FORTRAN III was designed by Irv Ziller. It provided the ability to intermix FORTRAN statements and machine instructions with FORTRAN variables as addresses. It also introduced a number of features such as Boolean expressions and string handling facilities that later appeared in FORTRAN IV. In fact, most of the facilities of FORTRAN IV are to be found in FORTRAN III. It was mostly an internal IBM system and was first used in the winter of 1958–1959.

So, in closing, let me remark that there seems to be general agreement among the original FORTRAN group that it was one of the most enjoyable projects any of us ever worked on. We were graced by this wonderful independent status that Cuthbert Hurd and my subsequent bosses conferred on us, so that we were completely on our own. We knew we were going to provide a valuable tool, even if few others did, and we worked quite hard. But perhaps the best part was the uncertainty and excitement of waiting to see what kinds of object code all that work was finally going to produce. I only wish that programming today could be half as exciting and enjoyable as the FORTRAN project was.

Thank you.

TRANSCRIPT OF DISCUSSANT'S REMARKS

JAN LEE: Thank you, John. We invited John to prepare a paper on FORTRAN, and then we searched around to find somebody who might have some other thoughts, and I use that word "other" very carefully. We were somewhat concerned it might be "opposite" thoughts, "corroborating" thoughts, or any other. And we looked hard and long. You may notice that John said that one of the motivations for FORTRAN in those early days was this growing space industry. And so we looked to a down-to-earth space industry, and came up with General Motors, as a source of a Discussant. And then went to our friends at General Motors and said, "Okay—who was it out there who really bore the brunt of trying to get FORTRAN operational and knew what was going on?" and came up with George Ryckman. George has a Bachelor's Degree from the University of Virginia, and then went to the University of Michigan where he obtained a degree in Electrical Engineering. He's been at General Motors since 1952, starting off as a research engineer, supervisor of computer operations when FORTRAN was getting its feet on the ground, and now Assistant Department Head of the Computer Science Department at General Motors.

GEORGE RYCKMAN: Thank you, JAN. My remarks will deal primarily with the impact FORTRAN had on our organization, General Motors Research Laboratories, although I will have a few comments on the language design and its implementation. I'll begin with some pre-FORTRAN history, followed by our early experience with FORTRAN, some operational issues, and finally a comment on our more recent history.

In the three years prior to the first distribution of FORTRAN, we were preoccupied with programming and operating our first stored program machine, the IBM 701, and in preparing and installing its successor, the IBM 704. Still we took advantage of several opportunities to examine, critique and plan for FORTRAN at our installation. For example, my log book notes on March 17, 1955: "Received proposed additions to the FORTRAN: incorporated new symbols and formulas." And another entry on March 16, 1956: "FORTRAN code plates ordered for the keypunch machines."

We also suggested the IF SENSE LIGHT and IF SENSE SWITCH statements be added in order to monitor and control program execution at the computer console itself. From the first, FORTRAN seemed to offer inherent advantages over a myriad of langauges that we had up to that point, including Speedcoding, ACOM, which was a local three-address system we had, and a number of absolute, regional, and symbolic assembly languages.

From the standpoint of the engineer or scientist, the algebralike formula of FORTRAN appeared to have many specific advantages. The language was closer to his thought processes, required fewer pencil marks to program an algorithm, was easier to learn and retain, easier to read and change, and lent itself to more accuracy. All of these attributes netted out to two important predictions: first, the existing 15 to 20 full-time programmers would be more productive, but more importantly, through training classes and opening up the programming process to other groups, we could relieve the programming bottleneck. The big unknown, of course, at the time, was the other side of the equation; namely, the cost and computer time for translating and executing FORTRAN programs. And even though IBM was placing great emphasis on minimizing this cost, the answer had to wait until we had experience of our own.

The FORTRAN Translator arrived in April, as I remember—April of 1957, and by that time manuals had been distributed, some training had been conducted, and some programs had already been written. These programs included solutions for simultaneous algebraic and differential equations, an interpolation routine, and a program for graphing engineering results on a cathode-ray tube plotter. We very early learned that the Translator's diagnostics were not among its strong points. For example, the uninitiated could never understand why a missing parenthesis was cryptically known as a "nonzero level reduction." It was great sport in those days to scan the object program and either marvel at the Translator or question its sanity!

