

9/12 - cc: J. Cocke

FYI - I. Ziller

Description of Source Language Additions to the FORTRAN II System.

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

Introduction

The major intention of F3 is to provide the programmer with the ability to manipulate data stored in the machine in a variety of ways that were not hitherto conveniently available to him within the existing FORTRAN framework.

The first, and perhaps most useful addition, is the ability to use the basic machine language instructions. These are written in a Sap-like form but are to be considered as new FORTRAN statements subject to restrictions dictated both by the real machine (704) and the synthetic machine (FORTRAN).

→ It should be noted here that the FORTRAN-Sap machine can be assumed to have an unlimited number of index registers. It will be assumed that the user has a knowledge of the 704, the Sap assembly program and FORTRAN.

To be addressed simultaneously
 ↓
 The second addition is a Boolean Algebraic statement. This provides the programmer the facility of algebraic expressions which will treat a word as a collection of logical bits and interpret certain arithmetic operations as union, intersection and negation.

The third and last addition is an assortment of devices that will allow alphabetic information to be handled in a variety of new ways.

NOTE: Incompatibility

In F1 and F2, the appearance of a non-subscripted array name in an I/O list produced a program which would read or write the entire array in reverse order. In F3, this facility is available for binary lists only. In decimal lists, this notation has reference to alphanumeric fields larger than six characters.

Boolean Expressions

In a FORTRAN Boolean statement the algebraic operators \wedge , \vee , and unary \neg are taken to be the logical operators **or**, **and**, and **complement**. The order of binding between symbols is \neg , \wedge , \vee (strongest to weakest). With respect to complementation the following rules must be observed:

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

Example 1. $D = A \wedge \overline{B \vee C}$ would be written in FORTRAN as
 $D = A * \neg (B + C)$

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

Example 2. $D = \overline{\text{IMPF}(\bar{B}, \bar{C})}$ is written as
 $D = \neg \text{IMPF}(\neg B, \neg C)$

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

A Boolean statement in FORTRAN requires a 'B' in card column 1 and all variables must have FORTRAN floating-point names. The variable names can be subscripted in the normal FORTRAN manner. All Boolean operations are performed upon the full 36 bit logical word.

Features for Handling Alphanumeric Information

1. Function and Subroutine names can be used as arguments in other sub-routines. Thus:

Example 1. SUBROUTINE BOB (DUMMY, Y)

 A = DUMMYF(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)

and

 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 occurring 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.

Format Additions

1. The ability to move alphanumeric characters about is now provided by the letter A in the Format statement. It is used in the same manner as I or E. The characters are packed to the left and incomplete words are filled out with blanks.

Example 1 READ 1, (RECORD(I), I=1, 8)

 1 FORMAT (7A6, A2)

Example 2 READ 2, (RECORD)

 2 FORMAT (A44)

The above statements would result in the first 44 characters of the card being placed in the first 8 words associated with the array 'Record'. The last 4 characters of the 8th word would be blanks. An appropriate dimension entry must be made for 'Record'.

2. The character 'B' has been added as shorthand for $nHb_1 \dots b_n$

Example READ 1

 PRINT 1

 1 FORMAT (72B)

This will print the first 72 characters read in from a card.

NOTE: For Format table limitation size, add to the character count of a Format statement, for each B field, a number 'm', which is the smallest multiple of 6 not less than n, in nB. Example, the element 72B counts as 3+ 72, the element 13B counts as 3+ 13+ 5.

Machine Instructions

This description will be divided into two parts. Part 1 will describe the regular machine and special ops. Part 2 will furnish restrictions and suggestions for their use.

Part 1 -

A. Format

col. 1 2 - 5 7 72

S statement # Instruction

B. Elements of an Instruction: All elements after the op are separated by commas.

1. Operation: This is a subset of the standard 3 letter SAP code for machine instructions.
2. Address: This can be one of the following:
 - a. Statement number (prefixed by an asterisk)
 - b. Positive decimal integer
 - c. Standard subscripted or unsubscripted FORTRAN variable name
3. Tag: This is a one dimension subscript without coefficient or addend enclosed in parentheses and occurs only in shift, read and write instructions as well as all non-indexable instructions.
4. Decrement: This is a positive decimal integer.

Examples of some F3 machine instructions are:

```
TRA*25      ; TXL*25, .4      ; TXH*25, (I), 1
RTB 6, (I)  ; RQL 27
PXD, (I)    ; LXA INDEX, (I)  ; STA*13
CLA A(I+3, 2*J, 5*K-1) ; LBT
```

Special Instructions

These special symbol-defining ops permit the entry of program constants. They must be placed before the first executable statement of a FORTRAN program. There can be only one constant per card and they cannot be referred to as arrays.

A. Format

col. 1 2 5 7 ----- 72

S FORTRAN variable name OP and constant

B. Instructions

DEC: Any positive fixed or floating-point number conforming to the FORTRAN specifications of constants.

OCT: A string of from 1 - 12 octal digits which will be packed to the right of the word. The sign must be made a part of the octal digit as opposed to signed octal numbers in SAP.

ALF: A string of 6 alphanumeric characters. Since blanks are significant, a rigid card format is required here. The Op must be in columns 7 - 9 followed by 3 blanks and the next 6 columns (10 - 15) are taken to be the constant.

