December 29, 1955
Poughkeepsie, New York

PROJECT STRETCH
MEMO #19

COMPANY CONFIDENTIAL

SUBJECT: Subprogram Technique on Stretch
By - N. Rochester

An objective of the logical design of Stretch is to make subprogram techniques so good that it will be practical to write interpretive programs that are many layers deep. This memo describes one subprogram technique that reflects current thinking. In this technique, the interpretation is done partly by the assembly program and partly on a current basis.

The assembly program converts the abstract instructions to calling sequences for closed subprograms and these subprograms are executed by the machine. The assembly problem is trivial and will not be discussed here.

The 704 subprogram technique is efficient but its scope is limited to a single layer of subprograms. A generalization is needed to make subsubprograms etc., work out efficiently.

One key element of this variation is the Transfer and Set Index instruction, TSX. This sends the contents of the program counter to the index register specified by the TSX instruction and sends the address part to the program counter.

Another key element is a generalized indirect instruction. When the machine encounters an indirect instruction it gets the contents of the indicated address, but it already has the information that this will not be the operand but merely another address that will eventually lead to the operand. This address will be marked as either direct or indirect. If it is direct it is the address of the operand. If it is indirect it merely leads to another address that may, in turn, be either direct or indirect. Eventually this course must lead to a direct address and thence to the operand.

Chain
addr.

Consider an instruction in a subsubprogram that needs data from the main program. It will be an indirect instruction and its address will locate an indirect address in the subprogram which will locate a direct address in the calling sequence in the main program which will lead to the operand elsewhere in the main program. This fundamental idea can easily be extended to any number of layers of subprograms.

There are certain ramifications of this technique that require quite a lot of indexing. The problem and solution can best be described by giving an example. The simple example chosen is double precision complex addition. The subprogram does complex addition while the subsubprogram does double precision addition. The object is to write a calling sequence in the main program in such a way that the calling sequence may be regarded as a three address instruction that causes the machine to execute double precision complex addition.

This "three address" instruction will need to be indexed separately in each of its three addresses. The example shown below is almost too simple to show the advantage of using subprograms. However, this example was chosen because it illustrates the subprogram technique without getting involved in any irrelevant complications.

In the program illustrated below, there are four parts of each instruction: operation part, tag, address part, and designation (of whether it is immediate (in), direct (d), or indirect (i)). An isolated address will have its operation part used as a second tag. In this case the address in the finished instruction ready for execution will consist of the sum of the address part, and the contents of the two index registers indicated by the two tags. The location of the instruction will precede the instruction.

A Detail of the Master Program

```
u-1   Previous Instruction
u     TSX   1   v   d    These four instructions are the three-address
u+1    5    2   α   d    double precision addition instruction.
u+2    5    3   β   d
u+3    5    4   γ   d
u+4   Next Instruction
```

In the above, index register 1 is used in the subprogram technique to enable the subprogram to find the data in the calling sequence. Index registers 2, 3, and 4 are used in indexing the add instruction. Index register 5 serves a purpose that has not yet been discussed. α and β as modified by 2 and 3 are the locations of the operands and γ as modified by 4 is the location of the sum.

The problem that has not yet been discussed is that not all of the addresses are given by the calling sequence. Each double precision complex number occupies four successive words in memory: lesser real, greater real, lesser imaginary, and greater imaginary. Only the address of the lesser real word is given by the master program and it is up to the subprograms to calculate the others. Index register 5, mentioned in the previous paragraph serves this purpose.

The Complex Addition Subprogram

```
v      LXA   5   0   m       (Load Index Register 5 with 0)
v+1    TSX   6   w   d   }    double precision
v+2     -    1   1   i   }    addition of the
v+3     -    1   2   i   }    real parts
v+4     -    1   3   i   }
v+5    LXA   5   2   m       (Load Index Register 5 with 2)
v+6    TSX   6   w   d   }    double precision
v+7     -    1   1   i   }    addition of the
v+8     -    1   2   i   }    imaginary parts
v+9     -    1   3   i   }
v+10   TRA   1   4   d       (Return to master program)
```

The Double Precision Subsubprogram

```
w     CLA   6    1    i        do the addition of the
w+1   ADD   6    2    i        lesser words.
w+3   STO   6    3    i
w+4   TXI   5    1    m        change to greater words
w+5   ARS   -    36   m        save overflow bit
w+6   ADD   6    1    i        do the addition of the
w+7   ADD   6    2    i        greater words
w+7   STO   6    3    i
w+8   TRA   6    4    d        return to subprogram
```

Consider the instruction at w in the subsubprogram. Its action will be first described roughly and then in great detail. It is indirect so it causes the machine to interpret the word at v+2 (or v+7) as an address. This in turn is indirect so it causes the machine to interpret the word at u+1 as an address. This is direct so the machine interprets the word at α (or $\alpha+2$) as an operand. If this is sufficiently clear to the reader he may skip the next three paragraphs.

Now consider the operation of the instruction at w in the subsubprogram in more detail. When it is used the first time, index register 6 contains v+1 which it got from the TSX instruction in the subprogram. This number from index register 6 is added to the 1 from the address part of the CLA instruction to give v+2. The instruction at w is marked i (indirect) so the machine interprets the word from v+2 as an address.

In executing the word from v+2, the machine gets u from index register 1 which it adds to the 1 in its own address part giving u+1. Since v+2 is marked i (indirect) it interprets the word from u+1 as an address.

In executing the word from u+1 the machine gets 0 from index register 5 and some unknown number that will be designated as X from index register 2. This yields the address $\alpha+0+X=\alpha+X$. The machine uses the number from $\alpha+X$ as an operand because the instruction at u+1 is marked d (direct).

Notice that the TXI (transfer with index incremented) instruction at w+4 is used to shift from the less significant to the more significant half of the word. The LXA (load index with address) instruction at v sets the machine to the real part and the LXA instruction at v+5 sets the machine to the complex part.

Now it is worthwhile to measure the cost of what has been done. The double precision complex instruction required 78 machine cycles (using 704 timing). With direct programming the job would have taken 34 cycles. An intermediate approach would have been to use a subprogram but no subsubprogram. This subprogram would be:

```
   v     LXA    5    0    m
   v+1   CLA    1    1    i
     2   ADD    1    2    i
     3   STO    1    3    i
     4   ARS    -   36    m
     5   TXI    5    1    m
     6   ADD    1    1    i
     7   ADD    1    2    i
     8   STO    1    3    i
     9   TXI    5    1    m
    10   CLA    1    1    i
    11   ADD    1    2    i
    12   STO    1    3    i
    13   ARS    -   36    m
    14.  TXI    5    1    m
    15   ADD    1    1    i
    16   ADD    1    2    i
    17   STO    1    3    i
    18   TRA    1    4    d
```

The speeds of these three approaches are:

	machine cycles
Direct Program	34
Subprogram	58
Subsubprogram	78

The speed disadvantage in the multi-layer subprogram technique would be less when it was applied to larger problems. This method appears to offer a considerable improvement over previously available subprogram methods but it still contains redundancies and should be capable of further improvement.

The original notebook entry on this subject is dated December 9, 1955 on pp42-43 of Notebook 1337 issued to N. Rochester.