COMMON

1800 Project

FORTRAN Review Committee

Final Report

Committee Membership:

William Delong	Inland Steel Co.
William L. Gee	IBM Corp.
Emily K. Kitcher	A. H. Robbins Co.
Edward J. Kloster	IBM Corp.
Jack J. Owen	Union Camp Corp.
Paul S. Pepin	General Motors Corp.
Peter E. Schilling	Aluminum Corp. of America
LeRoy Sluder Jr.	Long Island Lighting Co.
Philip A. Thompson (Chairman)	Princeton University
John Wolfgang Jr.	NASA

I. INTRODUCTION

At the New York COMMON meeting (August 1971) an informal group began discussing a review of the FORTRAN facilities on the IBM-1800. At subsequent meetings in Los Angeles (December 1971) and Chicago (April 1972) a formal committee met, subcommittee assignments were made, and tentative suggestions were written down for distribution to the rest of the committee. During the month before the Miami Beach meeting (October 1972) a questionnaire based on ideas generated by the Committee was mailed to the entire 1800 Project and the results from the 84 returns were summarized and analyzed to form an appendix to the final report. Also the report of the subcommittee on the present 1800 FORTRAN was made into an appendix. (The material by the ANSI standards subcommittee was published in CAST 51 and the report of the Purdue/ISA process control standards subcommittee in CAST 57.) The actual rough draft of the final report was prepared just prior to the Miami Beach meeting and presented at an 1800 session.

This final report of the Committee consists of a critical analysis of the 1800 FORTRAN facilities from the view-point of process control and data acquisition users --

"sensor-based" applications, if you will. The subroutine libraries are considered, but other parts of the monitor are not. The conclusions for this report include both some suggestions for improvements that could readily be made to the current facilities and the consensus of the FORTRAN Review Committee on what a future sensor-based high-level language (perhaps a Program Product) should be like.

II. THE PRESENT 1800 FORTRAN COMPILER

In December 1969 at the Montreal COMMON meeting the final report of the MPX Review Committee (CAST 26) included a section on the FORTRAN compiler, noting then that the analysis was valid also for TSX and Card System FORTRAN. Most of what was said in that report is still germane; however, the present Committee has done some further work on documenting both what the FORTRAN compiler actually does and what the effects are on sensor-based applications.

Historically, the 1800 FORTRAN compiler was developed directly from that for the 1130, which is, in turn, a subset of USA Standard FORTRAN (X3.9-1966) with some extensions. It is a phased compiler, residing on disk storage in 31 sections. These phases are passed over the source program, which resides in core storage during the entire compilation. Each phase of the compiler is sequentially read into core to perform its transformations on the source strings, moving all strings from one end of core to the other in the process and generating the relevant tables. If any phase detects a source error, the complete source statement string is replaced with the error information. When the compilation is complete, the error messages are listed or the generated object code is placed on disk storage in the proper format for handling by the disk management programs.

The compiler does appear to be quite rapid during compilation and does indeed meet the IBM specifications. However, it is apparent that the compiler achieves little if any optimization of the object code in terms of words of core occupied or of execution time required. In Appendix A is presented a sample program and the disassembly of the object code generated to illustrate some of the methods used by the compiler. Some of the inefficiencies that should be noted include:

1) the handling of negative constants,
2) "calculation" at execution time of array subscripts which are constants,
3) object store for DATA statements,
4) almost total lack of short instructions,
5) minimal optimization from statement to statement.

One of the features of the compiler which deserves special attention is the fact that much of the coding generated is not in-line, but linkages to subroutines. Furthermore, both the subroutines themselves and the linkages to them introduce additional inefficiencies in both code and time, such as overly universal routines for devices or facilities that may not even be present and checking routines that consume much time while providing little actual protection. (In fact, two of the potentially disastrous mistakes that any FORTRAN program might contain and which will cause undetermined errors--out-of-range array indices and the wrong number of parameters in a subroutine call--are not even reliably caught by all the the checking that is provided.)

A final note on the present 1800 compiler should be made about the error messages generated. In Appendix B examples of these messages and comments on them are given.

III. EFFECTS OF PRESENT COMPILER ON SENSOR-BASED APPLICATIONS

Although the 1800 compiler is relatively easy to use and perhaps even well suited to a compile-and-go environment, it provides for sensor-based work only with the inefficiences cited above. As the MPX Review Committee stated in 1969:

> Most 1800's are used in process control and real-time data acquisition and processing. In this environment a program may be compiled and tested a few times and then re-compiled for modification every month or so thereafter. However, during system operation, much of it on a continuous 365-days-a-year basis, these same programs may be executed hundreds or even thousands of times each day. Thus a fast compilation speed means little relative to fast execution times. Even minor program inefficiences result in many hours of wasted execution time.

The views of 1800 Project members were solicited September, 1972; Appendix C gives a summary and analysis of the answers of the 84 installations out of some 200 in the 1800 Project replying. It is seen from these responses that there is much dissatisfaction with the current FORTRAN facilities for the 1800; however, there is far from unanimity on what actual changes would be desirable.

IV. CONCLUSIONS

The Committee, considering the results of the questionnaire and factoring in their own knowledge and experience, decided to suggest eight improvements which could easily be made to the present 1800 FORTRAN facilities:

1) Make C80 (unused variable in DATA statements) and "UNDEFINED VARIABLES" when only in DIMENSION statement into warnings, not fatal errors.

2) Store negative constants as such, rather than as positive constants which have to be subtracted from zero when used at execution time.

3) Calculate addresses for variables with constant subscripts at compile time.

4) Drop the "standard FORTRAN" designation so that "*ONE WORD INTEGERS" and such will not be required.

5) Provide a method of storing INSKEL COMMON on disk for use at compilation source time in a manner analogous to "*SYSTEM SYMBOL TABLE" in the assembler.

6) Count all statements from beginning of program for giving statement numbers with C messages.*

7) Give the first statement in which an undefined variable is used.

8) Store arrays forward in memory.

---

*This improvement was subsequently included in the V3M3 release of MPX.

Besides recommending these improvements for the current FORTRAN facilities, the Review Committee also addressed itself to the question of the general features desired in future high-level languages for sensor-based use both on the 1800 and whatever IBM may replace it with. The main conclusion was that the language should not be limited in usefulness either by its name or by strict adherence to any standard. Specifically, it need not be called FORTRAN and desirable features should not be omitted just because they are beyond the scope of ordinary implementation of FORTRAN of whatever variety. On the other hand there is a real consensus that the language should be based on FORTRAN, rather than on PL/1, ALGOL, BASIC, etc.

As far as standards are concerned both the current ANSI and that now being prepared have many desirable features, some of which might well be included as individual items (but not *in toto* as a "standard") in a future language; however, the cost in core, in compilation time, and perhaps in higher software rental charges would have to be carefully weighed. Finally, the ISA proposed S61 standard for "external-procedure references for use in industrial computer control systems" (also known as "the Purdue extensions to FORTRAN for process control") was viewed with mixed feelings, which might best be summarized as follows: The Committee can see no objection to using the nomenclature, order of calling parameters, etc., of S61 in a future language for sensor-based applications, but this usage should in no way limit efficient utilization of good hardware design or restrict the implementation of features not included in the standard.