ENGINEERING DESCRIPTION OF EXPANDED CALCULATE

II Divide:

A Introduction

In any divide operation, the location of the high order quotient position must first be found. This is done by starting at the high order position of the quotient field (fig. 1-a) and counting toward the low order a number of digits equal to the number of significant digits (fig. 1-b) in the divisor. The divisor- of length N significant digits- is then compared to the N high order positions of the dividend (fig. 1-c). If the divisor is larger than the "partial dividend", the first quotient digit is zero, and the divisor is then compared to the (N+1) positions of the dividend, (fig. 1-d). When a partial dividend which is greater than the divisor is found, the divisor is subtracted from the partial dividend until the partial dividend becomes smaller than the divisor. The number of subtractions made determines the magnitude of the quotient digit (fig. 1-e) corresponding to this particular partial dividend. The number of significant digits in the quotient may be increased by adding zeros to the low order positions of the dividend field (fig. 2).

This basic procedure is used in the expanded calculate option of the 1401. However, if the magnitude of the partial dividend is at least twice the magnitude of the divisor, the divisor is doubled before being subtracted from the partial dividend, and the corresponding quotient digit is increased by two.

The locations of the various fields are shown on page IV and are referred to in the explanation which follows:

B Procedure

Prior to giving the divide instruction, an area must be provided in storage for the quotient and dividend. This area consists of a dividend field located anywhere, and a quotient field of the same length as the dividend field, but immediately to the "left" of the dividend field in storage (fig. 3). The number of significant

digits in the quotient may be increased by lengthening the dividend field with zeros in the low order positions, thus increasing the length of the quotient, (fig 4). In this case, the programmer must keep track of the decimal point.

Word Marks must be placed in storage to denote the high order positions of both the quotient and dividend fields. Then the dividend must be placed in the dividend field by either a "move", "load" "reset-add", or "reset-subtract" instruction. Also, the dividend must carry a sign (B bit only for "minus", both A and B bits for "plus") in its' units positions, and there may be no other B bits in the dividend field.

The divisor field may be located anywhere, but may not be greater than the length of the dividend field, and must have a sign (B bit only for "minus", both A and B bits for "plus") in its' units position. No other B bits may be located in the divisor field.

When all of the above conditions have been satisfied, the "Divide" instruction may be given. This is a two address instruction:

% (AAA)(BBB)

where (AAA) must be the address of the divisor field and (BBB) is the address of the dividend field.

This instruction first locates the high order quotient position and indicates this position by inserting the proper sign bits into the quotient field. The high order significant digit of the divisor is then aligned with the high order position of the dividend so that the divisor may be compared to the dividend. The divisor will then be subtracted from the dividend a number of times depending upon the results of this comparison, and the number of subtractions made determines the magnitude of the quotient digit. When the partial dividend becomes smaller than the divisor, the quotient sign will be shifted to the next lower order position and the partial dividend will be extended by increasing the address of its' low order digit, until another subtraction can be made. At the conclusion of the divide operation, the quotient will be located in the quotient field and the remainder will be located in the dividend field.

Two additional address registers are required for this option: the A auxiliary address register and the B auxiliary address register.

The A auxiliary address register always holds the address of the units position of the divisor so that whenever the divisor is to be compared to, or subtracted from, the dividend, the A address register may be returned to its' starting point by effecting a transfer of the contents of the A auxiliary address register to the A address register.

The B auxiliary address register holds the address of the storage location of the units position of the "partial dividend" to which the divisor is being compared and from which the divisor is being subtracted. The address is increased by "1" whenever the partial dividend is found to be smaller than the divisor. This address change, in turn, causes a "shift" in the addressing of the dividend field. The contents of the B auxiliary address register is transferred to the B address register to return the B address register to the proper starting point each time the partial dividend is to be compared to the divisor or the divisor is subtracted from the dividend.

The detailed operation is as follows:

When the Divide Op code (&) is detected in the B register, the A auxiliary address register is gated to receive the A address as is the A address register; then the B auxiliary address register is gated to receive the B address as is the B address register.

The quotient field must first be set to zero, the high order quotient position must be located, and the divisor and dividend must be properly aligned so that they may be compared.

