diode, or rectifier, although other elements may be used. The computing unit of SEAC consists entirely of rectifiers; the few tubes in use in that unit are for amplification only, do not compute, do not change pulses into other patterns.

How do we make use of rectifiers for a function table, for transforming one set of pulses into another set of pulses, such as the case shown in Fig. 1 of the three logical circuit elements, an AND circuit, an OR circuit, and an EXCEPT circuit, the first two can be constructed readily with rectifiers, and the third can be constructed if the negative of a pulse is available. Suppose that we have two voltages, which will be designated 1 for the higher voltage and 0 for the lower voltage. Then an AND circuit is shown Fig. 2-a, an OR circuit Fig. 2-b.

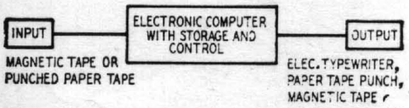


**Fig. 4—Computer input and output.**

In each of these circuits, there are four terminals. A common point in the circuit is connected across a resistance to terminal d, which in the AND circuit is kept uniformly at voltage 1, and in the OR circuit at voltage 0. Terminals *a* and *b* are input and *c* is output.

Now what happens? Examination of the AND circuit shows that the voltage at *c* will be 1 if and only if the voltage at *a* is 1 AND the voltage at *b* is 1. Also, in the OR circuit the voltage at *c* will be 1 if and only if either *a* or *b* or both *a* and *b* have the voltage 1. Hence, these circuits are properly AND circuits and OR circuits.

Using a rectifier there is no way of converting a pulse into its negative. But if the negative of a pulse is available—from a tube or otherwise—the AND circuit can be an EXCEPT circuit, simply by putting the negative pulse on one of the input lines.

## Coding conversion

One of the ways function tables can be used is for converting a decimal digit expressed in one kind of notation into a decimal digit expressed in another kind of notation. This is useful to do from time to time in an electronic computer because some kinds of operations are much easier in some notations than in others.

For example, one of the ways in which decimal digits can be represented is the following regular notation:

| Decimal | Binary 8 | 4 | 2 | 1 | Decimal | Binary 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 6 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 7 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 8 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 9 | 1 | 0 | 0 | 1 |

These binary columns have the value respectively of 8, 4, 2, and 1 (powers of 2); for example, 9 is one 8 plus one 1.

Another way decimal digits can be represented is in *biquinary notation* (like hands and fingers, or Roman numerals). In this notation the coding is:

| Decimal | Biquinary 00 | 5 | 0 | 1 | 2 | 3 | 4 | Decimal | Biquinary 00 | 5 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 7 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 8 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 9 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

These columns have the values 0, 5, 0, 1, 2, 3, 4. Seven, for example, is one 5 and one 2, 0100100. This notation was actually used in one of the big relay computers produced by Bell Telephone Laboratories, because the feature that two and only two pulses occurred in each row was useful for checking purposes.

Now, how do we convert biquinary notation into regular binary notation? This we can do with rectifiers in a function table. But how can we design the function table? That is easily done with one of the neat techniques of Boolean algebra (one of the algebras of symbolic logic), which has been alluded to from time to time in this series of articles. Here is the way. See Fig. 1 and Fig. 3.

Let's say that the A terminals are the 0, 1, 2, 3, 4 lines of the biquinary notation and the B terminals are the 00, 5 lines. These will be input. Let us say that the C terminals are the 1, 2, 4, 8 lines of the regular binary notation. These will be output. What conditions do we have to arrange?

Well, the C8 line is to have a pulse if and only if the B5 line has a pulse AND either the A3 line OR the A4 line has a pulse. Let us use "•" for AND and "v" for OR. Then we can write:

$$C8 = B5 \cdot (A3 \text{ v } A4)$$

where C8 stands for 1 if the C8 line has a pulse and 0 if the C8 line does not have a pulse (remember those truth-values?). We can also write down at once the other conditions:

$$C4 = (BO \cdot A4) \text{ v } B5 \cdot (AO \text{ v } A1 \text{ v } A2)$$
$$C2 = BO \cdot (A2 \text{ v } A3) \text{ v } B5 \cdot (A1 \text{ v } A2)$$
$$C1 = BO \cdot (A1 \text{ v } A3) \text{ v } B5 \cdot (AO \text{ v } A2 \text{ v } A4)$$

Every operation that appears in these equations is either an AND or an OR. So we can just make up a function table, connecting the lines with rectifiers, in just the fashion that the equations tell us to. The result appears in Fig. 3.

## Sample code

Now let's take a look at Fig. 3 and see how it works on a particular case. Suppose we want the number 7 in biquinary, 0100100, changed into the number 7 in binary, 0111. We use red numbers to stand for the values of the voltage, and put on each line in the circuit the value of the voltage which it will have. Line B5 and A2 will have the voltage 1. Consequently, vertical lines AO v A1 v A2, A2 v A3, A1 v A2, AO v A2 v A4, will have the voltage 1. Also the horizontal lines B5 • (AO v A1 v A2), B5 • (A1 v A2), B5 • (AO v A2 v A4) will have the voltage 1. As a result, lines C4, C2, C1 will have the voltage 1 and the C8 line will have the voltage 0. So the circuit works.

In the same way, other function-table circuits can be designed by simply writing down the conditions in convenient notation with AND and OR. If NOT or EXCEPT occurs, we use the negative of the pulse.

## Input and output

The relation of input and output to the rest of an electronic computer is shown schematically in Fig. 4. The choice of input for a small computer would be either magnetic tape or punched paper tape (for discussion of magnetic tape see article VIII). The choice for a large computer would be magnetic tape only; paper tape would be too slow. A mechanism feeds the tape past a reading device; the reading device converts the patterns of punched holes or magnetized spots, possibly through a function table, into appropriate patterns of pulses to be used or stored in the computer.

The choice of output for a small computer would be an electric typewriter, a paper tape perforator, or a magnetic tape recorder (one that could record discontinuous pulses, irrespective of continuous sound recording). The choice for a large computer would be only magnetic tape recording, for the others would be too slow. Some special large computers record also in a pattern of bright and dark spots on motion picture film, which is then developed. This of course is nonerasable storage.

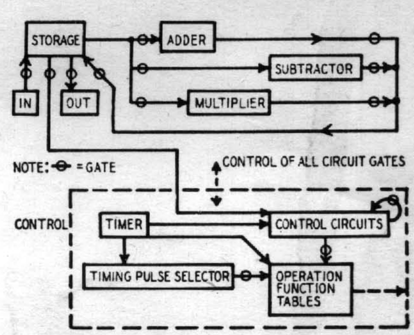


**Fig. 5—Block diagram of the computer.**

Both input and output give rise to problems of translation. For example, the number 7 expressed as 0111 in magnetized spots is to be translated into the number 7 expressed as 0111 in pulses circulating in a delay line. To carry out this translation, simply, cheaply, and reliably is an important engineering problem, although it does not appear as a problem at all in the logical design of the machine.

## A block diagram

We shall now redraw the block diagram of Fig. 4 in an expanded form, showing what may be called the "block diagram of an electronic computer." See Fig. 5. The "o" drawn on each of the flow lines indicates a gate (an AND circuit or an EXCEPT circuit), which may or may not be open allowing information or timing pulses to flow or not flow. The group of units together marked