RADIO — ELECTRONICS

K

LATEST IN
TELEVISION
SERVICING
AUDIO

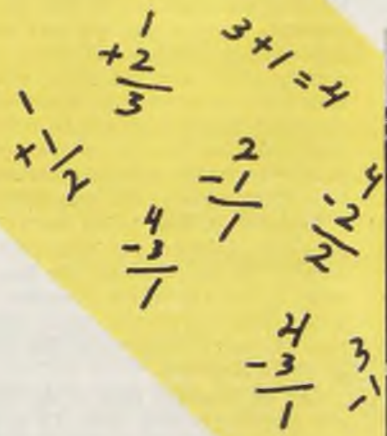
HUGO GERNSBACK, Editor

OCT
1950

30¢
U. S. and
CANADA

WORLD'S SMALLEST ELECTRIC BRAIN — SEE ELECTRONICS SECTION

World's Smallest Electric Brain



E. C. Berkeley explains how Simon gets instructions from a piece of punched tape.

By EDMUND C. BERKELEY*
and ROBERT A. JENSEN

Part I of a series of articles outlining principles and describing construction of electric and electronic computing devices

ON THE COVER of this issue of RADIO-ELECTRONICS is a picture of the smallest existing, complete electric brain. This midget electric brain is named Simon, in honor of Simple Simon of Mother Goose fame. He can be called electric or mechanical for he uses relays; but not electronic, for he does not use a single electron tube. Nevertheless he illustrates in solid hardware the principles of all the giant artificial brains, electronic, electric, or mechanical. He is perhaps the only electric brain small enough to be understood completely by one man.

Simon is about 24 inches long, 15 inches wide, and 6 inches high. He weighs (not counting his power supply) about 39 pounds. He runs on 24 volts d.c., drawing at most about 5 amperes. And in number mentality, Simon at present compares with a child of two years, for he knows only four numbers, 0, 1, 2, and 3.

Simon is slow. He performs each operation in about ⅔ second—unlike the electronic brain finished in 1949 called Binac, which adds at the rate of 3,500 additions per second. And yet Simon is a true mechanical brain, for he has the two essential properties that define a mechanical brain: he can transfer information automatically from any one of his 16 registers to any other, and he can perform endlessly long sequences of reasoning operations.

What is the purpose of this little idiot of an electric brain—or should he be looked on rather as a baby, with capacity to grow? Why was it worth while to build him?

The purpose of Simon

An editorial entitled "Simple Simon" in the *Wall Street Journal* for May 22, 1950, expressed in part the purpose of Simon: it said, "The world may admire a genius but it loves a moron." The same may perhaps be true of the crew of men who want to know how electric brains work, what they are all about, and how to construct them. It may be rather easier to understand the working of a little moron of an electric brain, that a student can easily feel superior to, than it is to understand the working of a giant electric brain, that a student can easily feel inferior to.

Simon was designed and built to exhibit in simple understandable form the essential principles of any artificial brain. He will be useful in lecturing, educating, training, and entertaining—just as a spinning toy gyroscope is both entertaining and instructive. For it is certainly true that the demand for computer-trained electronics engineers, operators, maintenance men, mathematicians, etc., is steadily growing in the new field of automatic computing machinery.

There are now more than a dozen kinds or species of these giant artificial brains. Most are represented by just one example, such as the rather old—but still spry—Harvard IBM automatic sequence-controlled calculator, finished in 1944. This machine handles numbers of 23 decimal digits and can remember 72 of them at one time. There are now in use more than 20 machines of the type known as the International Business Machines card-programmed calculator, and more than 80 of the type known as the Reeves electronic analogue computer.

Digital and analogue computers

These artificial brains are of two main types: *digital* and *analogue.* A digital machine expresses information by the positioning of devices in any one of a small number of exact positions. For example, a human hand with fingers up or down may express 0, 1, 2, 3,

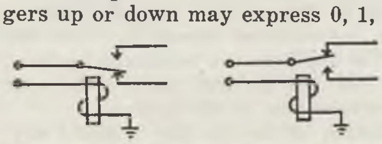


Fig. 1—Diagram showing a register of Simon expressing the information "1,0".

4, 5; or a counter wheel can stop at any one of the spots 0 to 9; or a light can be on or off, 2 positions; or a relay may be energized or not energized; or an electron tube may be conducting or not. All these devices are *digital.*

An analogue machine, on the other hand, expresses information as the measurement of a physical quantity, such as distance moved, or amount of rotation, or electric potential, etc. The

* Author: Giant Brains

measurement is *analogous* to a number in the computation.