Examples of these special instructions are:

IZ DEC 1 ; MR OCT 7777777777

NAME ALFbbbROBERT ; PI DEC 3.1416

Part 2 -

There is a set of restrictions imposed on the use of these instructions. They are as follows:

1. The TSX instruction cannot be used. In its place it is recommended that the Call statement of FORTRAN II should be used when referring to a subroutine.

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

a. If a copy skip	CPY
is anticipated	TRA
	TRA
	TRA

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

3. Observe that MSE and PSE are missing from the instruction list. The equivalent SAP ops should be used instead.
4. All transfer instructions must have addresses which are statement numbers in the program. Moreover they should not be modified either by actual address modification or effective address modification (indexing). The GO TO statement provides this ability.
5. All index registers are symbolic. Absolute tags are not permitted.
6. All instructions referring to index registers in an active way (PAX, SKD, TXL ----) can be only single dimensioned and without a coefficient or addend.
7. Relative addressing is not permitted except via addends to subscripts, i. e. A(I+3).
8. As in FORTRAN all parts of the program must have some path of control leading to it. Calling sequences and subroutines in general should use the CALL subroutine mechanism of FORTRAN II.

Along with these restrictions, there is a list of coding habits to be observed as well as assorted information about instruction behavior. These are listed as follows:

1. All blanks anywhere in an instruction are ignored except for ALF, and empty fields may be indicated by 2 consecutive commas.
2. All skip type test instructions result in 3 actual instructions and the CAS in 4 actual instructions. This is required by the FORTRAN apparatus. The additional instructions are unconditional transfers. As a result caution should be exercised when using these instructions in the trapping mode.
3. The definition of subscripts and indices:

The value of a subscript - i. e., the value of the (symbolic) index register - may be established by defining the indices of the subscript by means of FORTRAN statements (DO, arithmetic, and input) or the machine instructions, STO, SXD, STD, SLQ, STQ. The value of such an index, -

i. e., the value of the fixed-point variable - is thereby also established. This value must be a positive integer in the decrement, with the rest of the word zero.

Alternatively, in the case of a one dimensional subscript without coefficient or addend, the value of the subscript may be established by means of an active instruction, i. e., one that initially establishes or modifies a Tag (LXA, TIX, . . .) whose tag is the same as the subscript. This method, however, does not serve to establish the value of the fixed-point variable which is the index of the subscript.

Example 1.

```
S          STO J
          DO   10 I = 1, 5
S          LDQ A(I, J)
S          10  SLQ B(I, J)
```

This example illustrates the use of the DO and the STO to define the indices I and J and, hence, the subscript, (I, J).

Example 2.

```
S          LXD N, (I)
S          TXI*2, (I), 1
S          2  CLA A(I)
```

This demonstrates how the initializing and modifying indexing instructions serve to define a subscript. However,

```
S          LXD N, (I)
S          TXI*2, (I), 1
S          2  CLA A(3*I)
```

will not work because, in fact, the tags in the LXD and TXI are not the same as in the CLA. It should be

```
S          LXD N, (I)
S          TXI*2 (I), 1
S          2  SXD I, (I)
S          CLA A(3*I)
```

The SXD serves to define the value of I.

4. 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.

STO, SXD, STD, SLQ, STQ

c. Execution of the following FORTRAN statements:

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

5. The DED instruction written as 'DED, (I)' is a non-executable instruction (there should be no transfers to it) that indicates a subscript is no longer of any use. This is not necessary but will improve the efficiency of the program.

6. Frequency statements can be used to describe all the conditional transfers, and skip type machine instructions. For conditional transfers list the frequency of the transfer address first and then the frequency of the alternate path. For skip type instructions frequencies are specified in the order of skip not taken, skip one instruction and skip two instructions. As in FORTRAN this information is not required, but will improve object program efficiency.

7. All storage allocation must be done via Common, Dimension and Equivalence statements.

8. It should be noted that the FORTRAN treatment of symbolic index registers can result in additional instructions for preserving and loading. In order to minimize these insertions where timing is critical three things can be done.

- a. Use as few symbolic tags as possible.
 - b. Use the DED instruction.
 - c. Use Frequency statements.
9. Note that all SAP pseudo-ops other than the modified OCT, DEC and ALF ops are omitted from the instruction list.

C Example using F3 to generate truth table of
C any Boolean expression 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

 DO 10 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 7777

 2 FORMAT (6(5B11))

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.

FORTRAN Machine Instructions - Mnemonics Used Are The Same As SAP

ACL	DVP	OCT*	SLT*	TOV
ADD	ETM	ORA	SLW	TPL
ADM	FAD	ORS	SPR*	TQO
ALF*	FDH	PAX	SPT*	TQP
ALS	FDP	PBT	SPU*	TRA
ANA	FMP	PDX	SSM	TTR
ANS	FSB		SSP	TXH
ARS	HPR	PXD	STA	TXI
BST	HTR	RCD	STD	TXL
CAD	IOD	RDR	STO	TZE
CAL	LBT	RDS	STP	UFA
CAS	LDA	REW	STQ	UFM
CFF*	LDQ	RND	STZ	UFS
CHS	LGL	RPR	SUB	WDR
CLA	LLS	RQL	SWT*	WEF
CLM	LRS	RTB	SXD	WPR
CLS	LTM	RTD	TIX	WPU
COM	LXA	RTT	TLQ	WRS
CPY*	LXD	SBM	TMI	WTB
DEC*				
DED*	MPR	SLF*	TNO	WTD
DCT	MPY	SLN*	TNX	WTS
DVH	NOP	SLQ	TNZ	WTV

* see comments on Operations