INTERVIEWEE: Irwin Greenwald

INTERVIEWER: Robina Mapstone

DATE: April 3, 1973

RM: The date is April 3, 1973. This is Bobbi Mapstone and I'm talking with Mr. Irwin Greenwald at my home in Venice, California. This is an interview for the Smithsonian Computer History Project.

Okay. Let's start off then with where you went to school, what your contact was with computers and computing at that point, and events that led up to RAND. Then we'll go from there.

IG: I went to school in New York at New York University. I got my bachelor's there and had relatively zero exposure to computers. I had one statistics course where we used some sorters and tabs but I didn't even know what they were being used for.

Then my family moved to California and I came out here and started doing some graduate work at UCLA. About that time I ran out of the GI Bill and went to the Bureau of Occupations at UCLA. They sent me down to RAND where I

was interviewed in what was then called the Numerical Analysis Department and the Econ. Department. That afternoon, when I got home, there was a call to come to work in the Numerical Analysis Department.

Early on I sat over a Monroe calculator and a Marchant calculator doing various and sundry things that I didn't understand. We were also trained in the use of EAM equipment which at that time consisted of 604s, 402s, 407 printers, reproducers, collators, what have you. The Card Programmed Calculator was just coming into being at that point in time. This was in June, 1950, that I started at RAND.

Some of the early work that I did using that equipment was for the AEC doing some hydrogen shock wave problems. Then John Matousek and I got involved with the System Research Laboratory which was some experimentation into how to improve the training of people, which gave rise eventually to the System Training Project and the System Development Corporation.

About late 1951 I guess it was, the AEC asked us to do some work for them and I used my first electronic data processing equipment--the Univac I. We went back to Philadelphia to do some programs in support of the hydro-

gen explosion at Eniwetok and we used the Census Bureau machine which had not as yet been delivered. Are you interested in anecdotes?

RM: Yes, very much so.

IG: Wes Mehlan was working on that project and one of the things he kept asking the physicists for were test cases. They kept saying, "Well, we don't need any, we'll know when the answers are right."

"How will you know when they're right?"

"They'll plot smoothly."

"But if you know what the answers are going to be, why can't you tell us now and then we can check them?"

This argument went on for several weeks and finally they gave us a few things we could stick into the program to check, including stability criteria, energy, and what have you. We got on the machine in Philadelphia and it took us four hours to check our program out. We started running it in and energy went negative. For about two weeks Edward Teller insisted that under these conditions

energy could go negative. He finally left and we discovered the error in the equations and fixed them. So much for Univac experience. (Laughter)

There was another interesting experience in that we worked probably twenty-six hours on and four off. We finally got out to have breakfast and I discovered that Edward Teller speaks very fluidly, he was sitting across the table talking and spitting. (Laughter)

RM: Oh! (Laughter)

IG: We then ran the same kind of problems on the 701, the IBM Defense Calculator.

RM: Was this for the hydrogen bomb?

IG: Yes. Hydrodynamic shock wave implosion problems. I don't know what any of that means. Implosion means you go in until you crush the core and then it explodes. We're up to 1952. We did similar work on the 704.

Let's see, I guess I became responsible at RAND for all programming for the physics division and all utility programming, concentrating more on the latter when we got our 704. Somewhere around 1952-53 the RAND JOHNNIAC went on the air. I did the first large job for it in which, because it was a kind of a test vehicle for the machine,

everything was checked inside the programming. For example, we would make sure that a division could take place before we did it just to make sure the hardware was working.

Oh, backtracking a little bit. One of the things that was occurring in the Southern California area at that time, and this is one of the beautiful things of having worked at RAND, there were a whole bunch of aircraft manufacturers around here as well as other people who didn't talk to one another. Because RAND was nonprofit and had a reasonably good reputation for treating privileged information as such, people started getting together through RAND's offices and talking with one another. The Southern California group: Douglas, Lockheed, North American, Marquardt Aircraft, RAND, and later on SDC, became involved in a compiler writing activity called PACT, Project for the Advancement of Coding Techniques. As a result there was a lot of communication amongst us in this area.

When the 704 was announced, my boss, who at that time was Gene Jacobs, caught me in the hall one day and

said, "Hey you're going to be responsible for our 704."

I said, "Does that mean I can make commitments?"

He said, "Yes, up to the limit of the people you have."

The first thing I did was call Jack Strong at North American and asked him what he was doing to get ready for the 704. He said, "Nothing. What are you doing?"

I said, "Nothing. Let's get together."

The same thing happened with Lee Amaya at Lockheed and Frank Wagner at a different North American division. I think that's about the extent. Marquardt may have been in on this. We all met at Lee Amaya's office at Lockheed, Burbank, and that became SHARE. It was only out of this Southern California area, I think, that something like that could have occurred. Maureen Bernstein got very upset when Herb Grosch said Paul Armer had founder SHARE. (Laughter) It was one of my two contributions to computing.

GREENWALD

RM: SHARE was really the beginning of a whole slough of user groups that have proliferated since.

IG: Yes, I might mention that around 1961 I proposed that SHARE be disbanded. But things like that never do get disbanded.

RM: Why did you do that?

IG: It was becoming too much, in my mind, a competing organization with ACM. One or the other should have been disbanded. SHARE was formed by the founding people primarily to alleviate getting started on a new machine. There was just too much software to write so we were going to share the effort; and that's how it got named. Then some people, including Armer, Grosch, Walt Ramshaw and Jack Strong, all of the boss types, came to the conclusion that they could use it to wield a club over IBM. As Grosch pointed out, what they discovered several years later is that IBM was using it to wield a club over them. The things that it is doing now and has been doing for the past ten or twelve years, I think are more appropriately done within the ACM, though the ACM will never do it. So I can understand why it continues, but I object on principle.

RM: The ACM seems to have a very bad reputation for some reason or other. Maybe it's worth discussing.

IG: I can quote some people's reasons as to their objections to the ACM. Early on it was run primarily by numerical analysts in the universities and the people out in the so called real world objected to the lack of pragmatism to their practical problems. Something like SHARE where a bunch of people with common problems get together and do something about it, seemed more pertinent. The ACM tended to be much more academic. I don't know if it's ever had a president that wasn't an academician. That may have been a reason. We on the West Coast were always upset that we could not get a president or vice-president elected to the ACM. So it was not only academic, it was also Eastern until Forsythe finally became president when he was at Stanford.

At any rate, SHARE will continue.

Okay. Onward and upward. The 709. I guess it was called the 708 at that time. IBM announced the 708 which later became the 709 and SHARE, under the prodding of a number of people decided to get ready early for that one. We had begun to have an appreciation of better ways of programming, debugging, what have you,

so we formed the SHARE 709 Systems Committee which was composed of about fifteen people. I don't know how many of them I can remember; there was myself from RAND, Tom Steel from SDC, Harvey Bratman from Lockheed, Charlie Swift from Convair, Owen Mock who was then with North American, Roy Nutt from United Aircraft, Ira Boldt from Douglas, Santa Monica, Don Shell from GE Evendale was chairing it, and then the IBMers were Elaine Boehm, Maureen Kane, and Vince Degris for awhile; I can't remember the other names.

There was somewhat of a dichotomy. Some of us were concerned with the assembly language, in particular the thing I was interested in was being able to make program modifications at the symbolic level and also do the debugging at the symbolic level. Others, Owen Mock and Charlie Swift in particular, were concerned with operating systems. Partly because IBM never understood the specs, and partly because they were then concerned with a system that they could sell to a lot of people, which meant that they were interested in a minimum of 4-K memory whereas we were interested in 32-K memory

systems, the whole thing never did successfully go, although a lot of the off-shoots have become rather standard over time.

Okay, we're up to around 1958, I guess, with the 709 system. There there were about two years that I didn't do anything. I hurt my back and was in and out of the hospital for two years.