This is accomplished in the following way:

A and B cycles are taken until the word mark in the divisor field is detected. (During the first two cycles, the divisor and dividend signs are analyzed, and the proper quotient sign is stored in a trigger.) B cycles only are then taken until the word mark in the dividend field is detected. This word mark signifies the end of the dividend field and the beginning of the quotient field. Zeros are then read into storage until the word mark in the quotient field is detected. Then,a reverse scan of the A field is taken until the first significant digit in the divisor is sensed. (If there are no significant digits in the divisor field, a latch will be turned on, enabling the programmer to test for overflow.) At this point, the high order position of the quotient field and the first significant

digit of the divisor are about to be addressed. B and A cycles are taken in the reverse direction until the B bit in the units position of the divisor is detected, causing the elimination of succeeding A cycles. During the following B cycle, the proper quotient sign is inserted into the quotient field. This sign indicates the position of the first quotient digit. The next step is to align the high order significant digit of the divisor with the high order position of the dividend. This is done by first continuing B cycles until the word mark in the high order position of the dividend field is detected. A cycles are now taken in the forward direction until the word mark in the high order position of the divisor field is detected. The detection of this word mark starts a reverse scan of the A (divisor) field and A cycles continue until the first significant digit of the divisor field is sensed. At this point, the high order position of the dividend field and the first significant digit of the divisor are about to be addressed. Now B and A cycles are taken until the B bit in the units position of the divisor is detected. The address of the low order position of the first partial dividend is read into the B auxiliary address register during the final B cycle of the alignment of the divisor and dividend fields, and the divide operation proper may begin.

A and B cycles are taken in the forward direction to compare the divisor (of length N digits) to the first "N" digits in the high order positions of the dividend. B-cycles only are then taken in the forward direction until the sign bit in the quotient field is detected.

There are five possible results of this comparison:

a) Twice divisor $<$ partial dividend
b) Twice divisor = partial dividend
c) Divisor $<$ partial dividend $<$ twice divisor
d) Divisor = partial dividend
e) Divisor $>$ partial dividend

The following action, based upon the results of this comparison, then takes place.

a) Twice divisor $<$ partial dividend

A B-cycle is taken to increase the quotient by two. The quotient sign does not change location. A and B-cycles are taken to subtract twice the divisor from the partial dividend and compare the divisor to the reduced partial dividend. (See Appendix) When a word mark is detected, B cycles only are taken until the B bit in the quotient field is detected.

b) Twice divisor = partial dividend

Same as (a). In this case, after subtraction, the partial dividend will be reduced to zero.

c) Divisor < partial dividend < twice divisor

Two B cycles are taken in the reverse direction to increase the quotient by one, and shift the quotient sign to the next lower order position. A and B cycles are then taken to subtract the divisor from the dividend. At the conclusion of this subtraction, the partial dividend is extended to the next lower order position, by increasing the address in the B auxiliary address register. A normal comparison of the divisor with the new partial dividend is now made.

d) Divisor = partial dividend

Same as (c). In this case, after subtraction, the partial dividend will be reduced to zero.

e) Divisor > partial dividend

Three B cycles are taken in the reverse direction. During the first B cycle, the quotient sign is read out of storage. During the second B cycle, the sign is read back into storage, causing a shift in the quotient to the next lower order position. During the third B cycle, the address in the B auxiliary address register is increased by one, causing a "shift" to the next lower order position in the partial dividend. The reverse scan is stopped, and a normal comparison of the divisor with the new partial dividend is made.

The divide process is terminated, and an I/E change is forced when the sign in the units position of the dividend is detected, and a comparison which shows that the dividend is less than the divisor is completed. An I/E change is also forced if, during the alignment process, it is determined that the divisor contains all zeros; that the divisor field is longer than the dividend field; or that the number of significant digits in the divisor is greater than the length of the quotient field.

John Pokoski

John Pokoski
Department 260

JP:bd

APPENDIX

Whenever a subtraction of the divisor from the partial dividend is performed, a simultaneous comparison of the divisor with the new partial dividend is made.

Figure 5 shows the data-flow path from storage into the adder.

A previous comparison has determined whether the divisor or twice the divisor is to be subtracted from the partial dividend.

