704 FORTRAN II

Error Halt List for BSS Loader, November 28, 1958.

Error Halts in the BSS Loader	Halt (Octal):	Reason for Halt:	Procedure:
	3	Instructions and symbol table of Loader overlap.	Get off machine. Combination of program and transfer vectors too long. Rewrite program.
	20	End of File in the card reader.	Press START to read more cards.
	77453	Instructions and data overlap.	Get off machine. Combination of instructions and data too long. Rewrite program.
	77556	Check sum error.	Press START to accept information.
	77756	More than 20 subroutines are missing.	If missing subroutines are at hand, press START until stop at 77775_8 is reached. Follow instructions (a) for that stop.
	77775	Missing Subroutines.	This stop indicates the TRANSFER CARD has been reached. It is caused by one of two occurances: (a) Loading has been completed, but at least one of the subroutines called for is missing. Location 77453 contains the BCD name of the first missing subroutines, location 77454, the second, etc. If the missing subroutine (s) is immediately available, it may be loaded without starting the entire loading process over again. Place another TRANSFER CARD (9 punch in col 1) at the end of the routine (s), ready in the card reader, and press START. (b) The TRANSFER CARD encountered is really a premature one that simply has not been withdrawn. Be certain that a TRANSFER CARD is the last card at the end of the deck and press START.

FORTRAN SUBPROGRAM TYPES

Following is a table of FORTRAN subprogram types and their general characteristics.

Subprogram Type	Open or Closed	Calling Sequence Type	Single- or Multiple-Valued	Name Type
Built-in Functions	open	(None)	S	1
Function Definition	closed	1	S	1
FORTRAN I Function	closed	1	S	1
FORTRAN II Function	closed	2	S	2
FORTRAN II Subroutine	closed	2	M	3

Subprogram Names

All Types:

a) Only the 36 alpha-numeric (non-special) characters may be used.

b) The first character must be alphabetic.

Type 1:

c) 4 to 7 characters.

d) The first character must be X if and only if the result will be fixed point.

e) The last character must be F.

Type 2:

c) 3 to 6 characters.

d) The first character must be I, J, K, L, M, or N if and only if the result will be fixed point.

e) The last character must not be F if the total number of characters is 4, 5, or 6.

Type 3:

c) 3 to 6 characters.

d) The last character must not be F if the total number of characters is 4, 5, or 6.

Calling Sequences

Type 1:

```
        CLA         ARG1

        LDQ         ARG2        (If ARG2 exists)

        TSX         NAME,4      Upon entry ARG3, if it exists, will be
                                stored at 77775, ARG4 at 77774, etc.
```

Type 2:

```
        TSX         NAME,4      Upon entry the n

        TSX         LARG1       argument locations

        TSX         LARG2       are specified in the

         "            "         address fields

        TSX         LARGN       of the n words immediately following
                                the entry word.  Control returns to the
                                main program at n+1.
```

POUGHKEEPSIE
South Road Laboratory
Department 535
November 26, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library And END Cards

There are currently 20 library functions available for use with the Department 535 FORTRAN system (in addition to the twenty built-in routines). Their names are listed below in one of three groups according to type. The decimal numbers in parenthesis indicate the storage required.

F I Functions:

LOGF	- (42)	- natural logarithm
SINF	- (91)	- sine
COSF		- cosine
EXPF	- (63)	- exponential
SQRTF	- (21)	- square root
ATANF	- (40)	- arctangent
TANHF	- (139)	- hyperbolic tangent
XRANDF	- (35)	- PE RAND, psuedo random number generator
CERF	- (28)	- PK CERF, complementary error function
LGAMF	- (110)	- PK LGAM, natural log of the gamma function

F II Functions: None

F II Subroutines

OFF	- (9)	- PE OVFL, to turn off AC and MQ overflow triggers.
TEST		- PE OVFL, to test the AC and MQ overflow triggers.
SAVE 1	- (129)	- EL SAVE 1 (Q1), core save and restore.
SAVE 2	- (148)	- EL SAVE 2, core and drums save and restore.
AIDE	- (368)	- GL AIDE (D2), integrator.
LAS885	- (229)	- LA S885 (F4), to solve the matrix equation AX = B.
BESJ	- (990)	- NU BES1 (C3), function of the first kind.
BESI		- NU BES1 (C3), modified function of the first kind.
BESY		- NU BES1 (C3), function of the second kind.
BESK		- NU BES1 (C3), modified function of the second kind.
QUOD		- PE QUOD, EAI Dataplotter output routine.

POUGHKEEPSIE
South Road Laboratory
Department 535
October 28, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

PE RAND - Psuedo random number generator.
(Not distributed through SHARE)

FORTRAN Usage: As a FORTRAN I Function.

Example

```
I = XRANDF  ( MOD )
```

where MOD is a fixed point interger variable. A fixed point random number modulo MOD will be generated and stored at the location of I.

Restriction:

See FORTRAN fixed point constants.

L. O. Nippe
Dept. 535

LON/hc

Names of the above programs must be used in FORTRAN statements exactly as listed followed by the necessary arguments. Function names should appear in the right hand side of arithmetic statements, and subroutine names should appear in CALL statements.

Additional storage used by the input-output-convert programs in your binary deck is listed below.

