COMPANY CONFIDENTIAL

PROJECT STRETCH

FILE MEMO #34
SUBJECT: Multiple Precision Multiplication
BY: W. Wolensky
DATE: April 23, 1956

Multiplication of two multiple precision numbers poses problems that can readily be solved by programming once the peculiarities of the operation are understood. The aspects of multiplication are discussed briefly and the peculiarities encountered in multiple precision operations explained.

Multiplication of a four digit number by a three digit number can at best produce a seven digit product. The rule being that the maximum number of digits in a product is equal to the sum of the digits in the multiplier and multiplicand. Digital computers are designed to provide for the maximum number of digits possible. An interesting discussion can be generated around the significance, or meaningfulness of the extra digits acquired as a result of multiplication, particularly when the numbers involved are fractions. If two quantities are assumed, which are known to be accurate only to the fourth decimal place, (hence are four digit fractions) and are multiplied together, the resulting product is composed of up to eight digits. The significance of the extra four digits will not be interpreted, but the approach followed in the illustrations of the memo will be that the same degree of precision will be retained at the conclusion, as was had at the inception.

For discussion and illustration purposes, the multiple precision numbers to be used will be A and B, where both are of precision three and can be written as: $A = A_1 + A_2 + A_3$. The notation $A_1 + A_2$ implies the sum of A_1 and $A2$ where the value of A_2 is to a power smaller than the power of A_1 by the number of significant digit positions in the fraction of A_1. If the exponent of A_1 is x and the number of digits comprising the fraction equals 48, then the exponent of A_2 is equal to x-48. Likewise, if the fraction of A_2 contains 48 digits then the exponent of A_3 is equal to x-48-48 or x-96.

Every word or word section upon being multiplied will double its size. The new double sized word section will be divided into two word sections, ultimately resulting (starting with words of precision 3) in a product with precision 6 (which will be rounded to precision 3 after normalization). Each section of a divided word will be identified by the factors composing it and a mark to designate whether it is the most significant "m", or least significant "s" portion of the split word.

The multiple precision multiplication described in the program procedure is symbolically represented here, starting with the given multiplier and multiplicand words of precision 3.

$$A = A_1^{x} + A_2^{x-48} + A_3^{x-96}$$

$$B = B_1^{y} + B_2^{y-48} + B_3^{y-96}$$

Multiplying the individual sections of A and B results in the following table.

Factors involved		Resulting term	Value of exponent	+	Resulting term	Value of exponent
$A_3 B_3$	=	$A_3 B_3$ m	x+y-192	+	$A_3 B_3$ s	x+y-240
$A_2 B_3$	=	$A_2 B_3$ m	x+y-144	+	$A_2 B_3$ s	x+y-192
$A_1 B_3$	=	$A_1 B_3$ m	x+y-96	+	$A_1 B_3$ s	x+y-144
$A_3 B_2$	=	$A_3 B_2$ m	x+y-144	+	$A_3 B_2$ s	x+y-192
$A_2 B_2$	=	$A_2 B_2$ m	x+y-96	+	$A_2 B_2$ s	x+y-144
$A_1 B_2$	=	$A_1 B_2$ m	x+y-48	+	$A_1 B_2$ s	x+y-96
$A_3 B_1$	=	$A_3 B_1$ m	x+y-96	+	$A_3 B_1$ s	x+y-144
$A_2 B_1$	=	$A_2 B_1$ m	x+y-48	+	$A_2 B_1$ s	x+y-96
$A_1 B_1$	=	$A_1 B_1$ m	x+y	+	$A_1 B_1$ s	x+y-48

The eighteen words resulting from the multiplication and stored in memory must be combined, or summed according to the exponent of the fraction. Carry outs must be propagated to terms with the next higher exponent. Final operations can include normalization and rounding off to the proper degree of precision.

Special instructions or special modes of instructions required for properly executing multiple precision multiplication include:

1. A multiplication instruction that does not embody an automatic rounding or an automatic normalization feature.

2. An instruction that normalizes and counts the number of bit positions shifted.

3. A rounding instruction which rounds off a word according to specification, going to register M if necessary, to determine the basis for rounding contents of register L.