RAND never did get a 709, which is kind of an interesting little anecdote. I wrote a seven or eight page memo to Paul Armer pointing out that there was no reason for RAND to get a 709, meaning that RAND could not justify it economically. So we canceled our order. The 709 came out and proved to be a terrible machine from the point of view of reliability. I remember going to a SHARE meeting and everybody was saying how right I was about the 709. I didn't bother to correct them. (Laughter)

Okay. In March, 1960, I went to work at SDC. I worked there in the so-called Research Directorate for awhile

then switched over to their 7090 programming group where I was responsible again for, we had finally changed the name from utility programming to, system programming and finally wound up in the Satellite Control Department using CDC equipment for the first time in my life. We used 1604s and 160As in support of the Satellite Control Facility all over the world. In 1964 I went back to work at RAND where Joe Smith, Ed Ryan, Chuck Baker and I did the second iteration of the JOSS system. When that was completed I switched over to work on the Videographic system. I don't know if you've every heard of it, it's primarily concerned with supplying graphic consoles at a price competitive with typewriterlike terminals. The view at RAND is that these tools should be available to a researcher the same way a telephone is. The whole attempt here was to get the cost of display consoles down.

The way we did it was to centralize a lot of the hardware so that the terminals themselves were rather cheap. We used a commercial CRT that cost about five hundred bucks, we built our own typewriters for about one hundred and forty bucks, and the RAND

tablet for those stations that had it. I don't know what it cost. An actual station like that would run somewhere between one thousand and two thousand dollars; the centralized hardware, which could be distributed over maybe thirty such stations, ran to about one hundred thousand dollars. So that the total cost we used was getting down into a cheap level. I was concerned with an 1800 computer which we used as a communications coupler between all these gadgets and host machines. I guess now they're running 360s, possibly the PDP-10, but I'm not sure what RAND is using now as its host machines.

RM: I take it that by the time you went back to RAND they were less defense oriented. Is that true?

IG: I would guess that at that time probably 90 percent of its funds were still coming from the Department of Defense in one way or another. While I was there on the second tour of duty, which was 1964-69, we changed presidents; Frank Collbohm retired and Harry Rowan came in. Rowan, as far as I could judge, was interested: number one, in finding clients other than defense; and number two, doing more work in the soft sciences rather than the hard sciences, and these two kind of

go together. It was during his reign--terrible word to use--that, for example, the New York RAND Institute was born. I don't know what RAND's contract mix is now, but I think that probably 80 percent of what they have is defense. At any rate, there was this change in desire, at least on the part of the management of the corporation. All of the work that I did--all of it--was supported by the Department of Defense, or the AEC. Oh, I got a little bit involved with something about the ILLIAC.

RM: What was RAND's ILLIAC involvement?

IG: RAND was one of the ARPA contractors. One of the things we were doing was various kinds of modeling, atmosphere, weather, what have you, and it was proposed that some of this work be done on the ILLIAC. I got involved more or less as a consultant to the people who were doing this kind of work to look at ILLIAC in various ways. We discovered that some of the problems would be better off on a CDC 7600; others the ILLIAC seemed to be quite geared for.

RM: Which ILLIAC are we talking about now?

IG: The one that eventually went to Ames. ILLIAC IV. I'm sorry. I call it the ILLIAC. (Laughter)

In 1969 I left RAND and went to what was then SDS. It has since become part of the Xerox Corporation. I can't think of anything I've done since I've been there that's worth talking about.

RM: Okay. We can do some backtracking. When you were at UCLA were you aware of any computer activities there?

IG: I was not aware. There was stuff going on with the SWAC.

RM: But you didn't know much about it?

IG: No. I didn't know anything about computers. Paul Armer was the primary one to interview me in the Numerical Analysis department at RAND and he took me to this machine room and talked about this, that, and the other thing. When he got all through he said, "Now, do you have any questions?" I said, "I didn't understand enough to have any questions." That may have been one of the things that got me the job because evidently they had had a bunch of con men come in for interviews. I didn't know a thing about computers.

RM: When you went to RAND in 1950, what kind of work

was going on in the Numerical Analysis group?

IG: Primarily then, and for lots of years subsequent to that, the Numerical Analysis department was, in my view, a service function to the rest of the company to do computational work as needed. Most of it was preparing jobs for the various machines as we grew up. I would say that calling it Numerical Analysis department was a terrible misnomer because almost all of the scientists did their own numerical analysis. For example, they'd come down to us with a bunch of difference equations which we then programmed and put on the computers. Every once in awhile we would get equations to solve. It's kind of funny because I guess our mental set was that if they ever got to us they must be something that would have to be done on a computer. We'd take integral equations and do Simpson's method or what have you, until one day someone brought down I think it was sixteen integral equations. I started to program them and Stan Rothman started looking at them, and he discovered about three or four that had analytical solutions right off the bat. The two of us went to work and we found fifteen of the sixteen had analytical solutions.

GREENWALD

(Laughter) It just kind of shows the mental state you get into.

But what we did particularly back in the EAM days at RAND was we split those people who, if you will, wrote the procedures and those people who wired the boards and pushed the cards through. This appeared to be very effective. However, there was one instance when Barbara Batchelder went in to the tab supervisor and said, "Can you wire a board to do such and such?"
He said, "yeah."
She went in the next day and said, "Can you wire a board to do such and such?"
He said, "Well, that's a little rougher. Yeah. We can do that."
This happened four or five times and then she submitted a procedure that had all of this in one step. (Laughter)

We always separated the machine function from the procedures function or the programming function or what have you, which was very difficult for some people that we hired to get accustomed to. Stan Rothman, for example, was used to doing the whole job himself. I guess all the years he was there he never did get

to the point where he could accept our style. I know three and four different people who tried to work with him and gave up because he insisted on doing it the way he used to, which was good, but not our style.

RM: What was the type of background of the people who were wiring the boards and pushing the cards?

IG: Most of them had some amount of college; some of them had degrees; a lot of them graduated from that function into programming. For example, Bill Orchard--Hayes, who did an awful lot of the early work on linear programming, started out in the tab room. All of us had to serve an apprenticeship in the tab room so we knew what the heck we could wire those machines to do, and also to get an appreciation of the problems the operators had. One of the things that has always stood me in good stead was that Don Madden and Bob Bozak who early on I had as my seers, both insisted that those are the people who are going to run those jobs for you and you better be on good relations with them. Among the ways to help is to write so they can read it, write so it's not ambiguous, and don't ask them to do things that are impossible. I was always very sensitive to that kind of thing; maybe it's because I'm a person who likes to be liked.

This stood me in very good stead because in the days when we were doing the STP work I had an EAM procedure that was about 113 steps that had to be run three times a week. One of the girls in the machine room remarked at one point in time, "If Irwin wrote it, that's the only way you can do it so it's okay." This having to get along with other people is crucial. For instance, when you write a keypunch manuscript somebody's got to be able to read it to punch it.

RM: That's true. So really at this point you haven't got the computer room operator as we know it today. They were probably at a better educated level than today's operator. Do you think that's a true statement?

IG: They were a real mix. There was anything from key-punch opeators that graduated to EAM, to guys with bachelor's degrees who eventually graduated to programming. So there was a whole mix of people there.

When we got our own machine--I guess the 701 went on the air before the JOHNNIAC did--as was kind of customary in those days, everybody ran their own stuff. Because this was the most idiotic way of doing anything, there were a couple of things we did. First we installed an operator, more as an assistant to people running stuff and as a guy to get on the phone and call the next one and what have you. Then, over time, we began to apply

pressures that led people to having him run their jobs for them. The first thing we did was to give priority to things that the operator would run, and so we had two hours in the morning for operator-run jobs and then an hour for personal time, et cetera.