(DBC)	462
(CSH)	137
(TSH)	21
(BDC) - (FIL)	435
(SCH)	90
(SPH)	158
(STH)	12
(RTN) - (LEV)	80
EXP(1	34
EXP(2	40
EXP(3	113

As you know, Sense Switch 5 controls the punching of library programs during compilation. With the increasing number of compilations per day, this on-line punching is beginning to represent a significant amount of machine time. Furthermore, most library programs punched are redundant because the programmer already has them in his binary deck from a previous compilation. In order to eliminate this problem, copies of all FORTRAN library programs have been placed in a card file drawer in the machine room. In the future, it will be appreciated if all FORTRAN programs would be compiled with the following END card:

END (X,X,2,X,0) where each X is 0, 1, or 2

After compilation, the programmer should refer to the last category of the second file of output for a list of the necessary library programs.

L. O. Nippe
Department 535

LON/c

POUGHKEEPSIE
South Road Laboratory
Department 535
November 14, 1958

Memorandum to: All 704 Users

Subject: FORTRAN Library Addition

Program Description:

PK CERF - Complimentary error function (1 - probability integral)
Not distributed through SHARE

FORTRAN Usage: As a FORTRAN I Function

Example:

C = CERF(X)

Where X is a positive floating point argument. The normalized floating point value of the function will be stored at location of C.

H. S. Long
L. O. Nippe
Department 535

HSL/LON/c

POUGHKEEPSIE
South Road Laboratory
Dept. 535
October 28, 1958

Memorandum to: All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

PK LGAM - Natural logarithm of the Gamma Function.
(Not distributed through SHARE)

FORTRAN Usage: As a FORTRAN I Function.

Example

F = LGAMF (X)

where X is a positive floating point argument. The normalized floating point value of the function will be stored at the location of F.

Restriction:

$X > O$. See PK LGAM write-up.

L. O. Nippe
Department 535

LON/hc

POUGHKEEPSIE
South Road Laboratory
Department 535
November 25, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

PE OVFL - Reset or test overflow triggers.

FORTRAN Usage: As a FORTRAN II Subroutine

CALL OFF - Turn off the AC and MQ overflow triggers and lights.

CALL TEST - Test the status of the AC and MQ overflow triggers. The SR address field will contain 2123_8 at stop for AC overflow and 4450_8 at stop for MQ overflow.

In order to determine the program location of the overflow, he should dump memory immediately following the overflow stop and use the contents of index register 4 to trace back to the main program location.

L. O. Nippe
Department 535

LON/c

POUGHKEEPSIE
South Road Laboratory
Department 535
October 28, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

EL SAVE 1 - Save and restore cores, and
EL SAVE 2 - Save and restore cores and drums.
(SHARE classification Q1)

FORTRAN Usage: As FORTRAN II Subroutine.

Example

```
        IF  ( SENSE SWITCH 6 )  101, 100

  100   CALL SAVE 1

  101   (next statement of sequence)
```

The above will save core storage only. SAVE 2 should be used when it is desired to save both core storage and drums.

Restrictions:

Sense switch 6 must be used with these two subroutines. See SHARE write-up for details.

L. O. Nippe
Department 535

LON/hc

POUGHKEEPSIE
South Road Laboratory
Department 535
October 28, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

LA S885 - to solve the matrix equation
AX = B for X and to evaluate the determinant A.
(SHARE classification F4)

FORTRAN Usage: As a FORTRAN II Subroutine.

Example

CALL LA S885 (M, N, TEMP (N, M), I, DETA)

where M is the number of columns and N is the number of rows of the combined AB matrix, Temp (N, M) is the first element of the AB matrix stored row-wise and backwards in memory, I = O means solve the equation only, I = 1 means solve the equation and evaluate the determinant A, etc., and DETA is the variable name assigned to the value of the determinant of A or its inverse.

Restrictions:

See SHARE write-up.

L. O. Nippe
Department 535

LON/hc

POUGHKEEPSIE
South Road Laboratory
Department 535
October 28, 1958

Memorandum for All 704 Users

Subject: FORTRAN Program Library Addition

Program Description:

NU BES1 - Bessel functions for real argument and order.
(SHARE classification - C3)

FORTRAN Usage: As a FORTRAN II Subroutine.

Examples

```
CALL   BESJ ( ARG, F, N, STORE (m))
CALL   BESI ( ARG, F, N, STORE (m))
CALL   BESY ( ARG, F, N, STORE (m))
CALL   BESK ( ARG, F, N, STORE (m))
```

where ARG is the name of the floating variable argument, F and N are the floating fraction and fixed integer parts of the Bessel function order, and STORE (m) is the last (low order) word of an m word array. Upon return from the subroutine, using the function of the first kind as an example, J_0 will be found at STORE (m), J_1 at STORE (m-1), etc. through J_N at STORE (m-N).

Restrictions:

The respective AC overflow error stops are located near the end of the program at BESJ+3, BESI+3, BESY+3, and BESK+3.

This FORTRAN version will cause a program stop at BESJ+ 61_8

October 28, 1958

if any entry is attempted with $\text{Arg} > 50$.

See the NU BES1 write-up for the minimum size of the STORE array.

