Proposed Specifications for FORTRAN II for the 704.

Programming Research Department
International Business Machines Corporation
590 Madison Avenue
New York 22, New York

November 18, 1957

Specifications for FORTRAN II for the 704.

General

FORTRAN has now had some use at a large number of installations and has been used heavily for some time at at least six installations. A large number plan to use the system for the majority of their programming work. Difficulties in using FORTRAN have fallen into two categories (1) Difficulties arising in the initial period of putting the system into use, and (2) Difficulties arising from the properties of the system. Problems in the first category are being solved by on-the-spot expert assistance to those installations experiencing an undue amount of difficulty. The consensus of opinion concerning difficulties in category (2) is that the principal areas needing improvement are (a) facilities for debugging source programs and (b) facilities for creating and using subroutines. FORTRAN II will contain improved facilities in both these areas along the lines described below.

Debugging Facilities

The main improvement contemplated is the incorporation of a general and expandable diagnostic procedure in the translator. The purpose of this procedure is to find and to print a description of every detectable error in a source program. Thus stops in the translation will be eliminated or reduced to a minimum and ambiguity of the reason for a failure to translate will be progressively reduced. Error print-outs will often include values of parameters helpful in locating the trouble.

A variety of routines which are helpful in debugging object programs during execution can be added to the library of FORTRAN and included in an object program during translation by making use of the improved subroutine facilities described below.

Subroutine Facilities

A) New statements in the language

(1) SUBROUTINE, NAME(X, Y, I, ...)

This statement, if it appears, must be the first statement in the program. The SUBROUTINE statement defines the

entire program to be a subroutine, the name of which is NAME (any alpha-numeric characters, not exceeding six, the first of which is alphabetic). The arguments, if any, are to be listed inside a pair of parentheses and separated by commas. An argument name is a non-subscripted variable name (six or fewer alpha-numeric characters, first of which is alphabetic). Any alpha-numeric variable name which occurs in an executable statement in the subroutine may be listed as an argument of the subroutine.

The program which follows the SUBROUTINE statement may be any FORTRAN program containing any FORTRAN statements except a SUBROUTINE statement.

(2) RETURN

This statement will normally occur in a subroutine (a program beginning with a SUBROUTINE statement). RETURN is a flow statement and causes control to be returned to the calling routine. Normal sequencing would be to the statement following the subroutine calling statement.

(3) COMMON A,B,.....

In FORTRAN II, data storage will be moved down in memory so that a gap no longer exists between program and data. The arrangement of such data will be in the normal manner. However, data can be located at the top of memory by listing such data in a COMMON statement. The ordering of storage will be that of the COMMON list, except as it is modified by EQUIVALENCE statements.

(4) CALL NAME (X,Y,I....)

Any statement which consists only of a variable name (six or less alpha-numeric characters, the first of which is alphabetic) and which may be followed by a string of arguments (as defined below) enclosed in parentheses, will be construed to be a call-in of a subroutine having that name, and a calling sequence will be set up to transfer control to that subroutine, and to present it with the arguments, if any, which follow the name.

The order of the arguments is taken from the list, reading from left to right. There must be agreement in number, order, and mode, in the argument list here and the argument list in the SUBROUTINE statement, if the subroutine was

compiled by FORTRAN.

Control from the subroutine continues with the statement following this statement (normal sequencing).

An argument in a subroutine calling statement must be one of the following.

1. fixed point constant
2. floating point constant
3. fixed point variable, with or without subscripts
4. floating point variable, with or without subscripts
5. the name of an array, without subscripts
6. an argument of the following form:

$$,nHx_1xx\text{-}\cdots\text{-}x_n,$$

The interpretation of this is identically the interpretation of a Hollerith field in a FORMAT statement. This is not the name of a variable, but, as with constants, is itself the data under consideration. It will be stored as follows: the 'nH" characters are dropped, the first character of the first word is the first Hollerith character (X_1), the remaining characters, including blanks, are stored as successive characters (six to a word) in successive words, and if the last word is not full, it is filled out with blanks.

(5) END $(I_1, I_2, I_3, I_4, I_5)$ ONE CARD ONLY $I = 0, 1, \text{OR } 2$

This statement is treated as an end-of-file condition thereby permitting the stacking of source programs when compiling.