One of the things people used to do was an awful lot of tracing and dumping which we thought was stupid. So we put a Polaroid camera in the machine room and whenever the machine halted the camera would take a picture of the console. Anybody who would accept that rather than insisting on a memory dump would get priority. So we were trying out various things. I've always felt strongly that the bugs were in that program, especially in those days when programs were rather simple. In fact my philosophy was, "Go back and look at it at your desk. You'll find it a lot faster." Out of that kind of thing at our installation, and at lots of other installations, gradually grew the idea of operators and eventually operatorless systems.

One of the things that I felt strongly about and could not sell--and it's an interesting kind of thing, but it's an unsalable idea--is that you have to be willing to pay your operator as much as you're paying your good programmers. Although management in some places

would accept that, they soon discovered that an operator that's worth that kind of money doesn't want to be an operator very long. So we had to come up with something to get around the problem of operators, and I guess operating systems was the natural step.

RM: Okay. Another question I should get back to: was the Numerical Analysis group at RAND in contact with the Institute of Numerical Analysis at UCLA? Did you work closely?

IG: No, other than for individual people knowing individual people, there was no formal tie there. A lot of the people that came to work for us were from the Institute. We had very limited use of the SWAC.

RM: Did you personally ever use the SWAC?

IG: No. Rothman put a job on the SWAC. I don't know how he ever got around it because as I recollect there were only 256 words of storage. (Laughter) I don't know how he got that job working but he did. He was a brilliant guy.

It's kind of funny, you know, I stated earlier that through the auspices of **RAND** lots of people in this area got together. However that was only through a very few people like Paul Armer, Don Madden, Wes Mehlan,

Bob Bozak, myself, and a few others. The rest of RAND, the computer community, was then, and has been for a lot of years since, very very parochial. There was this strong dichotomy between people who were interested in computing and people who just had a job to do. It's always bothered me because I don't think a place like RAND should have been that way; I think other places might have been. RAND at that time did a great deal to encourage people. For instance, going to meetings; they'd give you money, you didn't have to write papers to go.

RM: That's interesting because RAND was like the great umbrella over the West Coast computer industry. It must have had a tremendous impact in many ways. But what you're saying is that it was really a very small group of people who were spreading the gospel, so to speak.

IG: Yes, and as I say, RAND was not much spreading the gospel as a place where all these people whose companies were competing could get together. We learned as much as we contributed. I guess I was as involved with the other installations as anybody at RAND and

this was people learning from one another. For example, one of the things that we--at least some of us--never suffered from was the "not invented here" syndrome. Consequently, when we got our 701 we picked up programs from all over. We picked up everything that they had at Los Alamos--we did our first work up there; we picked up the Douglas operating system; we picked up IBM's stuff; we even got something from England, I vaguely remember picking up the ABC system. Then there were other people at RAND who were quite parochial; you know, if we didn't do it, it's no good. To most of us who were service oriented these were just tools to learn how to use, and if they helped do my job that's fine. So we received as much as we gave out, if not more.

RM: If you had to say what was really RAND's significance to the computing industry as a whole during the period, let's say late 1940s through the 1950s...

IG: I would say, primarily, that because of our nonprofit status and because RAND had established a reputation for being very sensitive to privileged information--that we could serve as the focus for people getting together. Part of it evolves from something very strange;

namely a personal foible of Paul Armer.

Paul Armer, for whatever reason, was very much concerned with knowing everything that was going on, so part of our responsibility whenever we went off on any kind of a trip was to write a trip report of what was going on there. He became a fountain of knowledge which he freely disseminated, other than sensitive stuff, to anybody around. It was that kind of a thing; an interest in computing, a free and easy way with nonsensitive information, and very protective with sensitive information.

RM: In other words if somebody with one of the aircraft companies, for example, was interested in knowing something about the merits of somebody's system, they might have gone and chatted with Paul and got some poop on that machine.

IG: Yes. For example, when Walt Bauer at TRW was setting up the computing center there (before he went to Informatics) he came over to RAND because we had nothing to keep from him, whereas one of the aircraft companies might have, though in reality they didn't. Everybody around here talked pretty freely. I think it gave them ego benefits

to help somebody else.

Having started as a nonprofit corporation, some of the problems associated with profit-making motives were never even apparent to me. In fact, some of these only began to come out, oh, maybe three or four years after SHARE was born. I guess that's about the time when people were less concerned with "How do I get this installation running?" and more concerned with the applications they were putting on it. So lots of things began to get very sensitive. Gradually this led to proprietary programs.

RM: That's interesting. That would have been about the late fifties?

IG: Yes, I would say starting around 1956 or 1957 or 1958 that people were beginning to develop programs which, in their view, could give them competitive advantages and they didn't want to be giving these out or talk about them.

RM: Can you enlarge on the process that you went through at RAND for training programming people?

IG: I'll try to recollect. Back in the days when we used the EAM equipment we had two kinds of training. As I mentioned earlier everybody served an apprenticeship in the machine room, and, as a result, learned how other

people were running procedures. We always paired a junior person with a senior person and gradually the junior person would take on more and more responsibilities. When we got to the EDP equipment it was half a process of osmosis and again we paired junior and senior people. When we got somebody off the street earlier on, I remember our famous technique was to go in and talk about computers.

I did a lot of the early so-called training which was a matter of explaining what a machine was like, gradually building them up to the point of going from binary to octal and finally to assembly line, which was all we used in those days. One of the exercises we gave them was a one-card binary loader which they had to decode. That was our standard practice. Subsequent to that in the later years when I was at RAND, we hired very few people who weren't experienced, so it's difficult to say how we went about the training process. We did bring people out of the machine room who kind of went through this process of osmosis. We never had a formal training program at RAND while I was there.

At one point in time we had a big staff of people who were assigned to any job. In time, I guess in the mid-fifties, we began to break these down into teams, primarily to work on different contracts. This was our way of doing our budgeting. The various team leaders

worked in different ways. I always worked with two people assigned to every job--senior and junior or quite often senior and senior-- in which work was interchanged and desk checked before it ever got to the computer. Other people worked strictly with one person on a task. There's been a running debate for about twenty years as to which way gets you more work done. One of the things that people who put one man on a task ignore is the fact that if the guy quits, dies, or gets assigned to another project, there is no back-up.

When SDC got involved in the SAGE system suddenly they realized that they would have to bring on-board about five hundred people and, in some fashion, train them. So they went to formal training programs. They had classes running, I guess, on the order of three or four months.

A more interesting kind of question, rather than the training problem, is the selection problem.

RM: That's my next question.

GREENWALD

IG: Very early on we submitted all of our people at RAND to a battery of various tests, after having had them ranked by their managers. We discovered some correlations between some of the tests and the rankings. I just recently read where Bob Reinstedt, who was later on one of the key figures in devising tests and what have you, had kind of said, "They're all a bunch of malarky." (Laughter) I think the selection process is apt to be more critical than the training process. Something the selection process has to take into account, and it doesn't very often, is who will this person be working with, and do they fit. To me that's much more crucial in the long-run. I've discovered in my own work where I tend to be senior on a project whenever I'm involved in it, that it's no real problem training people to do new things that they haven't seen before if they've got any brains at all and if you've got the patience to do it and do it and do it. Also the attitude that gets them to the point where they don't feel foolish if they have to ask a question. But that's very limited, that's the personal contact. I don't know what you do in the classroom; I had no exposure to it.

RM: In the selection process in the earlier days, were there any attributes that you specifically recall?

IG: I'm trying to remember. There were two tests that had a correlation. In one of them I think there was a negative correlation with sociability. (Laughter) The test was measuring whether you liked to work with other people or you liked to work by yourself in a corner. This was measured as a sociability trait; the more sociable you were the less correlation there was. I don't recall what the other one was.

RM: One that seems to stick in my mind is music training.