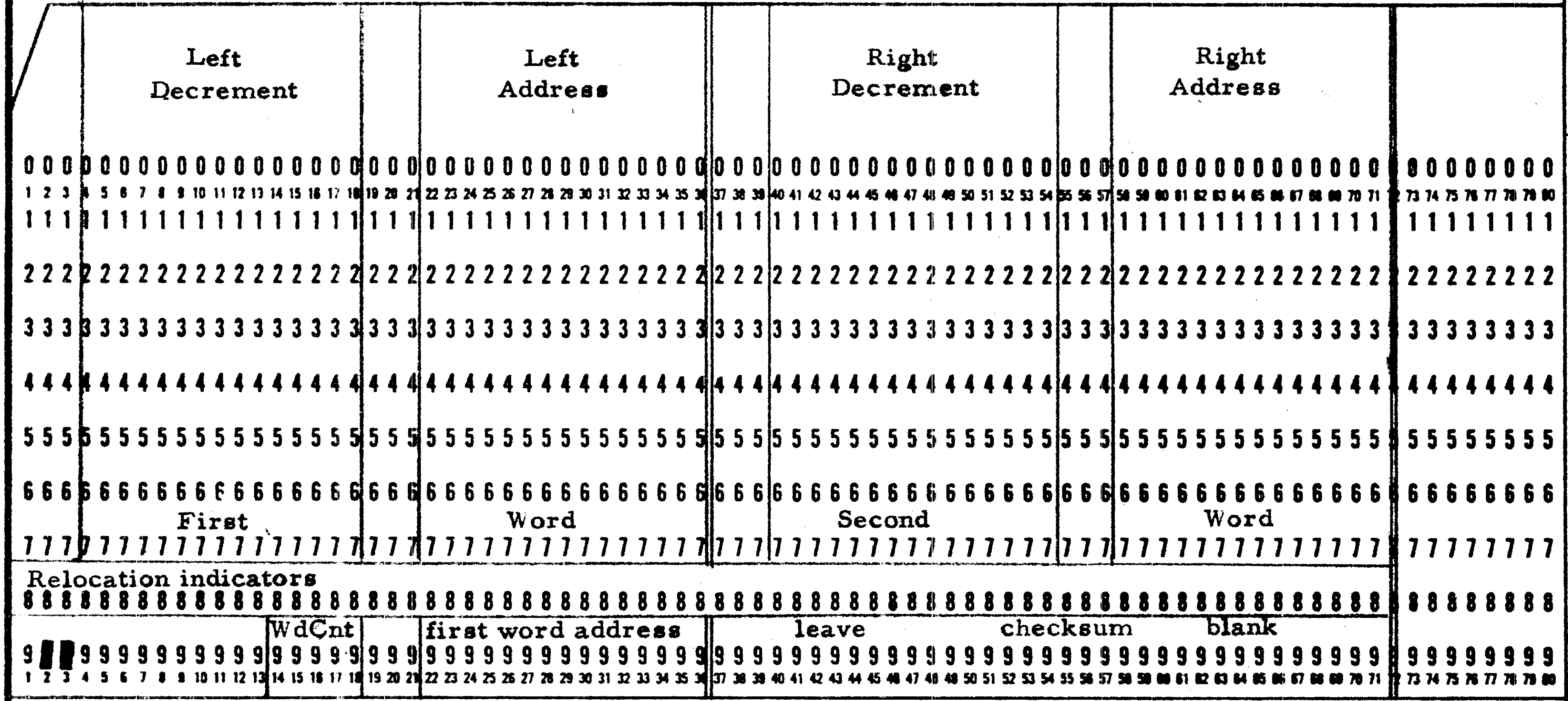
L. O. Nippe
Department 535

LON/hc

MASTER IBM CARD LAYOUT

DATE

FORTRAN RELOCATABLE BINARY INSTRUCTION OR DATA CARD



Row	Col.	
9	2, 3	Must be punched
	14-18	Card word count (1 - 24_8)
	22-36	Storage location relative to zero of first word (7-left)
9	Right	Blank
8	1-72	Relocation digits for 7Ld, 7La, 7Rd, 7Ra, 6Ld, etc. 0- do not relocate, 10- relocate directly, 11- relocate inversely
7	Left	First word
7	Right	Second word
etc.		

