described above does not appear to be easy. Still further complications are being investigated by Helmut Maak, IBM Muhlheim, Germany.

Still other variants, both of problem and technique, are possible. Reith has studied the very interesting inverse problem; i.e., given a demand for paper, what width paper machine; i.e., what standard roll is best? Also, in Europe, M. Genuys of IBM Paris has done work on a column-generating technique that uses integer programming at the point where dynamic programming was used in the method described above. Here in the United States, C. E. Berry of IBM Portland has a 1620 program which gives an approximate solution to the problem and proceeds by quite different methods—no linear programming is employed. A technical description of this work will be available soon.

It seems likely that various forms of the Trim Problem will be with us for some time to come.

On modifying the 1620 ADD table

by G. Gerson

The fact that arithmetic is performed on the IBM® 1620 Data Processing System by means of TABLE LOOKUP provides an opportunity for introducing special operations which may be executed by suitable table modification.

It is the purpose of this note to suggest, by means of an example, that many practical problems may be resolved by this technique. The problem faced was as follows.

A 40-digit field was to be tested. Each of the digits of this field arose as the result of a physical test, and was independent of each of the other digits in the field. The physical tests were associated with a production process, and the accompanying computations had to be performed in real time. The field was to be compared with two other 40-digit fields called the *upper limit field* and the *lower limit field.* The requirement was that each digit of the original field had to be less than or equal to the corresponding digit of the upper limit field and greater than or equal to the corresponding digit of the lower limit field. Furthermore, there were 96 sets of these upper and lower limit fields. In the worst possible case, the original field had to be compared to all 96 sets, and the real time requirements were such that all comparisons had to be performed on the 1620 in one second. A careful timing estimate using the COMPARE operation (which required that the digits be stripped away and tested one by one) indicated

that a minimum of 5 seconds would be required to perform all the necessary tests.

To solve the problem, in the case of the upper limit comparison, a noncommutative operation, denoted by $\oplus$, with the property, $a_1a_2 \cdots a_n \oplus b_1b_2 \cdots b_n = c_1c_2 \cdots c_n$ where the a's and b's are digits and $c_i = 0$ if $a_i \geq b_i$ and $c_i = 1$ if $a_i < b_i$, was introduced. Thus, for example, $5 \oplus 6 = 1$ whereas $6 \oplus 5 = 0$.

The manner in which the 1620 ADD table is consulted had been noted (e.g., for the computations $5 + 6$ and $6 + 5$ memory locations 00356 and 00365, respectively, are utilized). Therefore, to execute $\oplus$ the 1620 ADD table was modified by placing *zero* in each cell on the diagonal and below, and *one* in each cell above the diagonal, as shown below. With this table modification, the 1620 will execute $\oplus$ in place of $+$.

	0	1	2	3	4	5	6	7	8	9
30	0	1	1	1	1	1	1	1	1	1
31	0	0	1	1	1	1	1	1	1	1
32	0	0	0	1	1	1	1	1	1	1
33	0	0	0	0	1	1	1	1	1	1
34	0	0	0	0	0	1	1	1	1	1
35	0	0	0	0	0	0	1	1	1	1
36	0	0	0	0	0	0	0	1	1	1
37	0	0	0	0	0	0	0	0	1	1
38	0	0	0	0	0	0	0	0	0	1
39	0	0	0	0	0	0	0	0	0	0

To illustrate the use of $\oplus$ we have in the case of the upper limit and test fields, 356727 and 247616 respectively, $356727 \oplus 247616 = 001000$, with the appearance of a 1 in the third position of the answer indicating the third digit of the test field failing to meet requirements. Thus an ADD operation followed by a BRANCH ON ZERO served to take the place of 40 COMPARE operations. Comparison with the lower limit field was accomplished similarly by using the test and lower limit fields, respectively, as the first and second operands of $\oplus$.

However, with the ADD table changed it is not possible to do normal address arithmetic. So the sequence followed was to change the ADD table, perform all the necessary tests without utilizing address arithmetic, and then to change the table back to normal form for tallying, sorting, etc. Enough memory was available to allow the 96 sets of comparisons to be written out in "straight line" fashion.

Upon timing, it turned out that all the comparisons could be done in approximately 0.5 seconds, well within the real time requirements of the problem.