The procedure for programming a multiple precision multiplication is demonstrated in a broad program outline, detail being added where necessary to explain the individualities of multiple precision. Figure 1 should be used as a reference for the program.

Given: Triple precision words A and B, with no regard as to signs or relative value to each other necessary.

1. Check the signs of words A_1 and B_1, determine if they are alike or different and store this information for later use. Remove or make all signs of word sections involved positive.

2. Add the exponents of words A_1 and B_1, test for exponent overflow and if no overflow store this information for later use.

3. Adjust index registers to properly locate word sections for program actions and execute necessary housekeeping operations.

4. Load A_3 into register L, multiply by B_3 (with no automatic normalization or round off).

   a. Store contents of L into memory at A_3 B_3 m.
   b. Store contents of M into memory at A_3 B_3 s (which is also P_6).

5. Load A_2 into register L, multiply by B_3.

   a. Store contents of L into memory at A_2 B_3 m.
   b. Store contents of M into memory at A_2 B_3 s.

6. Load A_1 into register L, multiply by B_3.

   a. Store contents of L into memory at A_1 B_3 m.
   b. Store contents of M into memory at A_1 B_3 s.

7. Load A_3 into register L, multiply by B_2.

   a. Store contents of L into memory at A_3 B_2 m.
   b. Store contents of M into memory at A_3 B_2 s.

8. Load A_2 into register L, multiply by B_2.

   a. Store contents of L into memory at A_2 B_2 m.
   b. Store contents of M into memory at A_2 B_2 s.

9. Load A_1 into register L, multiply by B_2.

   a. Store contents of L into memory at A_1 B_2 m.
   b. Store contents of M into memory at A_1 B_2 s.

10. Load A_3 into register L, multiply by B_1.

   a. Store contents of L into memory at A_3 B_1 m.
   b. Store contents of M into memory at A_3 B_1 s.

11. Load A_2 into register L, multiply by B_1.

   a. Store contents of L into memory at A_2 B_1 m.
   b. Store contents of M into memory at A_2 B_1 s.

12. Load A_1 into register L, multiply by B_1.

   a. Store contents of L into memory at A_1 B_1 m.
   b. Store contents of M into memory at A_1 B_1 s.

13. Load A_3 B_3 m into register L, add A_2 B_3 s.

   a. Test for carry out (no carry out).
   b. Add A_2 B_3 s, and test for carry out (no carry out).
   c. Store contents of L in memory at P_5.

14. Load A_2 B_3 m into register L, add A_1 B_3 s.

   a. Test for carry out, there is a carry out, therefore clear the carry out indication count and remember the fact that there was a carry out.
   b. Add A_3 B_2 m, and test for carry out (no carry out).
   c. Add A_2 B_2 s, and test for carry out (no carry out).
   d. Add A_3 B_1 s, and test for carry out. There is a carry out and this should be counted with the carry out of 14a. (as well as carry outs of 14b., c. if they existed).
   e. Store contents of L in memory at P_4.

15. Load the count of carry outs from step 14 into register L and reset to zero the count word.

a. Add A_1 B_3 m; if carry out, count then proceed (no carry out).
b. Add A_2 B_2 m; if carry out, count then proceed (carry out).
c. Add A_1 B_2 s; if carry out, count then proceed (carry out).
d. Add A_3 B_1 m; if carry out, count then proceed (no carry out).
e. Add A_2 B_1 s; if carry out, count then proceed (no carry out).
f. Store contents of L in memory at P_3.

16. Load the count of carry outs from step 15 into register L and reset the count word to zero.

a. Add A_1 B_2 m; if carry out, count then proceed (no carry out).
b. Add A_2 B_1 m; if carry out, count then proceed (no carry out).
c. Add A_1 B_1 s; if carry out, count then proceed (carry out).
d. Store contents of L in memory at P_2.

17. Load the count of carry outs from step 16 into register L and reset the count word to zero.

a. Add A_1 B_1 m; if carry out there is an error, stop the machine (no carry out).

The multiplication is now complete, there remains only the job of normalizing, rounding and correctly signing the product.

18. Normalize and count the number of insignificant zeros removed from word P_1 in register L. (the illustration shows one insignificant zero removed)

a. For every zero removed, subtract one from the exponent sum created in step 2; attach correct sign to word section P_1. (as determined in step 1)
b. The product word sections P_1 to P_6 should be shifted to compensate for the normalization.