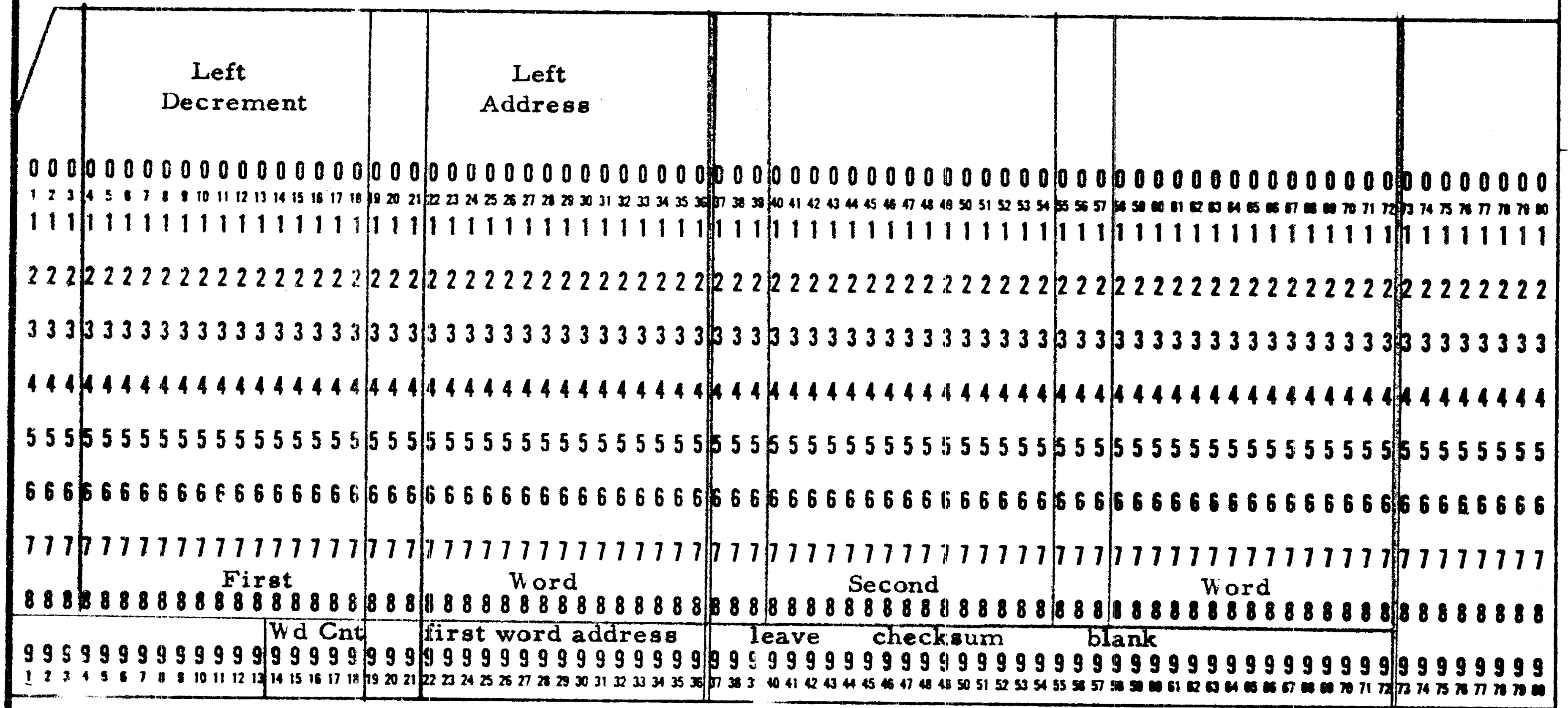
REPORT NO.

PAGE

MASTER IBM CARD LAYOUT

DATE

FORTRAN ABSOLUTE BINARY INSTRUCTION OR DATA CARD



Row	Col.	
9	3	Must be punched
	14-18	Card word count (1-26_8)
	22-36	Absolute storage location of first word (8- left)
9	Right	Blank
8	Left	First word
8	Right	Second word
etc.		

REPORT NO.

PAGE

FORTRAN MODIFICATIONS/BOOLEAN EXPRESSIONS

I. Description of Boolean Statement

FORTRAN will now accept Boolean statements. A Boolean statement is in the same form as an arithmetic statement, with the arithmetic operators +, *, and unary - taken to be the logical operators or, and, and complement. The operator * has greater binding strength than the operator +.

Because - is a unary operator it is part of the expression or symbol to which it applies. Therefore it must be bound to its expression or symbol by parentheses, with one exception.

If E is a variable or function name the complement can be written as - E when it is not part of a larger expression in the same statement. C = -E is correct, but C = - E + D is not correct. When E is part of a larger expression it must be written as (- E). Thus, the latter expression must be written as C = (-E) + D.

Example 1.*

$D = A \cap \overline{B \cup C}$ would be written in FORTRAN as
D = A * (- (B + C))

The inner pair of parentheses is required to indicate the scope of complementation. The outer pair of parentheses is required because the expression, - (B+C) is a part of a larger expression.

*The use of a Boolean statement in a complete FORTRAN problem is illustrated in FORTRAN MODIFICATIONS/ MACHINE LANGUAGE write-up, Appendix I.

Example 2.

$D = \overline{\text{IMPF}\ (\bar{B},\ \bar{C})}$ is written as
$D = -\text{IMPF}\ (-B,\ -C)$

No additional parentheses are required here because the function name as well as the argument names are not parts of a larger expression.

II. Use of Boolean statements.

1. A Boolean statement in FORTRAN requires a "B" in card column 1.

2. All variables in a Boolean statement must have FORTRAN floating point names.

3. Variable names can be subscripted in the normal FORTRAN manner.

4. All Boolean operations are performed upon the full 36 bit logical word.

5. Table size limitations, such as for Lambda and Alpha tables, apply in exactly the same way as for arithmetic statements.

FORTRAN MODIFICATIONS/FUNCTION AND SUBROUTINE NAMES AS ARGUMENTS

Fortran will accept Function and Subroutine names as arguments in other subroutines. Thus:

```
Example 1.     SUBROUTINE BOB (DUMMY, Y)
               A = DUMMY F(Y)
```

will permit the Dummy function to be different depending upon the arguments specified in the call statements. Thus:

```
Example 2.     CALL BOB (SIN, S)

               CALL BOB (COS, S)
```

will result in placing the Sin(S) and the Cos(S) in cell A respectively.

