6. Add the first halves of L and S and add the carry bit obtained from step 5.

e	s	v	

If $e = 1$ then an end-around-carry must be performed. This means that a one is added at the right end of the word produced in step 5. Since this might also produce a carry bit, the c in the diagram (see step 5) must be cleared to zero before the end-around-carry. If a carry bit is again produced, then a one must be added at the right end of the word above. It can be shown that this last operation can never produce another $e = 1$.

If $v = s$ then v is a sign bit. However, if $v \neq s$ then there has been overflow during the addition, and v is the most significant bit of the sum. In the latter case an adjustment of the exponent will be necessary to give the correct answer.

7. Shift of second half of the sum left one place to clear out the carry bit c. Then shift the double length sum left (a) one place if $v \neq s$; (b) two places if $v = s$.

This leaves the sum in the following form:

s	47 most significant bits of the sum

at least 37 bits of the sum	sign bits

8. If the double length sum was shifted one place left in step 7 ($v \neq s$) then the exponent must be adjusted to take care of the overflow. This means adding one to the exponent β or δ, whichever is larger. (This will be the exponent of the sum.) If the double length sum was shifted two places left in step 7, no adjustment of exponent is necessary.
9. The form of the sum given by step 8 must now be checked for normalization since it is possible that several of the leading bits of the sum may be zero. (Cancellation occurs when two numbers of opposite sign but nearly equal magnitude are added.) If the sum is not normalized then it should be normalized at this point and appropriate adjustments in the exponent should be made. If 84 left shifts are not sufficient for normalization then the sum should be made zero.
10. At this point the normalized sum may be rounded although the extra coding involved may not be worth the gain. If rounding is desired then there are two cases to be considered depending on the sign of the sum. These two cases require that care be taken in handling any carry bit produced by the rounding operation.
11. Now pack the 84 most significant bits of the sum, along with 12 bits representing the sign and exponent, into two 48-bit words (in the standard way). If the sign of the sum is negative, then the first 12 bits must be complemented before the packing takes place.

D. Substraction

No special subroutine is necessary since

$$F - G = F + (-G), \qquad (29)$$

and one merely complements G before entering the addition subroutine.

PRACNIQUES

The Techniques Department is interested in publishing short descriptions of Techniques which improve the logistics of information processing. To quote from the policy statement, Communications of the ACM 1 (Jan. 1958), 5: "It is preferable that techniques contributed be factual and in successful usage, rather than speculative or theoretical. One of the major criteria for acceptance and the question one should answer before submitting any material is—Can the reader use this tomorrow?" Clear, concise statments of fairly well-known but rarely documented methods will contribute significantly to raising the general level of professional competence.—C.L.McC.

A NOTE ON MULTIPLYING BOOLEAN MATRICES II

In a note by Baker [1], a method is given for getting the limiting connectivity matrix, B, of a matrix whose entries are Boolean 0's and 1's. Harary [2] suggests determining A^{n-1} since $A^{n-1} = A^n = \cdots$, where n is the order of the matrix. The purpose of this note is to give a method for determining A^{n-1}. The method is also applicable to finding the output matrix of a switching network as described in [3] and [4] where again $A^{n-1} = A^n = \cdots$, but A now has Boolean switching functions as entries instead of 0's and 1's.

The procedure is a simple variation of ordinary matrix multiplication. Given a Boolean matrix A whose only entries are 0's and 1's, to get $B = A^{n-1} = A^n = \cdots$, search the first row until a 0 is encountered, say in column j. Replace this 0 by $\sum_{k=1}^{n} a_{1k}a_{kj}$ (the operations here are Boolean). Continue to search columns $j+1, j+2, \cdots$ of row 1 until another 0 is found, say in column m. Replace 0 by $\sum_{k=1}^{n} a_{1k}a_{km}$. Continue this process until the whole row is searched, then process the remaining rows in the same manner. When all the rows have been processed, begin with the first row. Continue processing the matrix in this manner until one complete pass has been made without changing the matrix. A has now been transformed into B.

In the case where the entries of A are switching functions (see [3] and [4]) the same process applied to each entry that is not the Boolean 1 will yield the output matrix.

It can be easily shown that q passes through the matrix where $2^q \geq n - 1$ is sufficient to guarantee that A has been transformed into A^{n-1}. In practice q may be smaller depending on the nature of the matrix entries.

The advantages of this method lie in the reduction of the number of times the operations must be performed and in the fact that a duplicate matrix and other temporary storage locations are not required.

References:

1. Baker, J. J. A note on multiplying Boolean matrices. *Comm. ACM 5*, 2 (1962), 102.
2. Harary, F. A graph theoretic method for the complete reduction of a matrix with a view toward finding its eigenvalues. *J. Math. Physics 38* (1959–60), 104–111.
3. Hohn, F. E., and Schissler, L. R. Boolean matrices and the design of combinational relay switching circuits. *BSTJ 34*, 1 (1955), 177–202.
4. Luntz, A. G. Algebraic methods of analysis and synthesis of contact networks. *Izvestia Akad. Nauk SSSR, Ser. Mat. 19* (1952), 405–426.

D. R. Comstock
Oregon State University
Corvallis, Oregon