B) Examples

The following program is an example of a matrix multiplication subroutine (statements 1 - 7) followed by a sample main routine (statements 8 - 11).

```
    SUBROUTINE MATMPY(A,N,M,B,L,C)
    DIMENSION A(10,15), B(15,12), C(10,12)
    DO 5 I = 1,N
    DO 5 J= 1,L
    C(I,J) = 0.0
    DO 5 K = 1,M
5   C(I,J) = C(I,J)+A(I,K)*B(K,J)
    RETURN
7   END (2,2,2,2,2)
8   DIMENSION X(10,15), Y(15,12), Z(10,12), D(10,15),
                E(15,12), F(10,12)
    CALLMATMPY(X,5,10,Y,7,Z)
    CALLMATMPY(D,6,8,E,5,F)
    PRINT 10,Z,F  LIST
10  FORMAT(6E15.6)
11  STOP
```

These are two distinct programs which give rise to two separate and complete compilations, producing a reloctable binary deck for the subroutine and a relocatable binary deck for the main routine.

The following skeleton routine calls for a subroutine which compares Hollerith data.

```
                    ____________________
                    ____________________
Call                FILE(1HA,1HB,1HC)
                    ____________________
                    ____________________
                    END
                    SUBROUTINE FILE(X,Y,Z)
                    ____________________
                    ____________________
                    RECORD(D)
CALL                IF(D-X)1,10,1
1                   IF(D-Y)2,20,2
2                   IF(D-Z)3,30,3
3                   PAUSE
                    RETURN
10 CALL             PROG1(D)
                    RETURN
20 CALL             PROG2(D)
```

```
      RETURN
30 CALL PROG3(D)
      RETURN
      END
```

A subroutine may refer to as many other subroutines as desired and at any depth. In the last example the subroutine FILE refers to other subroutines, namelyhRECORD, PROG1, PROG2, and PROG3. In this example, the routine RECORD is assumed to be a SAP coded program which reads an identifying character from a tape file into a location D in such a way that decisions can be made to identify and process the file of data.

Notice that SAP coded programs can now be combined with FORTRAN programs in a convenient manner.

C) Subroutine Linkage and the BSS Loader

We must distinguish now between subroutines and functions. A function name is the name of a piece of data and can be used as an operand in an arithmetic expression. The calling sequence and the method of constructing programs which compute function values remain unchanged in FORTRAN II. The only change is that the linkage in the calling program will utilize the transfer vector. There will be no change in FORTRAN functions.

All routines produced by FORTRAN II, both master routines and subroutines, will be loadable by the proposed Binary Sumbolic Subroutine Loader. This BSS Loader will enable assembled programs in relocatable binary to retain symbolic references to subroutines at lower levels. As a result a routine and all its subroutines can be compiled or assembled independently. At execution time the relocatable binary decks of the main routine and all subroutines (all starting at relocatable O) may be stacked in the card reader in any order, loaded by the BSS Loader and run.

The BSS Loader will load absolute and relocatable binary cards and accept the usual control cards plus one which will give the symbolic name of the routine and various other information. The first n words of a routine in relocatable binary form which makes symbolic reference to n other routines will contain the BCD symbolic names of the routines referred to.

These n words are called the TRANSFER VECTOR. Reference to a routine B is made from routine A by transferring to that word of routine A which contains the BCD name of routine B. The BSS Loader will replace the symbolic name of B within the routine A by a transfer to the proper entry point in B after B has been assigned a location.

Since FORTRAN will compile routines which make symbolic reference to other routines, each routine in a program may be compiled or recompiled independently from the others. In fact, routines may be compiled which refer to subroutines which have yet to be written.

The construction of a FORTRAN subroutine which uses, say, the sine function and a FORTRAN subroutine called by the statement CALL NAME(A, B, C) can be outlined as follows.

	SINb00	BCD	SINb00	transfer vector
	NAMEb	BCD	NAMEb	HTR 0
entry point		SXD	$, 4	HTR 0
		SXD	$+1, 2	HTR 0
		SXD	$+2, 1	prologue for
		CLA	1, 4	this subroutine
		STA		
		etc.		

		CLA		
function		TSX	SINb00, 4	
		STO		
FORTRAN subroutine		TSX	NAMEb, 4	
		HTR	A	
		HTR	B	
		HTR	C	
		LXD	), 4	

		LXD	$, 4	
RETURN		LXD	$+1, 2	
		LXD	$+2, 1	
		TRA	n+1, 4	where this subroutine has n arguments

The operation of the loader will cause the location of the transfer vector, and the number of entries in it, to be placed in the symbol table. When the program control cards for the routines SIN and NAME are encountered, their names and absolute entry points will also be placed in the symbol table. When a transfer card is encountered, the 'second pass' will begin, and the two BCD words in the example will be replaced by trap transfers to the appropriate locations.