In order to distinguish between the data name and the function name in an argument list; an F card is required to list these subroutine names used as arguments. Thus for example 2, the F card is:

```
col. 1     7

  F        SIN, COS
```

Note: If a subroutine name requires a terminal 'F' when occuring within an arithmetic statement then the dummy name must also have the terminal 'F'. This terminal 'F' must be dropped from the name whenever it occurs in an F card list or as the argument of a Call or Subroutine statement. The F card must be in the program containing the CALL statement and may appear anywhere in the deck.

FORTRAN MODIFICATIONS/ADDITIONAL FORMAT FEATURES

Format facilities have been expanded, in the new Input-Output Hollerith Routine (IOH), as follows:

1. The control character X; written: nX, where $0 \leq n \leq 120$.

a) IOH, on input, will interpret this to mean that the following n characters of input should be skipped.

```
e.g.:     READ 1, K, A
    1     FORMAT (12X, I2, 8X, F8.3)
```

This will cause cols. 1-12 to be skipped, cols.13-14 (K) to be read, cols. 15-22 to be skipped, and cols. 23-30 (A) to be read.

b) IOH, on output, will interpret this to mean that the following n characters of output should be blanks.

```
e.g.:     PRINT 1, K, A
    (1 as in the above example)
```

This will print K, preceded by 12 blanks, and A, preceded by 8 blanks.

2. The control character O; written: nOw, where $0 \leq n*w \leq 120$.

a) IOH, on input, will interpret this to mean that the following n successive fields of w characters each are to be converted from octal to binary. If w is greater than 12, only the 12 rightmost characters will be significant. If w is less than 12, the number will be right-adjusted and filled out with zeros. Leading blanks will be treated as zeros.

```
e.g.:     READ 1, A, B, C
    1     FORMAT (2O15, O9)
```

Where A is 27, punched in cols. 14-15, preceded by 13 blanks; B is 777777777777, preceded by 3 blanks; and C is - 12345, preceded by 3 blanks; then, in memory A will be: 000000000027$_8$, B will be 777777777777$_8$, and C: 400000012345$_8$.

FORTRAN MODIFICATIONS/ADDITIONAL FORMAT FEATURES

b) IOH, on output, will interpret this to mean that the following n successive fields of w characters each of output are to be the result of conversion from binary to octal. If w exceeds 12, the excess will be blanks, and the result will be right-adjusted. If w is less than 12, only the w rightmost digits will be significant. Leading zeros will be converted to blanks, and the number will be signed if negative. However, 12 or more significant digits will be unsigned.

```
e.g.:      PRINT 1, A, B, C
       (1 as in the above example with same data)
```

This would print 27 preceded by 13 blanks, 777777777777 preceded by 3 blanks, and -12345 preceded by 3 blanks.

3. The control character A; written:nAw, where $0 \leq n*w \leq 120$.

a) IOH, on input, will interpret this to mean that the following n successive fields of w characters each are to be stored in memory as BCD information. If w is greater than 6, only the 6 rightmost characters will be significant. If w is less than 6, the characters will be left-adjusted, and the word filled out with blanks.

```
e.g.:      READ 1, (RECORD (I), I = 1, 13)
    1      FORMAT (11A6, A4, O2)
```

This would result in the first 70 characters of the card being stored in the first 12 words associated with the array "RECORD". The last 2 characters of the 12th word would be blanks. Characters 71 and 72 would be converted from octal to binary and stored right-adjusted in the 13th word. An appropriate Dimension entry must have been made for "RECORD".

b) IOH, on output, will interpret this to mean that the following n successive fields of w characters each of output are to be the result of transmission from memory without conversion. If w exceeds 6, only 6 characters of output will be transmitted, preceded by w-6 blanks. If w is less than 6, the w leftmost characters of the word will be transmitted.

FORTRAN MODIFICATIONS/ADDITIONAL FORMAT FEATURES

4. Format Statements read in at object time.

FORTRAN will accept a variable Format address. This provides the facility of describing a List at object time.

```
e.g.:      DIMENSION F MT (12)
           FORMAT (12A6)
           READ 1, (FMT (I), I =1, 12)
           READ FMT, A, B, (C (I), I = 1,5)
```

Thus A, B, and C would be converted and stored according to the Format Specification read into the array, FMT, at object time.

IBM Applied Programming
January 1960

FORTRAN MODIFICATIONS/MACHINE LANGUAGE INSTRUCTIONS IN FORTRAN PROGRAMS

A 704 FORTRAN program may now include machine-language code among its instructions. This code, which is in Share Mnemonics, must be considered part of a FORTRAN program, and is therefore subject to restrictions that do not apply when the code only is prepared for an assembly. These restrictions primarily are (1) a slight limitation on instructions available, (2) certain rules for specifying subscript values, and (3) maintenance of some instruction sequences with respect to transfer instructions.

I. These codes may be divided into machine instructions and pseudo-ops.

A. Format for a FORTRAN machine instruction.

Col.	1	2-5	7----------72
	S	Statement No.	Instruction

Statement Number: A decimal integer.

Instruction:

Operation - (a) Machine language code (listed in Appendix II)
(b) Pseudo-op (listed in B below.)

Address - (a) A statement number, preceded by an asterisk.
(b) A positive decimal integer.
(c) A variable name, subscripted or unsubscripted.

Tag - A symbolic tag name. This will always be a fixed point variable, with no coefficient or addend. It must be enclosed in parentheses. This field may be used for shift, read, write and non-indexable instructions. (With caution, it may be used with certain other indexable instructions. See IV, A below.)