If the divisor is to be subtracted from the partial dividend, an A cycle is taken during which the divisor digit is moved from storage, through the B register, and into the A register. It is then translated into qui-binary code and gated through the true complement circuitry and into the adder. The output of the translator, and the doubled output of the translator are both read into the qui-binary comparison circuitry.

The same paths are followed if twice the divisor is to be subtracted from the partial dividend except that the divisor digit is doubled before being complemented and read into the adder.

During the following B cycle, the partial dividend digit moves directly into the B register, is translated to qui-binary, and goes into the adder. A complement addition is performed so that either the divisor digit or twice the divisor digit is subtracted from the partial dividend digit. This operation is completed by 7.5 time of the cycle. At this time, the output of the adder is gated into the compare circuitry so that the reduced partial dividend digit is compared to the divisor digit and twice the divisor digit.

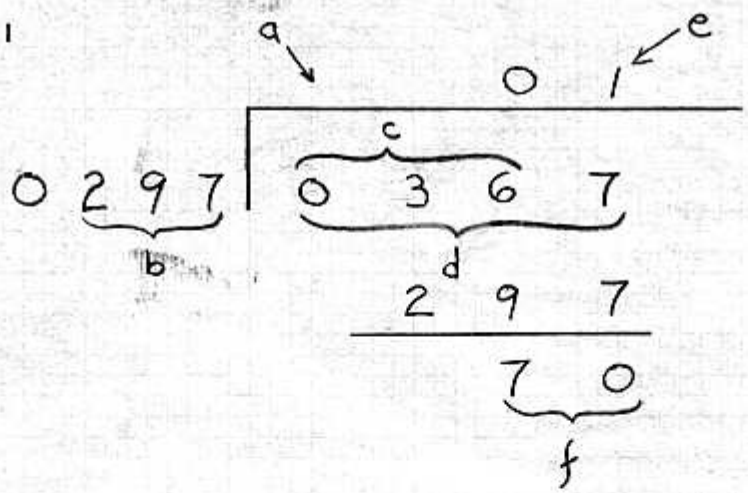
Although the divisor and twice the divisor are being compared to the reduced partial dividend during all subtractions, the results of this comparison are used only after a subtraction of twice the divisor from the partial dividend is completed. The comparison is not needed when a subtraction of once the divisor from the partial dividend is made, since it is already known that, at the completion of this operation, the partial dividend will be smaller than the divisor.

The "set to 1" and "set to 2" gates shown in figure 5 are used to gate either the digit "1" or the digit "2" through the true complement circuitry and into the adder so that the quotient digit may be increased by either "1" or "2", as determined by the previous divisor-partial dividend comparison.

The "set to 0" gate is used to force a zero into the adder when first aligning the divisor and dividend so that the digit in the B register will not be added to the digit in the A register before being gated out of the adder and into the compare circuitry. This gate is also up immediately after the partial dividend address is "shifted", so that a new comparison can be made.

cc: Messrs. O. R. Ahearn
F. W. Baldwin
G. E. Beers
A. A. Bissell
R. L. Bennett
S. Bespalko
E. M. Bloom
G. E. Bonteccu
N. C. Capettini
B. N. Carr
J. S. Conzola
P. Farbanish
J. G. Fitzgerald
E. J. Grenchus
J. A. Harvilchuck
J. J. Ingram
P. E. Johnson
W. L. Kelly
R. W. Phillips
T. Markulin
T. M. McAdon
F. M. Quigley
W. K. Ringa
T. N. Rowe
R. A. Rowley
B. T. Rucker
W. S. Schaffer
K. B. Stoffel
F. G. Underwood
W. J. Zehner

FIG. 1



a) High order quotient position
b) Divisor (of three significant digits)
c) First partial dividend
d) Second partial dividend
e) Quotient digit (determined by number of subtractions made)
f) Remainder

fig. 2

```
      0123
0297 ⌊036700
    -0297
      700
     -594
      1060
     -0594
       466
      -297
       169
```

fig. 3

	Quotient field				Dividend field			
	094	095	096	097	098	099	100	101 AB
B-field	*				0 *	3	6	7
A-field					213 0 *	214 2	215 9	216 AB 7

fig. 4

	Quotient field						Dividend field					
	090	091	092	093	094	095	096	097	098	099	100	101 AB
	*						0 *	3	6	7	0	0
								213	213 0 *	214 2	215 9	216 AB 7

