May 3, 1960

STRETCH

MATH SUBROUTINE CONVENTIONS

Array Storage Assumptions

We shall define the standard method of storing arrays to be as follows:

1. One-dimensional array are stored

$X_1 \in L$

$X_2 \in L + 1$

$\vdots$

$X_n \in L + n - 1$

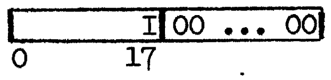
2. Two-dimensional arrays of order m x n may be embedded in a larger array of order M X N. Let $A = (a_{ij})$, $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$. Then standard storage is defined to be

$$\begin{array}{lll}
a_{11} \in L & a_{12} \in L + M & a_{1n} \in L + (n-1)M\ldots \\
a_{21} \in L + 1 & a_{22} \in L + M + 1 & a_{2n} \in L + (n-1)M + 1\ldots \\
\vdots & \vdots & \vdots \\
a_{m1} \in L + m - 1 & a_{m2} \in L + M + m - 1 & a_{mn} \in L + (n-1)M + m - 1\ldots \\
\vdots & \vdots & \vdots
\end{array}$$

Unless the nature of the routine is such as to indicate a more general type of storage is required, this will be the storage assumption made by subroutines.

Types of Data

Math subroutines operate with two types of data: (1) floating point numbers and (2) parameters or fixed point integers. The floating point number is the standard floating point number of the machine. An integer will be a positive 18 bit quantity occupying the leftmost 18 bits of a half word, i.e., an integer I should appear in a half word as follows:



Types of Routines

There will be two classes of math subroutines, functions and routines, defined as follows:

1. Functions are characterized by the fact that they have only a single result, and furthermore, this is a floating point result. We further subdivide functions into two classes:

    a. Functions which have only a single argument of input (floating point).

       EXAMPLE: The elementary functions, Sin X, etc.

    b. Functions which have multiple argument input.

       EXAMPLE: The determinant of a matrix.

2. Routines are characterized by the fact that they have multiple output.

   EXAMPLE: A Runge-Kutta routine for solving simultaneous first order differential equations.

Calling Sequences

1. Functions of Class I will have the following type calling sequence:

Enter with the argument in the accumulator

```
LVI, ∅15, ∅ +1.
BE, function
BE, ERR1
BE, ERR2
     .
     .
     .
BE, ERRn
```

return here with the result in the accumulator.

The indicators must reflect the status of the argument upon entry and the result upon exit. It is clear that the following form is sufficient to describe the necessary calling sequence: Function (ERR1, ERR2,...,ERRn)

2. Functions of Class II will have their arguments appearing in the calling sequence just following the branch to the subroutine. The form of the calling sequence is as follows:

```
LVI, ∅15, ∅ +1.
BE, funct
OP, arg 1
OP, arg 2
     .
     .
     .
OP, arg n
BE, ERR1
BE, ERR2
     .
     .
     .
BE, ERRm
```

return here with the single result in the accumulator.

The precise form and meaning of OP, arg is defined in the following section. The following form is sufficient to define the necessary calling sequence:

Function (arg 1, arg 2,...,arg n, ERR1, ERR2,...,ERRn).

Arg i represents result and input. Upon exit, the indicators will reflect the status of the result.

3. For routines the form of the calling sequence is as follows:

```
LVI, Ø15, Ø + 1,
 BE, routine
 OP, arg 1
  .
  .
  .
 OP, arg n
 BE, ERR1
 BE, ERR2
      .
      .
      .
 BE, ERRm
 return here.
```

The following form is sufficient to define the calling sequence:

Routine (arg 1,..., arg n, ERR1,...,ERRm).

<u>Form of Arguments for the Calling Sequence</u>.

Arguments for the calling sequence and their associated operations are of the following type:

1. The location of floating point data. The entry in the calling sequence is BE, X (I) where X (I) is the effective address of a floating point operand. I may be any index register from 1,...,12.

2. A parameter or integer. The entry in the calling sequence is of the form BE, N (I) where the effective address N (I) is the integer. The most frequent example of such an entry should be of the form BE, 5.0. Either N appears as an explicit integer, or is defined by a synonym card.
3. A location of an integer. The entry in the calling sequence is LVI, , FIVE (I) where Five is the location of a half word containing an integer of the form in 2. FIVE must not be defined by a synonym card.
4. A location of a function or a piece of code. The entry in the calling sequence is

   BE, funct

   or, BE, error

   wherever an integer is required in the calling sequence, the user may use either form 2 or 3. The code is so written that it is immaterial which type of entry is used for integer entry.

Restrictions on the subroutine user

In order that indexed arguments may appear in the C.S., index registers 1,2,...,12 may be used for this purpose.

Index $\not{}$15 will be used for linkage and $\not{}$13 and $\not{}$14 must be free for use upon entry. They will not be saved by the subroutine.

Special machine registers such as the accumulator, remainder register, etc. cannot be used as the location of subroutine arguments. Index registers 0 - 12 are allowed, but 13 - 15 are not.

Subroutines will assume that upon entry the "Standard Fix-ups" (as defined by the supervisor) are operative. If the user has gone non-standard with regard to fix-ups programs, it is his responsibility to see that the "Standard Fix-ups" have been restored.

Certain indicators may be altered by the subroutine without having to save the indicator register ($IND). Of the maskable indicators, these are:

Class	Type
Instruction exception	Data fetch only
Result exception	Lost carry, partial field, Zero Divisor
Result exception - floating point	All
Flagging	All

All non-maskable indicators except for the noisy mode indicator may be altered by the subroutine.

The error exits will be the last entries in the calling sequence.

Restrictions on the subroutine coder

If the subroutine affects indicators other than those listed, it must save the indicators and restore them. An effort should be made to determine the indicators that a subroutine can affect and publish these with the subroutine write-ups. If a subroutine needs a different fix-up routine from the standard then the indicator register and mask register must be saved and restored and the standard technique for changing the interrupt table will be used.

Arguments from the C.S. must be obtained by using an LVE instruction.

Index registers 0 thru 12 must be saved and restored if they are used by the subroutine. Furthermore, index registers 1 thru 12 must be in a restored status when an LVE is given in the subroutine to fetch an argument from the calling sequence.