JEG

Poughkeepsie, New York
November 1, 1955

PROJECT STRETCH
FILE MEMO #2

COMPANY CONFIDENTIAL

SUBJECT: Cycle Requirements of the Arithmetic System
By S. W. Dunwell

This memorandum discusses the cycle requirements of several forms of arithmetic units. It is based on the following assumptions, which will be examined in detail in subsequent memoranda.

1. That the master arithmetic unit, the exponent arithmetic unit and probably also the address arithmetic unit, will be binary.

2. That the word will be 60 bits, and that it will be divided 3 bits for the sign, 9 bits for the exponent, and 48 bits for the fraction.

3. That it will be permissible for multiplication to take an average of as many as 24 cycles.

Within this general context there are several basic factors which affect the number of cycles required to perform arithmetic operations. They are:

1. Carry.

2. Skipping over zeros in the multiplier and the quotient.

3. Switching into complementary operation when series of 1's are encountered in the multiplier and the quotient.

4. Generation of multiples of the multiplicand and the divisor.

Carry

It has been a general practice in parallel computers to execute a carry completely across the accumulator each adding cycle. The carry

travels as a wave front through a large number of successive stages of logical switching and amplification. This process takes place very rapidly, and in present machines is relatively instantaneous in relationship to related accumulator operations, such as actuation of triggers.

It is not certain that an analogous situation will exist in the Stretch computer. The 48 bit word is longer than any now in use, requiring that the carry wave pass through more stages of switching and amplification. Triggers and similar devices will probably be operated closer to their ultimate limit. Multiplication and division may take place without the setting of triggers each add cycle, and at a rate substantially faster than that at which triggers can be operated. All of these factors tend to make the carry propogation time take on more finite proportions, and suggest that it may be possible to make the machine multiply and divide faster if limits are placed on the length of permissible carry in any cycle. When the limit prevents a succession of associated carries from being completed, a subsequent additional cycle would be required for the purpose.

It should first of all be noted that a carry across the entire accumulator is not required each adding cycle during multiplication and division. It is only necessary that a carry take place between each position and its immediate neighbor. Further carries arising through carry into positions standing at 1 can be deferred until the next add cycle.

The number of carry chains remaining to be executed at the end of a multiplication or division in which carry has been performed in this way would be approximately the same as the number at the end of an addition or subtraction.

The table following indicates the number of carry chains remaining at the end of each cycle in this extreme case of carrying through only one accumulator stage each cycle.

At End Of	Carry Chains Probably Remaining
Last Add Cycle	12
First Extra Cycle	6
Second Extra Cycle	3
Third Extra Cycle	1.5
Fourth Extra Cycle	.75
Fifth Extra Cycle	.375

From the preceding, it can be seen that there is a probability of less than unity that more than four extra cycles would be required. The average number of extra carry cycles is about five.

A relatively simple compromise might be affected by linking each accumulator position to its two nearest neighbors on each side, and propogating carries through two stages each cycle. If the pairs were linked through a matrix, the duration of the carry pulse would not necessarily be increased at all.

The affect of linking in this way in a 48 binary position adder is as follows:

At End Of	Carries Probably Remaining
Last Adding Cycle	6
First Following Cycle	1.5
Second Following Cycle	.375

There is less than an equal chance that two extra cycles would be required. The average number of extra cycles also would be about two cycles.

Number Base

It should be noted that in a binary machine there is no advantage in using quaternary, octal or larger adder matrices, except for the implied simultaneous handling of carry for pairs, threes, etc. of binary bits. In fact, the terms quaternary, octal, etc. have meaning in a binary accumulator only in relationship to carry.

Skipping Zeros During Multiplication

If no shortcuts are taken, binary multiplication requires as many cycles as there are digits in the multiplier. In the case of Stretch, this would be 48 cycles. For those multiplier digits which are 1's, the multiplicand would be added to the partially developed product. For 0's, no action would be taken.

Since, on the average, 24 of the binary bits will be zeros, the most important single step in reducing the number of cycles is to eliminate

the cycles for zeros. To do this, it is necessary to add a device which is capable of looking ahead in the multiplier for the next 1 and of controlling shifting accordingly. The effect of such a device is to reduce from 48 to 24 the number of cycles required to multiply.

The device need not look ahead very far to be effective, as is indicated by the following chart:

Size of Zero Group	Frequency of Occurrence as A Group	Zeros in such Groups	Cumulation Total of Zeros
0	1/4	6	6.0000
0 0	1/8	6	12.0000
0 0 0	1/16	4.5	16.5000
0 0 0 0	1/32	3.0	19.5000
0 0 0 0 0	1/64	1.875	21.3750
0 0 0 0 0 0	1/128	1.125	22.5000
0 0 0 0 0 0 0	1/256	.6775	23.1775
-	-	-	-
Limits of Sums:	1/2	24.0000	24.0000

From the above table, it can be seen that a look ahead of six binary digits would save 22.5 cycles, and that all further look ahead would save only 1.5 cycles per multiplication.