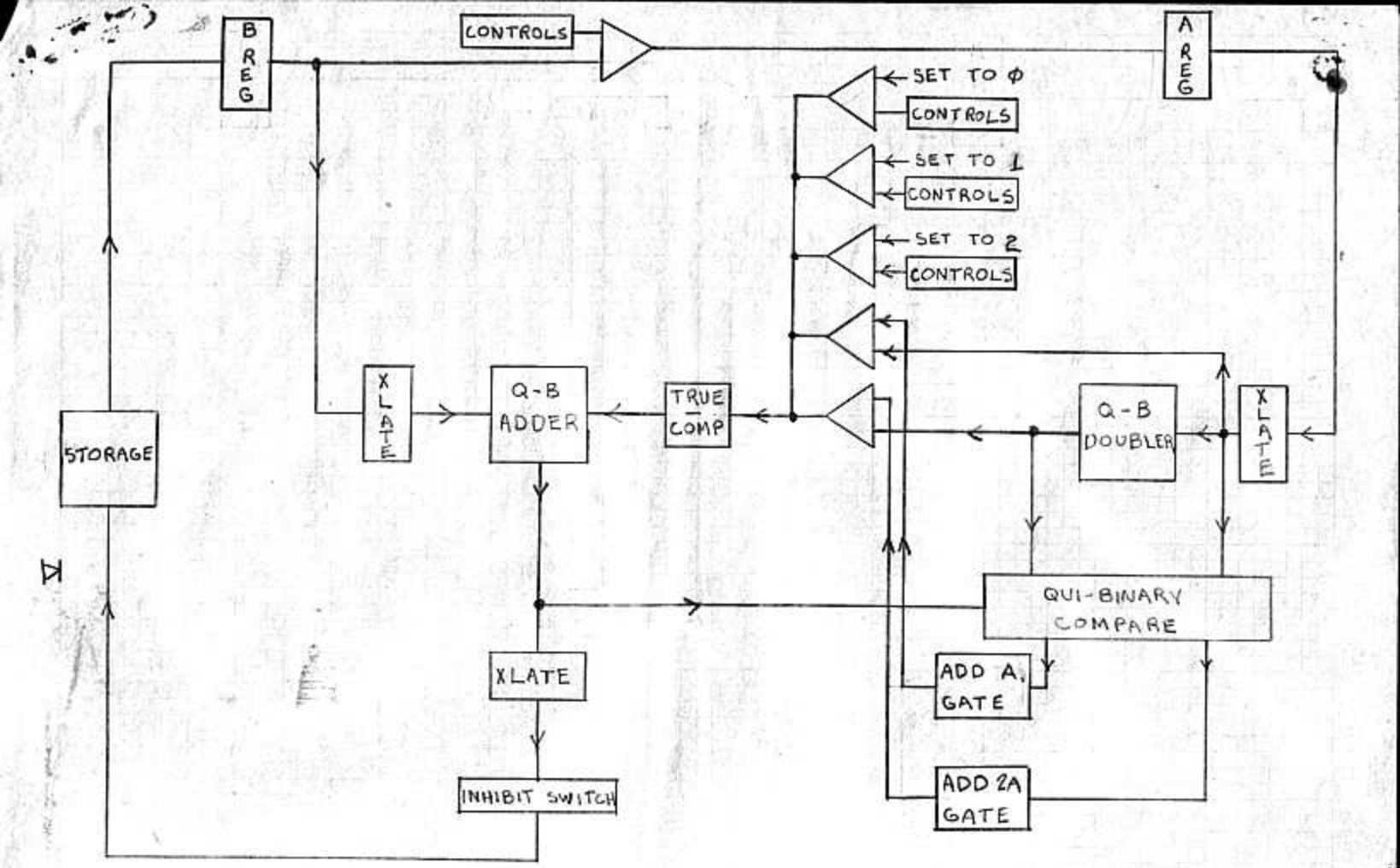


FIG. 5 EXPANDED ARITHMETIC DATA FLOW

J Bokosky 12/1/59