In spite of these problems, along with many translation errors, I can report that FORTRAN was a major success at GM Research. Between the internal and open shop programming groups, 40% of the 704 computer load was in FORTRAN by the end of 1957—a matter of eight months. A large fraction of this FORTRAN workload was converted from SAP but a more significant portion was worked generally by the other groups. Undoubtedly the rate of increase in computer workload at least tripled over what it would have been with the internal programming group using SAP. This estimate is based on some programmer productivity measurements that we made at the time.

Setting aside problem formulation and analysis time, studies showed that the programming and coding effort using FORTRAN was reduced by a factor between 5 and 10, compared to assembly language. Added to this, of course, was the fact that more people were trained to program the computer using FORTRAN.

Accompanying the reduction in programming effort was a decrease in the overall cost of the typical programming assignment. Taking into consideration the computer plus people costs of programming, compiling, recompiling and debugging, a cost reduction of $2\frac{1}{2}$ to 1 was achieved over a large set of programs.

In the real world, of course, these savings had less meaning simply because programs became larger, were more complex, and therefore the direct comparisons were no longer possible.

The greatest single design weakness of FORTRAN I was the lack of a full subroutine facility with load-time linking, because this generally limited the size of programs. In fact

the mean free error time on the 704 was a major factor in running large programs; it just didn't hold out that long. The economics of compiling, testing, and recompiling were typically prohibitive with programs exceeding 300–400 statements. Later with the advent of FORTRAN II, the ability to break up a program into pieces substantially removed this restriction and the language became firmly established with virtually all application programming at GM Research.

At this point I'd like to address some of the operational issues of the earliest FORTRAN, the manner in which the compiler/translator ran on the 704. It did not come with an operating system, but rather took over the whole machine when it ran, so a programmer typically approached the machine, getting four tape units ready, a printer, a card reader, and a card punch. During translation, listings appeared on the printer and a binary object deck issued forth from the punch. The programmer transferred this binary deck from the punch to the reader, pushed "GO," and hoped. He finally retrieved the printed output and the card decks, and that completed a session. Rarely could all of this activity take place in less than 10 or 15 minutes per session, so machine time was scheduled in 15-minute blocks. All of this, I should say, was in marked contrast to our normal mode of operating a machine. Earlier in 1955, in a joint effort with North American, we had designed and implemented a so-called "multijob, tape-in, tape-out monitor system" for the 704. I guess today we'd call it an operating system. The resulting throughput advantages prompted us to initiate a similar system for FORTRAN, and in fact, we had already designed and all but checked out a new system for FORTRAN at the time it arrived. Within a month it was also running in an operating system. Incidentally, after suggesting SENSE LIGHT and SENSE SWITCH additions to FORTRAN, this nonstop automatic operation was rendered almost useless.

Fortunately the earlier monitor had done a good job of tuning our computer system because the translator was highly reliant and extremely brutal on tapes and drums. From an operational standpoint the only problem that plagued us for any extended period was the numerous program halts built into the translator. But the translator eventually learned that we would not put up with such laziness, and that if it had trouble with one source program, it should simply go on and get the next one.

In the intervening years, the FORTRAN load built up to virtually 100% of our computing. Only in the last ten years has it yielded grudgingly to PL/I; until today it still accounts for 50% of the load on three IBM 168 systems.

Our best, most efficient numerical analysis codes are still written in FORTRAN, as are some huge programs for structural and fluid flow analysis, just to name a few.

I've enjoyed delving into the archives for this discussion and also sharing it with you. I wish to thank the sponsors for the invitation to join you and the privilege of participating.

TRANSCRIPT OF QUESTION AND ANSWER SESSION

JAN LEE: Thank you, George. So, John, let me start off with the most general question. It is from Nancy Stern, and really it's not directly related to FORTRAN. "I notice that in the May 1954 article, the term 'programmer' was used. From your experience, when did the term 'programmer' begin and replace the word 'coder'?"