IG: I don't recall that one. An interesting thing about the SDC experience and the SAGE system is they discovered that an interest in solving puzzles was apt to be a good criterion. They had English majors and Lit. majors and taxicab drivers and everybody and their brother that hired in. I was doing some of this early training, as I mentioned, and something that I have never understood in myself, is that after somebody had done the selection, whether it was happenstance or what, every evaluation that I did based on maybe two or three weeks with people stood up over time. Now, whether that's an art, luck or what have you, I'm not sure.

RM: Serendipity, maybe!

IG: I really haven't been particularly involved in either the selection or the training process, so I can't comment any further.

RM: Okay.

END OF SIDE ONE (Tape 1, Side 2)

RM: Let's talk about the first job on JOHNNIAC. The reason I'm specifically interested in first jobs is that it seems--and I picked this up from other interviews--that on very few of the computers that were built can anybody really recall some of the early work that was done. If you can talk about the first job in some detail it would be very helpful.

IG: Okay. Let me start off with the tools. Wes Mehlan was primarily responsible for the software aspects of JOHNNIAC at that time. He wrote an octal loader called JBL-03. I'll never forget that. It loaded programs written in octal and that was the tool that we had to build our programs--that I had, because nobody else worked on it at that time. The only other tools we had were the JOHNNIAC console which, in those days, was rather sophisticated in terms of being able to look at things in the machine, and a dump program that came spewing out on an Analex printer and produced toilet paper that we used to roll down the halls. (Laughter) At any rate, the problem was to be several things. It was a bread-

and-butter job, namely something to run on the JOHNNIAC when there was nothing else to do. It was supposed to verify, or at least catch, the hardware when it was making errors, and some of the errors that the JOHNNIAC was supposed to catch itself were impossible divisions--denominator less than the numerator--overflows of various sorts, the arithmetic-type faults, and that's pretty much it. The problem was a repetition of what we had done on the Univac, a hydrodynamic shock wave problem. All arithmetic was checked; in other words, every time there was an addition done we verified prior to doing it that an overflow could not result and then checked for overflow afterwards. All division was pre-checked that, in fact you could do the division, that the denominator was greater than the numerator. It was all done in fixed point, you might make a note. It was, by today's standards, a rather small program. JOHNNIAC at that time had 4096 words of core storage, and that's all we used. We didn't use a drum or anything else.

RM: This was after the Selectron Tube was removed.

IG: Yes. The core memory on the JOHNNIAC is a story in itself if you haven't heard it before. International Telemeter or Telemeter Magnetics, I get very confused, contracted to build this core memory for us with a penalty clause on the contract, and they paid through the nose.

RM: Yes.

IG: You've heard that story.

RM: Yes.

IG: I can't quire recall the debugging of that problem. I recall one interesting incident: JOHNNIAC was a ones complement machine, and double precision negative numbers, very small negative numbers, were a string of seventy-nine ones. One of the problems we had with a set of equations we were dealing with was stability. Certain results should have gone to zero but came out 2^{-39} because of the one's complement and truncation features with JOHNNIAC, and the problem blew up in our face. So we had to put in a test for: "is the number almost zero". (Laughter) I had many arguments with Wes Mehlan.

That first job was kind of interesting. As I say it was fixed point and had very detailed flow diagramming, actually expressing the scale in every intermediate quantity and what have you. I particularly remember that, because the job ran for years and was being massaged--changes made in the equations and what have you. Jack Little picked up on it. He's one of the people that I mentioned earlier as having worked at RAND and then became successful. He's now some wheel at PRC. That's about the extent of what I can recall

about that one. When was it done? That would have been about late 1952 or 1953 somewhere in that time period.

RM: So actually, after all that work that the builders of JOHNNIAC did on the Selectron tube, they very quickly brought in CORE?

IG: Yes, the Selectron turned out to be a very reliable tube that was commercially unfeasible. I guess we did some kind of testing on the tubes and I think we rejected 39 of 40, which meant that the price got pretty horrible. So RCA got out of the business. RCA delivered at least 40 and you can see them on various mantlepieces around. (Laughter)

RM: That's right. Okay. Did you do any useful work with the Selectron?

IG: We did some work using the Selectron. I'm trying to recall what. It was mainly testing the parts on it.

RM: But this hydrodynamic shock wave was the first real problem.

IG: That is what I would consider to be the first real problem.

RM: After that did you use JOHNNIAC?

IG: I didn't use JOHNNIAC much at all. I'm trying to recall

if there's anything else I did on it. Nothing that I can really think of. Among the people primarily using JOHNNIAC at RAND, Mort Bernstein was the guy responsible for JOHNNIAC. At that time I was responsible for systems programming, utility programming, or whatever it was for the 704--I'm trying to remember when the 704 can in--and the 701. Mort was responsible primarily for JOHNNIAC and I took primary responsibility for 701 and 704. He built something interesting; he built a little open shop system which they called SMAC for small compiler. It was a little compiler that the RAND staff could program the JOHNNIAC for. It was quite successful. That's an interesting story, too. Mort was doing that at night on his own time and I said, "Why?" He said, "Because nobody gave me the go-ahead to do it here." And I said, "Well you've got the go-ahead now." And he did it. I assume he told you the story about the JOHNNIAC being afraid of the dark.

RM: Yes. That's a lovely story.

IG: I guess we had a plotter hooked on. One of the things we did on the 701 and on JOHNNIAC because computers were so unreliable in those days and because we didn't trust tapes very much, we would punch out, under

operating control what was going on because these problems ran for ages. It turned out that was very profitable because we could then take the punched decks and run them. We didn't even have to make any changes on them. I'm not quite sure, we may have made a run on them and then put them through the plotter because that's all the physicists were interested in anyhow, the plots.

RM: Did you do any work with George Danzig or Bill Orchard-Hayes.

IG: Other than what I mentioned earlier about Bill wiring up some boards, and also because I was the primary interface between the 701 and 704, and programmers versus C.E.'s, I had a lot of interface with all the programmers. I knew George Danzig but I never did much in the way of work for him.

Bill Orchard-Hayes discovered something rather fascinating on the 704 tape units. He was doing some parity checking; the 704 had lateral and longitudinal parity check. Bill discovered that the parity check light was on after he had already taken corrective action,

which led to the discovery that on the 704 the longitudinal parity check was coming in 500 microseconds late, and therefore, when you got a longitudinal parity check it was for the wrong record. This gave rise to a big hullabaloo through SHARE. It also explained why SDC lost something like eighty hours worth of production on some work they had been doing.

It was kind of fascinating because Bill Vitek, who is now a Vice-president at Xerox, was then one of the CEs on the machine. I went down to him and said, "Hey, there's something real wrong with that longitudinal parity check."

He said, "No, there's nothing wrong with it, it comes in 500 microseconds late."

I said, "Where does it say that in any of your manuals?"

He said, "I don't know, but I can look at the circuits and see that it's coming in 500 microseconds late."
(Laughter)

RM: Talking about latitude and longitudinal parity check, I know what a parity check is but I'm not sure I understand the terms latitude and longitude.

IG: Okay. If you think of the bytes as being frames, there's a parity check across so the tape is eight tracks wide. One of those tracks will be used for a parity check. That's called the lateral or horizontal. Then as you go down the record, at the end of the record there will be a vertical or longitudinal check, so it's a double check which should, assuming a single failure, enable you to find out exactly where the failure is. It gives you the capability for single error correction and multiple error detection.

RM: Except it didn't work?

IG: In the days of the 704 we had what we called copy logic where each word that was brought into the computer had to be directly copied in and there was a timing loop that you went through. At the end of the timing loop supposedly, the logitudinal parity would come in and you'd do a simple Boolean function on it to determine whether it was okay. It turns out that you were doing this simple Boolean function before the logitudinal-parity was in. So if the error occurred on record N, you would recognize it on record N + 1. I guess the hardware did that check. So you would

check the tape, "Any errors?" It would say, "No." 500 microseconds later it would say, "Ah, I had an error." (Laughter) It was rather expensive putting a 500 microsecond delay in there, even on those slow computers.