But there is no easy way for an analogue machine: (1) to manipulate alphabetic information given in letters; (2) to express random numbers; (3) to express any numbers with an accuracy of more than 5 or 6 decimal digits; (4) to handle problems where the solution requires different decisions and subroutines, depending on what happens in the course of the problem. All these things a digital machine can do easily. Thus a digital machine can do rather more than an analogue machine. In fact it begins to look as if the digital machine of the future has within itself an unlimited capacity to think. This series of articles will deal mainly with digital electric brains.

How an electric brain works

How does an electric brain work? A good mental picture of the working of an electric brain is an isolated telegraph system, with a number of communicating central stations and a traffic control. The messages that this telegraph system handles are usually pieces of information of standard length, with a standard number of digits.

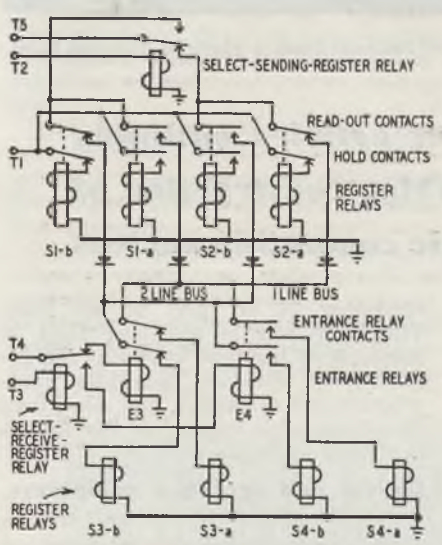


Fig. 2—A simplified schematic showing how Simon transfers information from either of two read-out storage registers to either of two read-in registers.

One of the stations is called INPUT. Here information comes in from the outside world to the telegraph system; it is put into a form ready to be sent somewhere else in the system.

Another of the stations is called OUTPUT. Here information that the telegraph system has produced is given back to the outside world.

There are a whole flock of stations called STORAGE No. 1, STORAGE No. 2, STORAGE No. 3, and so on. Here information may be stored without changing while waiting for some other part of the system to call for the information and do something more with it.

A very important station with room for several incoming pieces of information is the COMPUTER. This station is combined with a factory, a calculating device that can accept several pieces of information and manufacture new information out of them.

For example, the calculating device may have four receiving points or platforms. On two platforms, the computer takes in two numbers such as 140 and 25. On the third platform the computer takes in an order to subtract, multiply, or find which is bigger, etc. On the fourth platform the computer delivers a result (for example, 115), the result of combining 140 and 25 according to the order to subtract.

To calculate with this telegraph system, we must have some way of organizing traffic through it. That is the duty of the central traffic control. The most automatic way for sending information through the system is:

(1). At any one time connect just two telegraph stations, such as "Albany" and "Boston";

(2). Specify the direction of traffic, such as "from Albany to Boston." Then as soon as the proper connections have been completed, send the signal "go," and the information at Albany will be transferred automatically to Boston.

There are two ways to get the central traffic control to function properly. One is to have all the orders ready ahead of time, and tell it to do just as it is told. This is dictatorship. The second way is to have some special wires of the telegraph system run into the central control, and let information from time to time (though not all the time) come from the system into the central control—feedback. The central control then knows what is going on and can direct the following steps. This is democracy. This second technique of course is a honey, even with electric brains, and a good electric brain does compute some—or even most—of its own instructions.

Information

Such then is the mental picture of the working of an electric brain. But just what do we mean by information?

For the purposes of an electric brain, information is simply the arrangement of certain physical equipment. For example, a hand with two fingers up and three down is regularly considered to express the number two. Or suppose we take a pair of relays, a left-hand one and a right-hand one. Either one of these relays may be energized (let us report this condition as 1) or not energized (report this condition as 0). The information therefore that this pair of relays can represent is 00, 01, 10, and 11—four possibilities. (Here 10 is not ten, and 11 is not eleven). Let us number these four possible pieces of information 0, 1, 2, 3. Now we have the exact way a register of Simon expresses numbers (see Fig. 1).

Transfer of information

An electrical brain, like an automatic telegraph system, can transfer information automatically from one register to another. How does this take place?

Suppose we take some registers of Simon (a little simplified) and see how transfer does take place. Let us take two storage registers S1 and S2 (S for storage) from which we may read out information, and two more storage registers S3 and S4, into which we may read information. Each of these registers has two relays to supply the four possible pieces of information. Suppose we desire to transfer information from register S2 into register S3.