Decrement- A signed or unsigned decimal integer. If negative, the complement will be used.

Notice that there can be no reference to an absolute storage location in the address field or to an absolute index register in the tag field. As in all FORTRAN instructions, blanks are ignored. The fields of an operation must be separated by commas. When a blank field is followed by a non-blank field, the blank field must have a comma after it; a zero may occupy a blank field.

Examples of FORTRAN machine instructions:

```
TRA*25                      TXL*25,,4           STA*13
RTB6,(I)                    RQL 27              LBT
PXD,(I)                     LXA INDEX,(I)       ORS A(I)
PXD0,(I)                    TXH*25,(I),1        ETT
CLAA(I+3,2*J,5*K-1)
```

B. There are three pseudo-ops that may be used in a FORTRAN program. These define program variables.

Format for pseudo-operation.

Cols.	1	2-5	7-----------72
	S	FORTRAN variable name	Constant

Instructions(Pseudo-ops):

DEC: Any signed or unsigned fixed or floating-point number conforming to the FORTRAN specifications of constants. If the number is fixed point, it is stored in the decrement field of the word.

OCT: A string of from 1-12 octal digits. This octal number is unsigned; the sign must be made part of the number. These octal digits are right-adjusted in the storage word.

ALF: A string of 6 alphanumeric characters. In this case only, a strict card format must be adhered to. This is because blanks may be taken as alphanumeric characters. The Operation must be in columns 7-9; the six alphanumeric characters will be taken from columns 13-18.

Examples of FORTRAN Pseudo-ops.

```
IZ          DEC 23              |  MR     OCT    4637021
NAME        ALFbbbROBERT        |  PI     DEC    3.1416
A14         DEC 68.924 E + 8    |
```

The FORTRAN pseudo-op instructions must precede any executable statement of the problem, including the Arithmetic Statement functions. It should be noted from the above description that it is possible to associate a floating point number with a fixed point variable name and vice-versa. It is advisable not to do this as it can easily lead to errors; for example, if the variable is used in an arithmetic statement.

II Coding Rules.

A. Relative Constants.

A relative constant is a subscript symbol not under control of a DO. As pointed out in the FORTRAN Reference Manual, a relative constant must receive explicit definition. Within the context of a single problem this definition is provided by two means: (a) appearance on the left side of an arithmetic statement and (b) appearance in an input list.

In using machine language it is necessary to provide instructions corresponding to (a) and (b) whether the relative constant itself appears in a machine instruction (e.g. CLA A(I)) or a normal FORTRAN statement (e.g. B = A(I)).

The following machine instructions serve to define a relative constant:
STO, SXD, STD, SLQ, STQ, ORS, ANS, STZ, SLW, CPY, CAD.

Examples:

```
(1)  .          (2)   .           (3)   .
     .                .                 .
     .                .             STO J
STQ I           CPY I               DO 10 I = 1,5
     .                .             LDQ A(I,J)
     .                .          10 SLQ B(I,J)
     .                .
SLW J           TX1*5,(J),1
     .          SXD J,(J)
     .                .
     .                .
CLA A(I,J)      CLA A(I,J)
```

```
(4)  .          (5)   .
     .                .
     .                .
LXD K, (I)      TIX*8,(J),2
SXD I, (I)            .
     .                .
     .                .
     .          CLA A(J)
CLA A(3*I)
```

These illustrations show:

a. That all relative constants must be explicitly defined. In example (3) only "J" in the subscript combinations (I,J) is a relative constant.

b. That this definition must occur after any change of value of the Tag quantity, with one exception.

c. That this exception is shown in example (5). Notice how the latter differs from examples (2) and (4). It does not have the defining machine instruction after the change of value of the Tag quantity.

> Rule: Where the relative constant subscript expression is one-dimensional and does not have a coefficient, the machine instruction providing explicit definition is not required.
>
> In example (2) the subscript expression is not one-dimensional and in example (4) it has a coefficient.

B. The instruction "TSX" may not be used. It is suggested that the CALL statement be used when a transfer to a subroutine is required. In addition, of course, such transfers may be obtained by function references in FORTRAN arithmetic statements.

C. In the analysis FORTRAN makes of a program for optimization purposes, it expects to find certain instruction sequences at points where transfers occur. The machine language coding must conform to these sequences.

1. The CPY and CAD instructions must be used in one of the following ways:

   a. If a copy skip is anticipated

   ```
   CPY
   TRA
   TRA
   TRA
   ```

   b. If a skip is not anticipated, then the copy instruction can be followed by any instruction except a TRA.

2. A skip type test instruction must be followed either by two unconditional transfers or by none with the exception of the CAS which must be followed by three or by none.

D. Modification of Transfers.

1. In order to assign index registers optimally, the FORTRAN exeartive system must know all paths taken by any transfers. Therefore, transfer instruction addresses must not be modified by other instructions in the program.

2. For a corresponding reason the machine language program must not modify the operation field of an instruction to change a non-transfer instruction into a transfer instruction.

E. Instruction Format Requirements.

1. All transfer instructions must have addresses which are statement numbers in the program.

2. Observe that MSE and PSE are missing from the instruction list (Appendix II). The equivalent SHARE code should be used instead.

3. Relative addressing is not permitted (e.g. CLA A+2). Addends, of course, are permissible in subscript expressions (e.g. A(I+3) or A(2)).