RM: Okay. You talked about utility programming which later becomes part of an operating system. How about going back to the beginning of this period where you start to get into a need for utility programming? How did you arrive at this attitude? How did you get to or how did you write the first program? What were the things that were written into these programs? What was a utility program?

IG: I guess you get a lot of argument on that. Wes Mehlan had the philosophy that any program that anybody writes should be written as though it was going to go into a library. Now, even in those early days, everybody recognized that sub-routines like square root, sine, cosine, what have you, were to be in a library. Wes insisted that every program ought to be written that way, and people gradually without realizing it are coming to that philosophy. Every program needs to be thoroughly documented. As Frank Wagner says,

"If it ain't documented, it doesn't exist." (Laughter)

To my mind, there is this whole library of things that people use in common that I would classify as utilities, though other people might not.

Then there are the tools for getting your work done. There are programs to load other programs, there are the assemblers, compilers, what have you. I guess people stopped thinking about them being utility programs when we crossed that thin line between assemblers and compilers. There are dump routines, there are trace routines, all the debugging aids. Then there is a potpourri of miscellaneous kinds of things; for example, after doing a logistics experiment at RAND they wound up with something like 400 reels of tape with maybe a quarter inch of the tape utilized on each reel. They didn't want to throw them out; they wanted to archive the data and yet didn't want to keep 400 reels of tape. I wrote a utility program to combine these. I call it a utility program but it had no direct bearing on a production process; it was a quick and dirty kind of thing. I classify as utility anything that is not in the direct line of

production but assists the production process, including assistance to the operator. In the early days, we did some of our own diagnostic programming for special kinds of things where we knew that we were having hardware difficulties. I think people would kind of classify utility programs as being assemblers, loaders, dumps, patch routines. For example--oh, this is a horrible kind of thing--we would assemble a program, get it out in some form of binary and then load it. If something was wrong with it, it would be too expensive to go through the assembly process so, we would write something where corrections could be made to that deck. At one time people used to do weird things. They would take this binary card, which among other things had a check sum on it, and they would put Scotch Tape across the hole to fix that instruction, and then they would have to correct the check sum the same way. Then they learned that you can take the chips out of the tray and put them in there. One girl never learned that and she jammed the whole card header with Scotch Tape. (Laughter) "Why do that? We can write a little program to do the fixing for you." There were some people, and this is amazing, that would have the deck that had been assembled and produce the binary maybe an inch thick and patches about ten

inches thick behind it. One of the reasons I insisted on being able to make patches at the symbolic level was to get these people out of their bind. They would have litte decks of symbolics with hugh patch decks behind them. Some people can't be broken of habits. Anyway, those are the kinds of things I classify as utility.

When were we beginning to recognize the need for it. As I mentioned on the JOHNNIAC, the program that I used to write this first production job was written in absolute octal. Absolute octal is very difficult to write to begin with and extremely difficult to change.

RM: Why is that?

IG: Because if I have an instruction that I want to replace with ten instructions, since everything is absolute, what I have to do is branch out, put in the ten instructions, and branch back. That leads to a nightmare in terms of what that program begins to look like over time. What you would like is to get at least one step up from that, such that I have a program written quite independent of the locations

in which it's going to run, and I can put changes in where I see fit and go through the assembly process again.

I might mention the history of assembly programs, which is a little bit interesting. We started out, as I say, with absolute forms. Then we realized early on that the machine is good at clerical jobs, so you wrote things in what I'll call regional, meaning that every instruction has a location assigned to it except that location is an offset from some base which has yet to be specified. Then what we did was stick in the bases as part of the assembly processing. The machine can do that; you could do it on a sorter. And then somebody said, "Why are we putting a label on every instruction? Why don't we just put labels on the instructions that are referenced?" Then we got to symbolic assembly programs. The first example that I ever saw of one of those was the SAP assembler that Roy Nutt wrote and about the same time G.E. was writing a similar assembler, both for the 704.

RM: Of course back East Grace Hopper at Univac had already gotten into this background.

IG: Well, she had some strange views of life, but I'm really unfamiliar with the early Univac assemblers. Early on, Grace was interested in compilers and there were some smart people either working for her or associated with that. Tolly Holt is one of the people that always comes to mind. Holt and Turanski, those two names, go together. Turanski unfortunately died quite early; I think he was hit by a car or something. Brilliant man.

I recall early in my career going to an ACM meeting at the University of Pennsylvania and Grace Hopper got up and gave a talk about compilers. After the meeting was done Don Madden introduced me to her out in the hall. In my unsophisticated innocence I asked Grace, "How does a compiler compile code?" I had no idea how these things worked. She forever endeared herself to me when she looked me straight in the eye and said, "Better than any human." (Laughter) Beautiful woman.

RM: (Laughter) Oh, that's lovely.

IG: Anyhow, so much for utilities. What are we up to now?

RM: Do you feel that in a way then that you were really pioneering assembly programs even though other work was being done elsewhere?

IG: No. I guess there were certain things that I felt strongly about. One of them was the idea of symbolic modification. I guess I pioneered that, I sold it, lots of people bought it and it's current practice. I have always been more concerned with the service aspects of computers, and that's one of the things that I don't like about working for a manufacturer. I have never particularly believed in general purpose capabilities. You always have to compromise. I understand the need for it, but I am much more interested in the service function. We have a job to do for those people who are using computers and how can we best do that job. Now, I happen to have gotten involved early on in the thing called systems programming and I'm kind of stuck with it now, but I much more enjoy doing something like the JOSS system, which is customer oriented, or the RAND Videographic system, which in a sense was customer oriented where the costomers happened to be other programmers. Along with lots of other people, I prefer closed systems to open.

In a sense there were certain things I kind of pioneered in; I guess I wrote the first published paper on macro-instructions. It was not the first work ever done but it was the first paper ever published.

RM: Where was it published?

IG: In the ACM Communications in 1958. I have a vague recollection of November, but don't quote me on it.

RM: Do you remember the title of it?

IG: "A Technique for Handling Macro-Instructions," which is kind of interesting, too. In those days the Communications, in my estimation and the estimation of lots of other people, were a quick interchange of information, tutorial, "what have I done that you can profit from" and what have you. One of the unfortunate aspects of all the journals in our field is over time they degenerate, or grow, depending on your viewpoint, into much more formal publications in which the neophyte can't learn a damn thing. Communications of the ACM were beautiful when they started. I think they even referee the papers for them now. So they substituted the Computing Surveys, I guess, which started out having beautiful tutorial articles and now has a bunch of jibberish which nobody can read. Right now the best publication in the field, to my mind, is Software:

Practice and Experience which is oriented toward "What I did and how I did it," rather than, "What I might do and why I might do it."

RM: In other words, it's factual.

IG: Yes. Unfortunately, and I don't quite understand it, at Xerox, and I don't know if it's true at other places, it's amazing how few people are familiar with that publication.

RM: I'm not, but that doesn't mean much. To go back to your macro-instruction paper, what were you saying at that point?

IG: I was telling people what I did, both because I was interested in it and because I thought we might be able to make use of it at RAND. I wrote a preprocessor to the SHARE assembly language program which handled macro-instructions in a very limited form. I wrote a paper on how I did it so that other people who were interested might be able to do it. For all intents and purposes I stopped looking at macro-instructions after that, but everybody thinks I'm an expert. Just last week I read a paper that cited mine.

RM: In 1973?

IG: It was a tutorial kind of thing because I knew that most of the people at RAND who were using it thought

it was black magic. It isn't, nothing's black magic. Some of the things are very hard work, others are very easy. So I wrote down what I did so that other people might learn from it.

RM: What is the state of the art today? Is it macro, micro or other?

