's right.] The filament voltages stayed on all the time. [Comment: Is that right? That seems strange.] [WILLIS WARE: I guess I stand ...] [Comment: Software guy! - laughter] When you brought the machine back up, all you brought back up was B-plus. [BILL GUNNING: Okay - I was gone by then.] [laughter]

Q. In today's terminology, just for comparison, suppose you're building JOHNNIACs for sale, you want to put an ad in the Computer News, sell the thing, what clock speed, how much RAM did it have, how many ports -- what kind of capacity ...

WILLIS WARE: Zero, zero, zero and zero. [Laughter] It had no clock; it had no ports, in the current meaning of that word; had very nominal memory, 4,000 words by eight bytes per word -- five, rather, excuse me.

MORT BERNSTEIN: And it had a drum with three times as much memory as the main memory, and that was it. And the I/O, to answer about ports, I don't think there is hardly anybody here old enough to remember "copy logic I/O", which was what this machine did, and what the early IBM machines did, before channels came along, and so when the machine was doing I/O, that's all you were doing. You were copying words off of the device that you had connected, or selected, one bit -- not one bit, but one word at a time. [Comment: That was before interrupts, too.] No, there were no interrupts in the machine. No interrupts whatsoever.

Q. Can you describe your simulator that you're working on?

MORT BERNSTEIN: I'm trying to do a faithful simulation of a 40-bit computer, with the instruction set of this machine, that will carry out all the instructions, up to the very end. Part of the problem is, there is an environment one has to build in addition to just the simulator. And that says you have to build the simulation of a punched card environment, including something that looks like an 010, and something that looks like an 026, and [laughter - inaudible]. No, no, no -- we're going to put fluorescents in the base. [Laughter]

Q. Is that why that RAND payroll still runs? [Laughter]

MORT BERNSTEIN: I don't know, I suppose.

WILLIS WARE: The RAND payroll is now outsourced. [Laughter]

Q. With all the upgrading modifications that went into the machine, did you always maintain software backwards compatibility, except for the index bit?

MORT BERNSTEIN: Except for the indirect addressing, yes, everything always ran. We were always able -- well, the only thing that really changed when we modified the machine was how fast it ran. Now, there's a variac at this end of the machine that had the common name of speedbus. There were some programs where you had to readjust the speedbus in order to get them to run properly, like, there were very, very sensitive card I/O. So, if you were going to keep the card reader running at full speed you had to get a certain number of instructions down there after that last 12-hole read before you issued the select, to keep the card reader motor going. Well, if you got the speedbus setup down a little bit too low, and the multiplies were running more than 450 microseconds -- they were running up to 500 microseconds -- you may be unable to make it, and so you go jiggle the machine, and everything ran fine again. [Comment: Turbo switch!] It was continuous, it wasn't a flip-flop kind of ....

Q. You'd mentioned JOSS just briefly. Could you spend a few moments describing what that was, and how it operated, and what the significance of it was?

MORT BERNSTEIN: There's a JOSS console at the far end of the machine. JOSS was one of the very early time-shared user-oriented computation tools. It was not a general purpose time-sharing system; it was aimed at doing small programs. And the history behind it is, one of the ways we believed to keep the JOHNNIAC going, and justify its maintenance costs, was if we could open the shop. I mean, there were fewer and fewer programmers who became JOHNNIAC programmers as they joined the RAND Corporation, and we kept trying to find ways -- in fact, prior to JOSS there were at least four attempts at producing batch-oriented open shop languages. Fred Gruenberger created Quad, and I had created something called Smack, and another called MORTRAN -- you know why [laughter] -- and there was another one in there somewhere. But these were not successful because they were the kind of programming languages that programmers would create. They weren't the kind of thing that an engineer, or an economist, or somebody like that would take to, and it took too much. Even though the advertisement was "it's

easy to learn, it's easy to use," we've been hearing that for 40 years; it's still not true. [Laughter] And the idea was that JOSS would become your computational assistant. Cliff worked very, very hard at making things as natural and easily understandable as he could, and his goal was that the JOSS manual would be one page. Now I -- I don't think I brought it with me -- I have a one-page description of JOSS, which, if you're a bit of a mind reader, yeah, you can figure out what really you were able to do with it. But it was attractively enough done that it didn't suffer from all of the shortcomings of all of the floating point of the day. First of all, he did an integer floating point rather than a fractional floating point. It looked like decimal to everybody from the outside. There was nothing that wasn't decimal, and when you took the square root of 2 and squared it, it came back 2.000000, which is very important for people who don't understand that we've lost a bit somewhere, and now it's 1.999999. And the syntax was oriented to people. People hardly ever put the IF clause in front of a statement they want to condition; they put it at the end. And Cliff looked at that and said, "That's where it has to go. A = B IF ...." So it became the secondary clause in the statement. There were all kinds of little good things like that, that made the thing extremely easy to look at, easy to understand, and relatively straightforwardly easy to use. And it took off! And as Gordon observed, ran out of time, space and availability of the JOHNNIAC, and ended up creating a second version of it with some enhancements on a PDP-6.

The biggest shortcoming of the machine was its shortage of secondary memory, in that you couldn't create a program, even a small five-line program, and save it anywhere. There was no "save" space for users. So people had to retype things. Now, there was some facility for punching out decks and loading them back in, but that apparently created some problems that I -- I wasn't there and don't really understand how it all went, but that was also a lot of the reason for the pressure to move it onto the more modern machine with truly proper storage facilities where people had allocated storage to them and they could recall programs that they had written, and they could write bigger and bigger programs, and build on them, and build them slowly but surely.

How long did it last, Willis? How long did JOSS ... ?

WILLIS WARE: I don't know when we finally turned it off.

MORT BERNSTEIN: The PDP-6? I don't, either, but it served -- it grew like Topsy and somewhere there's a compilation of the number of JOSS-like languages that were propagated into the world by other organizations, and my recollection is, the number was well over 20, so it did have an influence. A guy named Ed Bryan, who was one of the implementors of the PDP-6 version, who looked at what Cliff had done on this machine, and his time-sharing algorithm, as far as Ed was concerned, was better than any time-sharing algorithm that anybody had created up to that time. Interestingly enough, I do have a full source listing of JOSS which I intend, when I get the simulator running, to get running on this thing. Now it's not going to be a big user of JOSS. I don't think I can handle that. But it would be a lovely [inaudible].

Q. Just a quick one. What machine are you going to do the emulation on?

MORT BERNSTEIN: On a PC. What else?

GORDON BELL: OK. There's some refreshments here.

END OF VIDEOTAPE

Transcribed by John Amos, a volunteer for The History Museum History Center
San Antonio, Texas October, 1998