4. To a limited extent, the programmer may insert constants into unused fields of instructions at source program time. The Appendix II chart indicates when this may be done.

III Optimizing Programs.

Certain practices, if observed, can result in more efficient FORTRAN programs with respect to indexing instructions.

A. The DED instruction, Another pseudo-operation--of an unique type--is available to programmers. The use of this pseudo-operation can reduce the number of LXD and SXD instructions FORTRAN must use. It is an instruction which informs the compiler that a symbolic tag will not be used subsequently in the program (without having its value reset first) and that it is, therefore, dead.

Example:

```
S   DED, (I)
```

The DED instruction is non-executable and must not have a statement number.

B. Frequency Statements. Frequency statements may be applied to machine language transfers as well as to ordinary FORTRAN statements.

1. Conditional transfers. Each such transfer instruction, regardless of how many appear in sequence, must have a different frequency statement entry. This means, of course, that each must have a statement number.

2. Skip type transfers. An external statement number is associated with the test instruction and three frequency estimates are listed if it is a CAS and two if it is any other. These frequency estimates apply to the following instructions whether they be FORTRAN statements or machine operations.

C. Paths of flow. As in FORTRAN programs generally, all executable instructions must have a path of flow leading to them.

IV Additional Information.

A. Where the indexable instruction addresses an array the symbolic tag may be used. Of course, this will rarely be necessary because merely subscripting the array refers to the proper element of the array. The danger in using the symbolic tag here is that it does not take into account the relative address which FORTRAN automatically supplies when a subscripted variable is referenced. To illustrate:

CLA A, (I) and CLA A(I)

are not equivalent. In the former case, the indexing is done from location A; in the latter, from location A + 1.

B. Under the following circumstances, the contents of the Accumulator and MQ will be destroyed.

a. Execution of a FORTRAN or machine instruction transfers out of the range of a DO.

b. Execution of one of the following machine instructions if the symbol in the address is an index of any subscript in the program not under control of a DO.

STO, SXD, STD, SLQ, STQ, ORS, ANS, STZ,
SLW, CPY, CAD

c. Execution of the following FORTRAN statements:

Arithmetic
Arithmetic IF
ASSIGN
Boolean
CALL
DO
All Input/Output statements
RETURN

C. Storage allocation can be done only by means of Common, Dimension, and Equivalence statements.

Appendix I.

Example of program using machine language and Boolean statement.

```
C              Program to generate truth table of Boolean expression
C              in four variables V1, V2, V3, V4.

               DIMENSION V(4)

               EQUIVALENCE (V1, V(1)), (V2, V(2)), (V3, V(3)), (V4, V(4))

               FREQUENCY 8(4,1)

S     CNT      DEC 4

S              STZ  IVARY

               DO10 I = 1, 16

S              LDQ IVARY

S              LLS 13

S              LXD CNT, (J)

S     5        PXD

S              LLS 1

S              ALS 18

S              STO V(J)

S     8        TIX*5, (J), 1

B              TVALUE = V1*(-V2)+(-((V3+V1)(-(V3+V2))))+V4

               PRINT 2, (V1, V2, V3, V4, TVALUE)

      10       IVARY = IVARY 1

               STOP 77777

      2        FORMAT (6(5XI1))
```

It should be noted that for Boolean output it is possible to use a fixed-point Format description for a floating-point variable name. If a full word fixed-point, octal or binary printout is desired, a FORTRAN II subroutine must be written.

APPENDIX 2

LIST OF FORTRAN MACHINE LANGUAGE INSTRUCTIONS

	VARIABLE FIELD				
	ADDRESS			TAG	DEC
	EFN	DAT	NUM		
ACL	NO	YES	NO	YES	NO
ADD	YES	YES	NO	DAT	NO
ADM	YES	YES	NO	DAT	NO
ALS	NO	NO	YES	YES	NO
ANA	NO	YES	NO	YES	NO
ANS	NO	YES	NO	YES	NO
ARS	NO	NO	YES	YES	NO
BST	NO	NO	YES	YES	NO
CAD	YES	YES	NO	YES	NO
CAL	YES	YES	NO	DAT	NO
CAS	NO	YES	NO	YES	NO
CFF	NO	NO	NO	NO	NO
CHS	NO	NO	NO	NO	NO
CLA	YES	YES	NO	DAT	NO
CLM	NO	NO	NO	NO	NO
CLS	YES	YES	NO	DAT	NO
COM	NO	NO	NO	NO	NO
CPY	YES	YES	NO	YES	NO
DCT	NO	NO	NO	NO	NO
DVH	NO	YES	NO	YES	NO
DVP	NO	YES	NO	YES	NO
EFM	NO	NO	NO	NO	NO
ETM	NO	NO	NO	NO	NO
ETT	NO	NO	NO	NO	NO
FAD	NO	YES	NO	YES	NO
FDH	NO	YES	NO	YES	NO
FDP	NO	YES	NO	YES	NO
FMP	NO	YES	NO	YES	NO
FSB	NO	YES	NO	YES	NO
HPR	YES	YES	YES	YES	NO
HTR	REQ	NO	NO	NO	NO
IOD	NO	NO	NO	NO	NO
LBT	NO	NO	NO	NO	NO
LDA	NO	YES	NO	YES	NO
LDQ	YES	YES	NO	DAT	NO