IG: Every manufacturer has a relatively sophisticated macro-type processor; macros can be personified by talking about them as parameter substitution by string. For example, the simplest case that everybody cites is: I can create a macro instruction, called sum. Sum A plus B and store in C. In some point in time you can use this macro by saying sum X,Y,Z. That is about the extent of the paper I wrote in 1958, and people have added sophistications to these. Most of the pseudo-instructions that an operating system uses are supplied in the form of macros to the programmer using an operating system. For example, a programmer might write something like, "Read my file," and that expands into a whole bunch of instructions which the operating system does various things with.

Now I'm going to mention something that I can't explain

the difference of because I don't fully comprehend it myself. We have it and other people have it and Dave Ferguson from Programmatics was the first one that I know of that promulgated the thing. Currently, we have something called a meta-assembly. His idea was to write a language in which I could write assembly programs. It turns out that that language has great facilities for writing programs, per se. I really can't make the distinction between the macro and the meta, except I recognize the advantages of one over the other in certain applications. I'm not a language theorist by any shape of the imagination.

RM: A term which is bandied around a lot and I don't think I've every fully understood it is pseudo-instructions.

IG: Okay. The machine repertoire has usual instructions, arithmetic, manipulative, what have you. The assembly language programmer wants other instructions in order to control this assembly process. For example, in a symbolic assembler he wishes to be able to equate one symbol to a location, he wishes to reserve blocks of storage data that he's going to read into, he wishes to control the format of the listing of his program so he has space control, page eject, headings, and what have you. All these things that are not part of

the repertoire of the instructions of the machine are called pseudo-instructions. I guess that word has been walked around and has gotten larger meanings over time, but that was the first meaning of it.

RM: When did it first start to be used?

IG: The first time I saw it was, again, with the SAP United Aircraft assembly program for the 704. There were pseudo-instructions which had nothing to do with the machine, but were for control of the assembly process itself. That kind of thing had been used a little bit in compilers. For example, and they call it macros again, which is fascinating, in PL-1, which is a language for writing programs. There is this language for writing program, for specifying algorithms, data structures, and what have you. Then there are a class of instructions to tell the compiler what to do, having nothing to do with the operating program. The architects for PL-1 call those macro-instructions to the compiler. This all gets in to the very fuzzy realm of the extensible languages. Any language anybody domes up with will not be either accurate or ideal for a given user and so he wants to be able to extend the language for his particular problem.

GREENWALD

PL-1, on paper at least, has done this through a set of directives to the compiler itself. Have you got George Mealy down on your list?

RM: No, that's a name I know but I haven't got him down. Why should I have him?

IG: George Mealy, who is now at Harvard, is one of the brightest people in computing, understands all this stuff, understands the implications of lots of things. Talking about extensible language, George has just sent me volumes of stuff of work that some people at Harvard had done on extensible languages. There's a whole class of people on the East coast that have done pioneering work in very many ways that most current people in the field are unaware of, don't understand and don't appreciate. A lot of this comes out of a place called Tech-Ops which was in Washington, headed by Tom Cheatham who is also now at Harvard. There are a set of exceptionally smart people there, especially in programming rather than in the hardware sense, who did some pioneering work which people have never understood and which I think might be interesting.

RM: Pioneering work of which era?

GREENWALD

IG" Mid-fifties on. For example, Tech-Ops produced an operating system called CL-1 and later on CL-2, which today people are designing without even being aware that Tech-Ops produced it way back then.

Something that just struck me from the point of view of the history of computing is that all I've been talking about thus far are things that are production oriented, doing jobs for customers and what have you. Around the late 1950s or thereabouts, lots of installations became concerned with doing research into the computer sciences. RAND did, SDC did, the universities even more so. About the same time, ARPA decided to start funding some of this stuff. I think it's through the interest of various organizations plus the funding from ARPA that an awful lot of developments have occurred. All the area of graphics work primarily comes out of that. Organizations such as General Motors did a lot of graphics work as an aid to their auto design process. But graphics work, time sharing, the computer utility concept, I think primarily result from interest of people and the fact that ARPA started funding. Things like McCarthy's work with list processing and artificial intelligence.

One of the reasons that this struck me now rather than before is that I have worked for a manufacturer for about four years who tends not to use those kinds of things. Some of the rather important things going on in computing, things that have gone on, things that are currently going on, and things that have an implication for a long-term future, I think are coming out of research in computer science. I'm not engaged in that now so I can't talk an awful lot about it, but we went from pragmatic solutions to practical problems to people thinking about "how would I solve general problems" which led to research and, for whatever reasons, ARPA funding many of these things.

RM: ARPA is what?

IG: Advanced Research Projects Agency of the Department of Defense, and particularly the Information Processing Technology office, which to my knowledge was headed by J.C.R. Licklider, then by Ivan Sutherland who did Sketchpad, the first primary research into graphics at MIT. Then he became head of that office with a rather large budget.

RM: Is that where graphics really got started?

IG: Sketchpad was a big step there.

GREENWALD

RM: Around what time was this?

IG: Oh, 1956, 1957 I guess. He became director of the IPT office of ARPA, followed by David Taylor who is now at the Palo Alto Research Center with Xerox, followed by Larry Roberts who's been there for ages. I believe Roberts did his thesis, at MIT, on work following up the Sketchpad kind of concept. While to some extent, unfortunately, there's an ingraining of MIT in all of this, a lot of money has been spent that has, amongst other things, kept people gainfully employed, but also on the frontiers of what's going on. Whether manufacturers will apply those over time or not is debatable.

For example, the recent IBM announcements, virtual memory systems, were pushed very hard because of the Multics system in which they could not supply what MIT wanted. Therefore they went to G.E.(which has since been sold to Honeywell). It would be interesting to get a consensus of opinion on the impact of that kind of stuff in computing versus what the manufacturer has done.

RM: What was G.M.'s role in this?

IG: Oh, G. M., on the 7094 I guess, built the DAC system. I don't know what that stands for. Ed Jacks of G.M. was in charge and they were interested in aiding the auto design process such that the design engineer would sit at a graphics terminal, draw pictures and get various views, perspectives, as well as some idea as to the physical characteristics. I can state this a little better for the airplane designer. Both Lockheed and Boeing, I know, had an awful lot of work going on in, for example, the design of the wing in terms of the airflow characteristics, the structural characteristics, how strong was it, varying the design and getting out a bunch of numbers that reflected the interesting parameters.

RM: They were able to do not only a physical simulation but also a visual wherein they could...

IG: Right.

RM: . . . figure out simulation by mathematics; they could see what its design was going to look like.

IG: Yes. The field of architecture has been very interested in graphic stuff and I think some has been applied. Brown University has a beautiful publishing system;

GREENWALD

I think they call it Hydra. Andy VanDam is the guy who is chiefly responsible for it. It takes composition, editing, etc., and working at a big, graphic display you eventually wind up with a publication. There's an awful lot of work going on in the publishing industry in the use of computers.

RM: What was the manufacturers' response to this need?

IG: Well, it varied.

RM: In fact, which came first? I guess the CRT tube was there.

IG: Yes, back in the original Sketchpad days. I guess they were using Whirlwind at MIT; though I'm not sure. The computer sciences courses at MIT are a combination of engineering and programming, so people would build their own stuff and hook it onto the computer and use it. The manufacturers' acceptance or use, in my very unknowledgeable estimation, varied all over the place.

For example, IBM, who despite a lot of things they say, to my mind is mostly interested in commercial data processing because that's where the money is, could not see its way clear to doing an awful lot of support of this kind of effort, time sharing being one of them; large addressing space, virtual memory

concepts, being the second one. By the same token, they had an awful lot of work going on with the 2250. I don't remember the guy who was involved in that, but he gave a name to one of the systems.