The procedure for propagating the word shifting caused by normalizing is shown in Stretch Memo #30. After bit shifting, all that remains is to round off the product to precision three and attach the proper sign (from step 1) to each word section if it was not done during the word shifting procedure.

19. Load P_4 into register M, load P_3 into register L.

	Exp.	Fract.		Exp.	Fract.		Exp.	Fract.
+A =	010	101	+	—	111	+	—	010
+B =	011	011	+	—	110	+	—	111
+	101							

$$\begin{array}{rl} 010 & = A_3 \\ \times\ 111 & = B_3 \\ \hline 010 & \\ 010 & \\ 010 & \\ \hline \underbrace{001}_{m}\underbrace{110}_{s} & = A_3\ B_3 \end{array}$$

$$\begin{array}{rl} 111 & = A_2 \\ \times\ 111 & = B_3 \\ \hline 110001 & = A_2\ B_3 \end{array}$$

$$\begin{array}{rl} 010 & = A_1 \\ \times\ 111 & = B_3 \\ \hline 100011 & = A_1\ B_3 \end{array}$$

$$\begin{array}{rl} 010 & = A_3 \\ \times\ 110 & = B_2 \\ \hline 001100 & = A_3\ B_2 \end{array}$$

$$\begin{array}{rl} 111 & = A_2 \\ \times\ 110 & = B_2 \\ \hline 101010 & = A_2\ B_2 \end{array}$$

$$\begin{array}{rl} 101 & = A_1 \\ \times\ 110 & = B_2 \\ \hline 011110 & = A_1\ B_2 \end{array}$$

$$\begin{array}{rl} 010 & = A_3 \\ \times\ 011 & = B_1 \\ \hline 000110 & = A_3\ B_1 \end{array}$$

$$\begin{array}{rl} 111 & = A_2 \\ \times\ 011 & = B_1 \\ \hline 010101 & = A_2\ B_1 \end{array}$$

$$\begin{array}{rl} 101 & = A_1 \\ \times\ 011 & = B_1 \\ \hline 001111 & = A_1\ B_1 \end{array}$$

Step 13

$$\begin{array}{rl} 001 & A_3\ B_3\ m \\ +\,100 & A_3\ B_2\ s \\ \hline 101 & \\ +\,001 & A_2\ B_3\ s \\ \hline 110 & = P_5 \end{array}$$

Step 14

$$\begin{array}{rl} 110 & A_2\ B_3\ m \\ +\,011 & A_1\ B_3\ s \\ \hline 1001 & \\ +\,001 & A_3\ B_2\ m \\ \hline 010 & \\ +\,010 & A_2\ B_2\ s \\ \hline 100 & \\ +\,110 & A_3\ B_1\ s \\ \hline 1010 & = P_4 \end{array}$$

Step 15

$$\begin{array}{rl} 010 & Ci \\ 100 & A_1\ B_3\ m \\ \hline 110 & \\ 101 & A_2\ B_2\ m \\ \hline \text{Co} \to 1011 & \\ 110 & A_1\ B_2\ s \\ \hline \text{Co} \to 1001 & \\ 000 & A_3\ B_1\ m \\ \hline 001 & \\ 101 & A_2\ B_1\ s \\ \hline 110 & = P_3 \end{array}$$

Step 16

$$\begin{array}{rl} 010 & Ci \\ 011 & A_1\ B_2\ m \\ \hline 101 & \\ 010 & A_2\ B_1\ m \\ \hline 111 & \\ 111 & A_1\ B_1\ s \\ \hline \text{Co} \to 1110 & = P_2 \end{array}$$

Step 17

$$\begin{array}{rl} 001 & Ci \\ 001 & A_1\ B_1\ m \\ \hline 010 & = P_1 \end{array}$$

	Exp.	P_1		P_2		P_3		P_4		P_5		P_6
+P =	101	010	+	110	+	110	+	010	+	110	+	110
normalized												
+P =	100	101	+	101	+	100	+	101	+	101	+	100
normalized and rounded												
+P =	100	101	+	101	+	101						

Fig. 1