Skipping Zeros During Division

A zero skip situation exists during division which is quite similar to that during multiplication. Because division is a process of analysis, it is necessary to in some manner determine at each possible juxtaposition of the divisor and the dividend remainder whether or not a subtraction can be made.

If, as in a desk calculator, an arbitrary subtraction is made at each juxtaposition, a single cycle is required for each one in the quotient and two cycles for each zero, or a total of 72 cycles. This number can be reduced to 48 by reversing the add-subtract control instead of taking a correction cycle for each zero in the quotient. Beyond this point, it is necessary to provide a look ahead feature which compares one or more

digits of the divisor with the corresponding digits of the dividend remainder at the proper offset and controls the shift unit accordingly.

The percentage cycle saving achieved with this kind of device is the same as that of a similar length look ahead device for multiplication. A look ahead of 6 digits would save 45 out of the 48 cycles associated with zeros, and would permit 48 place division in 27 cycles.

Shift Unit

It may already be apparent from the above that the shift unit required by the system must be able to juxtapose a 48 bit number against a 96 bit number in any of 48 adjacent positions. This requirement is an inescapable part of all fast multiply-divide systems. Less complex shift units can be used (for example, one shifting to 24 alternate positions) but a price in additional cycles is inevitably exacted.

Complementary Multiplication

Wehave already noted that the number of multiply and divide cycles is related to the number of 1's in the multiplier and the quotient, and that 50% of the 1's occur in groups of two or more. As a consequence, a significant reduction in the number of computing cycles is obtained by any of several systems which shift into complementary operation when a series of consecutive 1's is encountered, in effect permitting them to be passed over like 0's.

The most general case is to substitute ((a + 1) b - b) for (ab) within any portion of the multiplication represented by a succession of 1's. The principle is equally applicable to division.

As an example, let us assume that the machine encounters the sequence -- 0 1 1 1 1 0 -- in the multiplier (or the quotient in prospect). It can substitute -- 1 0 0 0 (-1) 0 -- as the multiplier and take two add-subtract cycles rather than four. Instead of taking as many cycles as digits, it would take two cycles for each series of more than a single 1. Single bit groups would be added.

The device is made even more effective by the fact that the 1 which must be added at the left of the multiplier group may fall in place of a lone zero and effect a closure with a series of 1's as the left.

For example, adjacent groups might be -- 0 1 1 1 0 1 1 0 -- . In this case, the multiplier can be restated as -- 1 0 0 0 (-1) 0 (-1) 0 -- and a total of three add-subtract cycles taken for the two groups. The process can continue on from group to group as long as no more than one zero lies between. Such a closure occurs 50% of the time between any one group and the next.

The number of cycles required with this system is readily computed. As already noted, there are an average of 6 single 1 groups and the same number of single 0 groups. There are also an average of six multiple 1 groups. The average number of cycles is:

	Number of Groups	Cycles
Single 1's	6	6
Multiple 1's followed by single zeros	3	3
Multiple 1's followed by multiple zeros	3	6
Total	12	15

The number of columns of look ahead has the same percentage importance as for the previously discussed zero shift control.

Complementary Division

A situation similar to the above exists in division, except that it is necessary to base control on the quotient predicted by comparing the divisor and the dividend remainder. This is relatively more complex than searching for 1's in the multiplier.

In view of the relatively small importance of division in comparison to the other arithmetic operations and the relative complexity of its control, it is probable that a less effective division control will be used. This may result in a division time 25% or so greater than that of multiplication.

Generation of Multiples

The binary multiples of the multiplicand are obtained by shifting,

and it is for this reason that a 48 way shift unit is so desirable. Of the 2^{48} multiples possible, the 48 obtained by shifting are the most easily generated and the most necessary because of their equal logarithmic distribution through the number spectrum of the machine. They will probably turn out to be sufficient for our purpose. However, the following chart indicates the additional advantage gained from a hexadecimal system involving the generation of the multiples 3, 5, 6 and 7, which cannot be obtained by shifting or complementing.

Case	Value	Obtain As	Cycles
0 0 0 0	0	0	1
0 0 0 1	1	1	1
0 0 1 0	2	2	1
0 0 1 1	3	3	1
0 1 0 0	4	4	1
0 1 0 1	5	5	1
0 1 1 0	6	6	1
0 1 1 1	7	7	1
1 0 0 0	8	8	1
1 0 0 1	9	16-7	1
1 0 1 0	10	16-6	1
1 0 1 1	11	16-5	1
1 1 0 0	12	16-4	1
1 1 0 1	13	16-3	1
1 1 1 0	14	16-2	1
1 1 1 1	15	16-1	1
	Total		16

The above indicates an average of 1 cycle per four bit hexadecimal character, or 12 cycles in all. The further addition of a zero look ahead device would save less than one cycle for a net of 11+ cycles. These speeds compare with the 15 cycles obtained earlier for complementary binary multiplication.

Conclusion:

It is the purpose of this memo to outline some of the elementary considerations in the development of the arithmetic system for STRETCH. No specific conclusions can be reached until more is known about the capabilities of the components.

S. W. Dunwell

SWD/jhm