DEC, for example, who is in the MIT, Harvard area, has a large customer base in the universities, especially for the PDP-6 originally, which became the 10, and the PDP-11, is much more attuned to the needs of the researcher. This serves a dual purpose; they are attuned to the need and they also gain from it. As an example, DEC built the PDP-10. Bolt, Beranek, and Newman, an ARPA contractor, built a paging box for that machine and built an operating system; now DEC is marketing the PDP-10I which has that box built in. A good many of the ARPA contractors have DEC equipment and so DEC is able to profit by this. For example, in an indirect way they are part of the ARPA network because an awful lot of their computers are on the ARPA network; whereas SDS and Xerox, who did have a computer on the ARPA net, the Sigma 7 at UCLA was being used to monitor it, didn't take advantage of that connection. A lot of people were very upset that they didn't take advantage of it. DEC has profited from that. They profited from the fact that

an awful lot of the programs used by researchers are available on their equipment and not on others.

RM: How about talking about the classic argument of floating point versus fixed point?

IG: Oh, whew. I'm a fixed point man, but before proceeding with my arguments about fixed point I have to allow that without floating point a hell of a lot of work wouldn't have gotten done.

Fixed point I can recite from somewhat of a historical perspective. When I first started doing some of the problems I mentioned earlier, especially the shock wave problems all of which were done in fixed point initially, one of the things we had to be concerned with was scale factors; how big can this number be, how small can it be. On the first iteration on one of these problems I came to the conclusion rapidly that we had zero significant digits when we got all through, which said something was wrong. You start talking to the people who know the physics and point out that when you are multiplying X times Y, and given that the range of X is 10^{-1} to 10^{+14} and the range of Y is the same, there's no way of keeping significance. The physicist points out that they vary inversely with one another so that the range of

the product is like 10^{-1} to 10^{+3} which is dealable with in some sense of the word. The thing about fixed point is that it takes a lot more work and therefore problems don't get solved as rapidly. But it also means that when you get through you know what you've done. I might backtrack and say I am not a numerical analyst; I am a computer arithmetician, if you will. I appreciate the problems though I don't know the solutions, so I have to go to the guy who knows the problems to get the solutions. It involves thinking and knowing what you're doing, and gives you some confidence in the result you finally get. With floating point people are coding things without thinking. In the scientific world it's argued that because things have to plot smoothly, everything's okay anyhow.

We had an interesting experience. Our Card Programmed Calculator boards were all floating point and at RAND each job was assigned to two people who coded them independently. Then, when their results compared within some tolerance, they were assumed to be correct. Another guy and I did one problem, we ran it, and the results didn't compare in the first digit. We started

analyzing the results and it turned out, and this was purely accident, that I had one and a half significant digits when we were through and he had half a significant digit. The question that comes up is how many problems are being run that way? Again, given the number of people who are currently programming, I don't think it's possible to teach them all how to do fixed point.

We ran another interesting experiment at RAND. We wired up what we called a range board in which all intermediate quantities were computed to their maximum and minimum, which meant that the ultimate results were computed to their maximum and minimum. We discovered that the answer lay somewhere between plus and minus infinity. (laughter)

Bill Orchard-Hayes pointed out something interesting when he was doing linear programming. By all analysis he could not justify keeping anything to double precision because the second half of the double precision was garbage. Yet, by empirical evidence he knew when he kept that garbage he got results that were meaningful and when he didn't he had results that were not meaningful.

I guess my feeling is anyone who is doing scientific computation should at least be trained in what fixed point is all about so he has some understanding of it. I don't know how you go about utilizing it in this day and age. It's always scared me. We have

other idiocies. In our floating point we used to put out results to eight digits; people would put in three digit numbers that they read off of graphs and come out with eight-digit answers and publish them. The whole thing just kind of bothers me. That's about the most I can state about it.

RM: Isn't floating point used, primarily, for scientific computing and not used in regular business data processing?

IG: I'm not that familiar with business data processing. For the most part they would tend not to need floating point, though in some of the work they do they might. But even in business data processing the crossfooting quite often doesn't come out to the penny the way it's supposed to.

That gets into something else. With binary computers we tend to work with, if we're lucky, binary integers to some power of two, eight, or sixteen, quite often with binary fractions to some power of two or eight or sixteen. As long as you work with integers you have an exact internal external representation, but as soon as you start working with other than powers of ten, you don't. One of the things we did with the JOSS system was work with integers with powers of ten internally. It's not that hard to build that into hardware; in fact, we proposed to do that with a

Xerox product line that has since been canceled. But just to get this kind of exact representation. That's another thing that always troubles me: why an engineer has to be faced with the fact that when he sums .1 ten times he gets .99999. There's no excuse for that. If he sums one-third three times he can expect not to get 1, but that's a little different.

RM: What was your involvement with JOSS, if any?

IG: JOSS was originally done by Cliff Shaw on the RAND JOHNNIAC and was successful enough to where we decided that we ought to make it more widely available within RAND. A group headed by Chuck Baker, consisting of Joe Smith, Ed Bryan and myself, set out to build a system that could support more users, could do more for them, and what have you. Joe Smith did the language interpreter; Ed Bryan did the supervisor; I did the console I/O, the disk I/O file storage, and all of the arithmetic which was done in the form I just described, and elementary function routines. That is kind of interesting too. Three people are about the maximum, I believe, that can work together on any project. Therefore, when manufacturers are putting thirty to five hundred people on a project, you know you're going to get a mess.

GREENWALD

RM: What did JOSS actually do and what did it give that you haven't had before?

IG: Well, I don't know my history that well, but JOSS may well have been the first system which put the user directly on line to a computer. It's cited in some of the literature as being the first. It was designed for the solution of rather small numerical problems. It had no string handling capability or any of that stuff. It was designed to be very simple to use, high recall value; to be fairly natural: for example, the exact numbers are exact inside and outside the computer; and, as I say, simple to use. We did some analysis which said, that for numerical problems a guy sitting at a JOSS terminal, which by the way is a typewriter, not a teletype, has at his disposal approximately the power of a four to eight-K 704. For the RAND staff, and in time for Air Force clients at RAND who started using it, it has been very widely accepted.

One of the difficulties with JOSS, and I believe a mistake that was made, is that RAND put a copyright on the name JOSS and would not release that title to anyone who didn't have essentially an exact copy and, in particular, a terminal like a typewriter. Had they not done that, I believe JOSS might have

been quite competitive with BASIC.

RM: What was the reason for doing this? It seems to be counter to the whole philosophy of RAND.

IG: Well, there are two ways of doing it. One is you want to promulgate this kind of way of doing business, which says that you ought to release the name and get lots of people doing it. The other is, you have a definite view of how it should be done and anything that deviates very much from this is not the way you want to do things, and therefore you won't release the name. Cliff Shaw was of the latter school, because he was the originator of JOSS, he had a great deal of influence on the management of RAND. Originally I felt the way he did, although quite early in the game, I came to the opposite conclusion. That was a decision somebody had to make, and I believe the wrong decision was made. However, it's hard to evaluate what would have happened had the other been made. One of two things; either JOSS would have been a more popular language or JOSS would have been totally bastardized. There are lots of copies of it but they all have different names. In fact,

there's one in Britain called JEAN.

RM: You made reference to the difference between assemblers and compilers, and I think that's worth talking about. What is the difference? What is an assembler? What makes an assembler an assembler and a compiler a compiler?

IG: They're both translators. They translate from some language into another language. The assembler is personified by being very close to the language of the machine. In general, there is a one to one translation. The early assemblers, instead of writing something down in binary or octal or what have you, wrote down with a mnemonic and symbolic designations for things. In general, they're one for one; they translate from some external representation of an instruction to an internal representation of an instruction.

Compilers are procedure oriented languages which have nothing to do with the instruction repertoire of the machine. For example, you express an assignment statement for $A=X^2+Y+Z$, which has very little bearing on what the instruction repertoire of the machine is, and they're many to one.

