July 30, 1957

Mr. John Backus IBM Corporation 590 Madison Avenue New York, N. Y.

Dear John:

I am enclosing a copy of a paper titled "FORTRAN EXPERIENCE AT THE NEW YORK DATA PROCESSING CENTER" which I presented at the recent SHARE meeting in Dallas, Texas. I thought this might be of interest to you since the Data Processing Center has had more experience using the FORTRAN System than any other group in the country. Additional copies are available if they would be of interest.

Very truly yours,

Brun Olifiely

B. G. Oldfield, Manager New York Data Processing Center

BGO:pg Enc. cc: Mr. J. McPherson-

FORTRAN EXPERIENCE AT THE NEW YORK DATA PROCESSING CENTER

The following remarks and observations concern experiences of the New York Data Processing Center in using FORTRAN through April 1957. Our experience began many months ago with an abreviated FORTRAN system which still contained many errors and which did not include the FORTRAN output system and the FORTRAN error detecting features. Our more recent experience with the complete FORTRAN system in a relatively error free condition is more indicative of what to expect from FORTRAN and more valuable in trying to make an initial evaluation of the system.

The following types of problems have been programmed using FORTRAN, and the statistics which follow were gathered from these problems.

- 1. Computing the coefficients and evaluating the smallest real root of quartic equations. This problem required 47 FORTRAN statements.
- 2. A problem concerned with the design of a distillation column requiring 700 FORTRAN statements.
- 3. A complicated helicopter design problem requiring 1250 FORTRAN statements.
- 4. A quardruple integration problem with complicating integrands requiring 140 FORTRAN statements.
- 5. A relaxation problem which required 800 FORTRAN statements.
- 6. A projectile flow problem requiring 133 FORTRAN statements.
- 7. A problem concerned with the investigation of the action of pile driving hammers which required 93 FORTRAN statements.

A total of 7 different problems were considered involving 3163 FORTRAN statements.

Expansion Ratio

The number of machine language instructions produced for each FORTRAN statement is commonly termed the expansion ratio and this ratio varied between 5 and 21 for the different problems considered. The 3163 FORTRAN statements produced approximately 22,000 machine language instructions excluding subroutines and input and output. Consequently, our average expansion ratio has been 7 machine language instructions produced for each FORTRAN statement.

FORTRAN Statements Written Per Hour

If one considers writing time to include all time spent on a problem up to the time the problem is ready to compile, our experience has been that an experienced programmer can produce from 4 to 12 FORTRAN statements per hour and our average is approximately 7 FORTRAN statements written per hour.

Compiling Speed

Compiling time will vary considerably depending upon the number and type of FORTRAN statements to be compiled. For example, two different sets of 250 FORTRAN statements required 8 minutes and 20 minutes respectively to compile. There is also a point at which the compiling time seems to increase rapidly relative to the number of FORTRAN statements being compiled. Our average compiling time is approximately 15 FORTRAN statements per minute and using an expansion ratio of 7 this means that 105 machine language instructions per minute can be compiled. Considering the addition of subroutines and input and output, compiling time will normally require two to three times as much 704 time as an equivalent assembly.

FORTRAN Coding Errors

We have had experienced machine language programmers using FORTRAN, and, consequently, have no experience with respect to people who do not know machine language programming. From very limited experience, we have found that there is approximately one error for each 24 FORTRAN statements written. Using our expansion ratio of 7 this means that there is one machine error for each 168 machine language instructions exclusive of subroutines and input and output routines. This represents a pretty accurate programmer, and I am sure that with cross checking and additional FORTRAN experience, this figure will be even better. The reduction in programming errors is a very attractive feature of the FORTRAN system.

An interesting example was a problem that was divided into four parts of approximately equal complexity and each part required approximately 120 FORTRAN statements. There was, however, certain similarity with respect to these various parts. In coding part one of this problem, our programmer made 12 errors. In coding part two, he made just 3 errors. In coding part three, only 1 error was made and part four was coded without a single error. This one example would seem to indicate that experience with FORTRAN will significantly reduce the number of coding errors which can be expected in solving a typical problem. I do not have any figures with respect to the number of errors normally caught with the error detection feature built into phase one of the FORTRAN system. However, our programmers indicated that many errors were caught, and the usual procedure has been to make a preliminary compiling pass through phase one, correct the errors indicated and then complete the compiling operation. We have had just two cases where a program passed through phase one without an error, and, consequently, compiling was completed on the first try. Our experience to date indicates that the error detection features built into the FORTRAN system are excellent and will greatly reduce the number of programming errors which must be found in the debugging stages of a problem.

Debugging

Programming errors can often be caught by re-examining the FORTRAN statements. When errors cannot be found in this manner, our programmers have been debugging the object program and then patching wherever possible since the object program can be debugged almost as easily as a SAP symbolic program. To date there seems to be considerable reluctance to recompile which I would attribute primarily to the additional 704 time required and to the debugging habits of our experienced programmers. As time passes, the tendency will certainly be to recompile more often and for people inexperienced in machine language programming recompiling will be the only satisfactory method of debugging.

I would like to summarize by saying that our experience to date indicates that FORTRAN will 1) be very easy to learn, 2) quicker to code by a factor of 4 or 5, 3) have fewer coding errors approximately 1 per 150 to 200 machine language instructions, 4) will reduce the overall cost of a written instruction by a factor of approximately 3 or 4, 5) will produce very efficient object programs, 6) should handle at least 75 per cent of our problems. From our experience to date, I am convinced that FORTRAN is a very significant step forward in program preparation for the 704 and will prove to be a very satisfactory programming system.