PROJECT STRETCH
LINK COMPUTER MEMO NO. 4

September 18, 1956

TITLE: Core Shift Matrix and Arithmetic Operations
BY: W. Wolensky

The operation of the core shift matrix, being considered for the Link Computer, is briefly, but specifically described. The arithmetic operations are also briefly described. It is implied that the arithmetic qualities are applicable to both binary and decimal modes of operation. Restrictions presently considered are:

a. For addition and subtraction - none.

b. For multiplication - limited to a maximum of one word of multiplicand, and one word of multiplier, resulting in a two word product.

c. For division - limited to a two word dividend and a single word divisor, resulting in a one word quotient and a one word remainder.

The actual organization and control of the register and logic system will be described in a subsequent memo.

## Core Shift Matrix Operation Definition

1. Contents of registers drive down into vertical lines storing ones or zeros into the whole column of cores. The contents of the register are unchanged unless a specific register reset is provided separately.

2. Information is driven out of the matrix (set to zero) by driving the appropriate diagonals; sensing is done by the horizontal wires.

3. Two methods of setting the matrix are possible.

    a. Reset the matrix, drive only one bit lines for a complete cycle.

    b. Never reset the matrix, always use a bi-polar drive when setting e. g., drive to set desired cores to ones and drive to set desired cores to zeros.

    Therefore, every core will be driven on a "set matrix opn.".

4. Re-entry of data into registers through core shift matrix. Two methods are possible.

   a. Use the core matrix as a crosspoint switch. A diagonal is energized, specifying through half select, the cores to be used; information travelling on horizontal lines provides the second half select for cores on intersecting lines. The cores transmit this one and zero coincidence and bi-polar drive into register area specified by half select diagonal.

   b. Use the core matrix to store the zeros and ones of coincidence. When entry to register is desired, the horizontal lines drive cores to zero (reset), outputs sensed by the vertical lines drive zeros or ones into the register depending upon the condition of the core associated with the register stage.

- - - - - -

Link - Addition (Subtraction)

Add: Addend in A, Augend in B, Sum in C or B (at this point optional choice)

1. Drive both A & B into their respective core shift matrices.

2. Pulse read-out diagonals of respective shift matrices to provide first byte of each arithmetic field to adder.

3. Adder accepts and adds (or subtracts) with modifications according to system (binary or decimal, etc.) or carry from preceding cycle.

4. Carry out stored for subsequent cycle, and sum transmitted to first byte position of register C.

   a. If sum is to be returned to register B then, Register B is reset after Step 1 above or sum is bi-polar driven into Register B through input matrix switch of Register B.

   b. Sum into Register B can either be restricted to starting at least significant position or to starting at the least significant position of the augend word in Register B (which may not be the least significant byte position of the Register B).

5. Repeat stepping out successive bytes until:

   a. The first (smallest) field is exhausted, then step only the larger field through the adder until it too is exhausted.

   b. Both fields are exhausted simultaneously.

6. If a carry out remains after both fields are exhausted, indicate an overflow, or if the equivalent occurs and an actual subtraction was in progress, recomplement the sum (difference), from the least to the most significant digit.

- - - - - -

Link - Multiplication
Stepping Thru

Multiplicand in A, multiplier in B, Product in C (low) to B (high).

1. Drive both A and B into their respective core shift matrices, Reset Register C (if C not initially reset, product can be added to contents of C).

2. Pulse out first least significant digit multiplier byte into analyzer and determine the cumulative addition sequence to be employed e.g. Mpr = 6, therefore, dec. might be plus 2 Md, plus 2 Md plus 2 Md.

3. According to cumulative addition sequence. Step out progressively one byte at a time from the low order of Register C and Register A. Add the bytes together after first modifying the byte from A as specified in Step 2. (e.g. through doubler).

4. Store the resulting sum (partial product) in low to high order of Register C.

   a. Cycle Steps 3 and 4 through every multiplicand byte of Register A.
   b. Carry over partial product into Register B if necessary.
   c. Since Register B cannot be reset, any partial product to be stored in Register B will blank out or replace a multiplier digit. The entry of the partial product into Register B, must be from a bi-polar drive.

5. If the multiplier digit calls for a repeat or another cycle, reset all stepping to their original setting, Drive A, B and C into their respective shift matrices and repeat Steps 3, 4 and 5.

    a. Since partial products entering Register B are required to be bi-polar, it is advisable and beneficial to have partial product entering Register C, be bi-polar also.

6. After completing the multiplier digit specified sequence (of step 2):

    a. Drive Register A, B, C into their respective core shift matrices.
    b. Advance counter controls on Register B and Register C.
    c. Drive out a new multiplier digit and have analyzer determine new sequence.

7. Repeat operations above until all multiplier digits are exhausted. Then, the product will be found in Registers B and C and the retained multiplicand in Register A.

8. Sign conventions to be determined, automatic skip over multiplier digit of zero is implied.

- - - - - -

Link - Division

Divisor in A. Dividend in B, C. Quotient in B, Remainder in C.

A. Drive A, B and C into their respective core shift matrices.

1. From known or identified starting points, step through dividend and divisor to find most significant digit (byte) of each.

    a. In fixed word operations, it may be possible to automatically start with the most significant byte position and step toward the least significant byte position until the first non zero digit is located.

    b. From the identified starting point, step toward the most significant byte, then when limit of length is reached, start stepping back until first non zero digit is located.

    c. From the identified starting point and given field lengths, go right to the most significant byte, step toward least significant byte until first non zero digit is located.

2. Feed the most significant dividend and divisor digits to the divide analyzer.

3. Execute a divide cycle: e. g. sample indicated here in generation of quotient digit = 2, analyzer indicated an operation of -2 Dr. from Dd. Therefore,

   a. Step through divisor and dividend simultaneously, one time for each digit in the divisor.

   b. Feed the divisor digit through the doubler (specified by analyzer) and subtract it from the dividend digit. Store the result of the digits subtraction in the location of the used dividend digit (into the register).

   c. Shift to use the next most significant dividend and divisor digits, repeat steps 3b and 3c.

   d. Continue items in step 3 until the most significant digit of the divisor has been processed. (This implies that the most significant digit of the dividend has gradually been reduced to zero, and a divide cycle completed.

4. Control functions relative to a divide cycle.

   a. If remainder is + add 2 (in this case) to quotient digit generator, and analyze new significant dividend digit (remainder) vs divisor most significant digit.

   b. If remainder is - (assuming new method of divide control, it cannot be minus in this illustration), shift dividend one position to left, (no actual shift is involved. The newly established relationship appears as though the Dd was shifted), start negative remainder type of operation (add + 1 Dr. to Dd shifted to return remainder to plus.

   c. When quotient digit is determined, and a dividend digit effectively reduced to zero, drive the quotient digit into the register (through the core matrix and bi-polar drive) in the position originally occupied by the most significant dividend digit.

   d. The divide procedure is completed when the number of dividend digits equal in number to the difference of number of digits in original dividend and divisor has been reduced to zero.

   e. The dividend remaining after step 4d is the remainder.

f. After a subtraction cycle, and re-analysis of the most significant dividend vs divisor digit, the contents of the Registers A, B and C are again driven into their respective core shift matrices.

g. The core shift matrices can be operated as defined in Item 1.

WW:gmp

W. Wolensky