It's kind of fascinating. I don't know where the word assembler originated. It's a terrible way to use up a word because the assembly process has to do with the combining of elements of a program into a whole, and we can't use it for that purpose any more because it has a very definite meaning in the computing profession, which is unfortunate.

They're both translators; one is at a microscopic level, the other a macroscopic level. Compilers attempted to make programs machine independent. They didn't suceed particularly well at that. One of the things that a lot of people did not recognize is that programs are not machine independent so long as the data structures are machine dependent. However, the basic difference is a language very close to the language of the machine and a language far away from it. FORTRAN, COBOL, PL-1 are relatively, as languages, machine independent, whereas the assembler for the IBM 360 is very dependent on the 360 and the assembler for the SIGMA line of Xerox is very dependent on it.

RM: One thing I haven't really gone into with anybody that maybe you can talk about, is SDC and how it came about.

IG: Okay. I know a little bit of that. I was involved. A bunch of psychologists, (some names were: Chapman,

GREENWALD

I can't recall his first name, and another was John Kennedy). Anyhow, Chapman and Kennedy were psychologists at RAND who were interested in ways of improving performance of people. They selected as their vehicle of study, improved performance on the part of radar operators. This was imbued in the Systems Research Lab at RAND.

They set up some training aids for radar operators and hired a bunch of college kids to come in and see if they could improve their performance as radar operators. The Air Force became cognizant of this, looked at it and said, "Hey! Maybe it can really improve the performance of real radar operators." So the System Research Lab became the System Training Project for the RAND Corporation in which they brought in Air Force types and discovered that indeed they could improve performances.

The type of things that we were doing at that time, were done on EAM equipment. We were producing quarter second by quarter second profiles of what a radar screen would look like with certain loading of aircraft. These were printed out on tab runs from a printer and run through a strange, black

box that clicked over at the radar interval. At any rate, that gave rise to STP, the System Training Project at RAND, and the Air Force got very enthused.

The System Development Division of RAND was then formed to carry on the training of Air Force radar operators. It was very upsetting to Chapman and Kennedy who never did get to the point where they could analyze and publish the results of their original experiment. Then in December, 1957, the System Development Corporation was formed to do this. Well, somewhere in that interim, I guess it all started around 1952, we began to switch the whole thing to the 701. I'm trying to remember. That's another first: RAND got the first CRT ever hooked to IBM equipment as part of this whole thing. It seems to me that was on the 704, though.

At any rate, sometime, I guess it must have been earlier that 1957, the Air Force asked RAND to get involved in the SAGE System which was being done by Lincoln Labs of MIT. There were about five people from RAND that originally started out on this.

Wes Mehlan was one, and he became president of SDC. He could probably tell you this history a little better. Who were some of the other key people? I can't really recall who those other people were. Pat Haverty comes to mind. I can't remember all their names. I'm getting a little mixed up as to what part SAGE played and what part STP played. At any rate, STP got into the act somewhere along the line, and SDC was formed to both do the system training exercises as well as support the SAGE project.

RM: The support of the SAGE project was in Systems Training, is that correct?

IG: For sure that was part of the Systems Development Division. It may have also been part of SDC. At any rate one of their functions was to go back to Lincoln Labs, learn what those people were doing, and this was where the 500 programmers came in, and supply the people to man the field sites. I think they were estimating, in the order of ten people for fifty sites at that time. Gradually, as SDC bloomed, other things got into the act; the back up intercept, something or other which was called BUIC, and a host of other contracts that they got involved in.

GREENWALD

The whole thing grew out of this original interest on the part of two psychologists to study ways of improving training.

RM: And they never really finished their work?

IG: They never finished their work. I don't know what became of Chapman. Kennedy, at some point during his career, went to the Institute for Advanced Study at Princeton. I know that he came back as a consultant at SDC.

RM: Did they document their work in any way?

IG: I don't know. I don't know.

RM: Do you know what kind of methods were used for this training?

IG: I can only give you some overviews. They're the people to talk to. John Matousek was deeply involved in this and he's still at SDC. I would guess by now you've already talked to Wes Mehlan.

RM: No, I haven't because he's in Boulder.

IG: What's he doing in Boulder?

RM: He's at the Center for Atmospheric Study, something like that.

IG: Maybe later on I can think of other people for you to contact that are still at SDC.

GREENWALD

RM: I want to talk to some key person.

IG: Pat Haverty is at SDC now. He was involved in the early SAGE days.

RM: Would he know about the whole development?

IG: He might not know the real early stuff, but from the time of SAGE for a goodly number of years he will know what's going on. He's also a good source who might be otherwise helpful.

RM: I think I should pursue it because it really had its start quite early in the RAND structure.

IG: Also, from an advance-in-computing point of view, an awful lot of the work done at Lincoln Labs in 1957 in terms of how you write programs, how you test them, et cetera, et cetera, especially very large programs, is in many installations still the state-of-the-art. It's kind of inexcusable that lots of people haven't learned from their experience, and they haven't.

Jules Schwartz, who is at Computer Sciences now, was involved in those early days, too. When I say early days I'm talking of early SAGE days, not early STP days. The people who were involved in the early STP days were Chapman and Kennedy, myself, John Matousek, and Allan Newell who is at Carnegie-Mellon now. In many ways he was the interface between

Chapman and Kennedy and our department. He was the guy that came down with all the criteria, the specifications that John brought in.

RM: Newell worked on JOSS, is that right?

IG: No.

RM: He worked on something.

IG: Newell, Simon, and Shaw worked on IPL-5.

I don't know exactly how they proposed to measure all this stuff, but one of the things that they did was to run these flight patterns through, and there were hostiles as well as our own aircraft. I remember going over to the lab and hearing Chapman's voice come over the loudspeaker saying, "We have just lost Seattle." (Laughter) To my limited knowledge, it was just a matter of did we stop losing these cities. I'm sure they had better techniques for evaluating the results.

RM: A whole fascinating area, it really is.

IG: I'm trying to think who all else. Don Madden, who is in Palo Alto now.

RM: Oh, I'm seeing Don next week.

IG: He would know a lot of that early history.

RM: I'm going up to the bay area to see Paul Armer, Don

Madden, and Joe Weizenbaum.

IG: Joe Weizenbaum is beautiful. He's such a beautiful person. And so smart!

RM: (Laughter) You've just opened up a whole new area for me.

IG: Joe Weizenbaum probably also knows Cheetham, Mealy, and those people and can talk about that.

RM: My reason for contacting Joe is ERMA. What other areas should I research?

IG: Which one's ERMA?

RM: That's the big G.E. monster.

IG: That's Joe Weizenbaum from MIT?

RM: Maybe we're talking about a different person.

IG: One way to find out is to ask him about ELIZA.

RM: Yes, ELIZA. I think that's the same guy. ELIZA is the language recognition.

IG: That's the one that talks to you.

RM: Right. You can get yourself analyzed.

IG: It's got to be the same Joe. Somebody else that you might talk to is Joe Smith.

RM: That's a good name.

IG: Yes. It was very funny, when I was at SDC I had just acquired a secretary and Joe was over at RAND at the time. The secretary walks in the door and says, "Some character wants to talk to you and he insists that his

GREENWALD

name is Joe Smith." (Laughter) Joe's down at NCR in Rancho San Bernardo. Joe has some interesting background. He comes out of Tech-Ops. He also comes out of Purdue with Perlis. I don't know if he ever went to Carnegie or not, but he did a lot of work on CL-1 and CL-2 at Tech-Ops. He worked at RAND, as I mentioned he was on JOSS amongst other things. He's one of the guys that talked me into going to Xerox.

RM: Irwin, thank you very much.

(END OF INTERVIEW)

Index - Irwin Greenwald
3 Apr 1973