	VARIABLE FIELD				
	ADDRESS			TAG	DEC
	EFN	DAT	NUM		
LFM	NO	NO	NO	NO	NO
LGL	NO	NO	YES	YES	NO
LLS	NO	NO	YES	YES	NO
LRS	NO	NO	YES	YES	NO
LTM	NO	NO	NO	NO	NO
LXA	YES	YES	NO	YES	NO
LXD	YES	YES	NO	YES	NO
MPR	NO	YES	NO	YES	NO
MPY	NO	YES	NO	YES	NO
NOP	YES	YES	YES	YES	NO
ORA	NO	YES	NO	YES	NO
ORS	NO	YES	NO	YES	NO
PAX	YES	YES	YES	YES	NO
PBT	NO	NO	NO	NO	NO
PDX	YES	YES	YES	YES	NO
PXD	YES	YES	YES	YES	NO
RCD	NO	NO	NO	NO	NO
RDR	NO	NO	YES	YES	NO
RDS	NO	NO	YES	YES	NO
REW	NO	NO	YES	YES	NO
RND	NO	NO	NO	NO	NO
RPR	NO	NO	NO	NO	NO
RQL	NO	NO	YES	YES	NO
RTB	NO	NO	YES	YES	NO
RTD	NO	NO	YES	YES	NO
RTT	NO	NO	NO	NO	NO
SBM	YES	YES	NO	DAT	NO
SLF	NO	NO	NO	NO	NO
SLN	NO	NO	YES	YES	NO
SLQ	YES	YES	NO	DAT	NO
SLT	NO	NO	YES	YES	NO
SLW	NO	YES	NO	YES	NO
SPR	NO	NO	YES	YES	NO
SPT	NO	NO	NO	NO	NO
SPU	NO	NO	YES	YES	NO
SSM	NO	NO	NO	NO	NO
SSP	NO	NO	NO	NO	NO
STA	YES	YES	NO	DAT	NO
STD	YES	YES	NO	DAT	NO
STO	NO	YES	NO	YES	NO
STP	YES	YES	NO	DAT	NO
STQ	NO	YES	NO	YES	NO
STZ	NO	YES	NO	YES	NO

	VARIABLE FIELD				
	ADDRESS			TAG	DEC
	EFN	DAT	NUM		
SUB	YES	YES	NO	DAT	NO
SWT	NO	NO	YES	YES	NO
SXD	YES	YES	NO	YES	NO
TIX	REQ	NO	NO	YES	YES
TLQ	REQ	NO	NO	NO	NO
TMI	REQ	NO	NO	NO	NO
TNO	REQ	NO	NO	NO	NO
TNX	REQ	NO	NO	YES	YES
TNZ	REQ	NO	NO	NO	NO
TOV	REQ	NO	NO	NO	NO
TPL	REQ	NO	NO	NO	NO
TQO	REQ	NO	NO	NO	NO
TQP	REQ	NO	NO	NO	NO
TRA	REQ	NO	NO	NO	NO
TTR	REQ	NO	NO	NO	NO
TXH	REQ	NO	NO	YES	YES
TXI	REQ	NO	NO	YES	YES
TXL	REQ	NO	NO	YES	YES
TZE	REQ	NO	NO	NO	NO
UFA	NO	YES	NO	YES	NO
UFM	NO	YES	NO	YES	NO
UFS	NO	YES	NO	YES	NO
WDR	NO	NO	YES	YES	NO
WEF	NO	NO	YES	YES	NO
WPR	NO	NO	NO	NO	NO
WPU	NO	NO	NO	NO	NO
WRS	NO	NO	YES	YES	NO
WTB	NO	NO	YES	YES	NO
WTD	NO	NO	YES	YES	NO
WTS	NO	NO	YES	YES	NO
WTV	NO	NO	NO	NO	NO
ALF	PSEUDOOPERATION				
DEC	PSEUDOOPERATION				
OCT	PSEUDOOPERATION				
DED	TAG FIELD ONLY				

FORTRAN PROGRAM CARDS

In preparing a SAP coded program for use with a FORTRAN program the relocatable binary deck must be preceded by a "program card". The simplest way to prepare this card is to have the assembler punch it for you during the assembly process. This may be done with a program arranged like the following example.

```
                REM  (program identification)
                FUL
 9L             FOR  0,,END-1   (program card word count)
 9R             PZE             (leave checksum blank)
 8L             PZE  (last program location + 1),,(number of words in transfer list)
 8R             PZE  -N-205     (where N is the number of common used)
 7L             BCD  1NAME1     (FORTRAN name assigned to first entry point)
 7R             PZE  ENTRY1     (SAP symbolic address of first entry point)
                ...
                BCD  1NAMEN     (FORTRAN name assigned to last entry point)
 END            PZE  ENTRYN     (SAP symbolic address of last entry point)
                REL
                ORG  0
COMMON          SYN  -N-205     (where N is the number of common used)
                    }
                    } SAP Program
                    }
                END  0
```

After assembly the transfer card should be removed from the back of the relocatable binary deck. This deck may now be loaded as part of your FORTRAN object deck. It may also be written on the FORTRAN library tape if the program card check sum is first corrected.

Acknowledgement is given to W. R. Couch and associates of the Endicott Computing Laboratory (EL) who originated the above idea and gave it to us.

L. O. Nippe
Department 535

LON/hc
October 19, 1959