Looking at Fig. 2, we see 12 relays, of which eight are the relays for registers S1 to S4. We also see five terminals, T1 to T5, which energize the relays. The terminals are energized, that is, carry current, in the sequence of their numbers.

Let us consider time 1. At this time the circuit running from T1 to ground passes through both the closed HOLD contracts and the coils of (two out of) four relays S1-b, S1-a, S2-b, and S2-a. By a previous operation, the two relays S1-b and S2-a were energized and are now held up by continuous current from terminal T1. We see that information "1,0" is stored in register S1 and that information "0,1" is stored in register S2.

Let us pass to time 2, and look for terminal T2. At time 2 we see that the SELECT-SENDING-REGISTER relay is energized, and consequently register S2 is selected to send out its information.

Now let us pass to time 3, and look for terminal T3. The SELECT-RECEIVING-REGISTER relay, whose pickup coil is connected to T3, is not in this case energized. As a result, register S3 is selected to receive.

Passing to time 4, we look for terminal T4. As current flows along the wire from T4, the entrance relay for register S3 is energized. We have connected the pickup coils of register S3 to the bus, and therefore S3 can receive information from the bus.

We have now completed all the preparations needed to transfer information from register S2 to register S3. We now pass to time 5. Pulsing terminal 5, we see that the pulse of current flows as follows:

(1). through the selection circuit that selects the sending register S2;

(2). through the readout contacts of the sending register S2;

(3). through the rectifiers (which prevent back circuits);

(4). through the bus;

(5). through the contacts of the entrance relay belonging to the receiving register S3;

(6). into the coils of the receiving register S3 (naturally and properly, only the right-hand relay S3-a is energized, however); and

(7). to ground.

This then is an illustration of the principle of transferring information from one register to another. The scheme is entirely general: a pattern of information "written" in one register is "copied" in another.

How an Electric Brain Works

PART II

Simon, the little moron described last month, now learns how to add

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

IN the last article we observed how an electrical brain could store information in a register and transfer information from one register to another.

What do we mean by a "register" and "information"? A register is any physical equipment that can store information, and information is an arrangement of equipment that has meaning. For example, the registers we spoke of were relays, and the information stored in them was a pattern of 1's (relays energized) and 0's (relays not energized).

* *Author: Giant Brains*

Relays vs. tubes

Why use relays instead of tubes? After all, tubes can represent a pattern of 1's (conducting) or 0's (not conducting). And in changing its state from one condition to the other, a tube is very much faster than a relay.

We shall discuss tubes later in these articles. In the meanwhile, there are several reasons for concerning ourselves with relays. In the development of electrical brains, relays were used before electronic tubes; and it is some help in explaining electrical brains to follow the course of their development. In the second place, a circuit involving relays can be understood simply by seeing whether current will flow or not. But understanding a tube circuit involves many more factors, such as plate voltages, nature and strength of signal, the tube characteristics, values of the resistances, etc. In the third place, one relay may take the place of a number of tubes. For example, a six-pole, double-throw relay has 20 terminals including those of the pickup coil; a considerable number of tubes is required to do as much as that one relay. And finally, after the logical relations of circuit elements have been expressed once in any one form such as relays, it is easier to translate them into other circuit elements such as tubes.

We have seen how we can store information in a register and transfer it from one register to another, and now we wish to manipulate that information according to mathematical and logical processes. That is calculating. And the first of the calculating processes is addition. How can we add and calculate with electrical circuits?

Calculation is possible only with a good system for representing information. For example, the decimal notation using Arabic numerals, considered one of the great human inventions, is a remarkably fine system for representing numerical data. As a counterexample, the ancient Greeks used letters of the alphabet in a rather unsystematic way for numbers, and never got far with arithmetic. To calculate with electrical circuits we first must understand systems for representing information.

Representing information

Suppose we write down a piece of information such as 1,011. We are referring to a number which is equal to one 1,000, zero 100, one 10, and one 1. The positions of the digits report *powers* of 10—1; 10; 100; 1,000; 10,000; etc. Numbers written in this style are said to be in *decimal notation.* This notation is convenient for a calculating machine that contains dials that may take one of ten positions, like the speedometer of an automobile.

But this notation, convenient as it may be for human beings and some machines, may actually be inconvenient for most digital electrical brains containing two-position circuit elements.

Numbers may be written in other more useful notations. Two of them

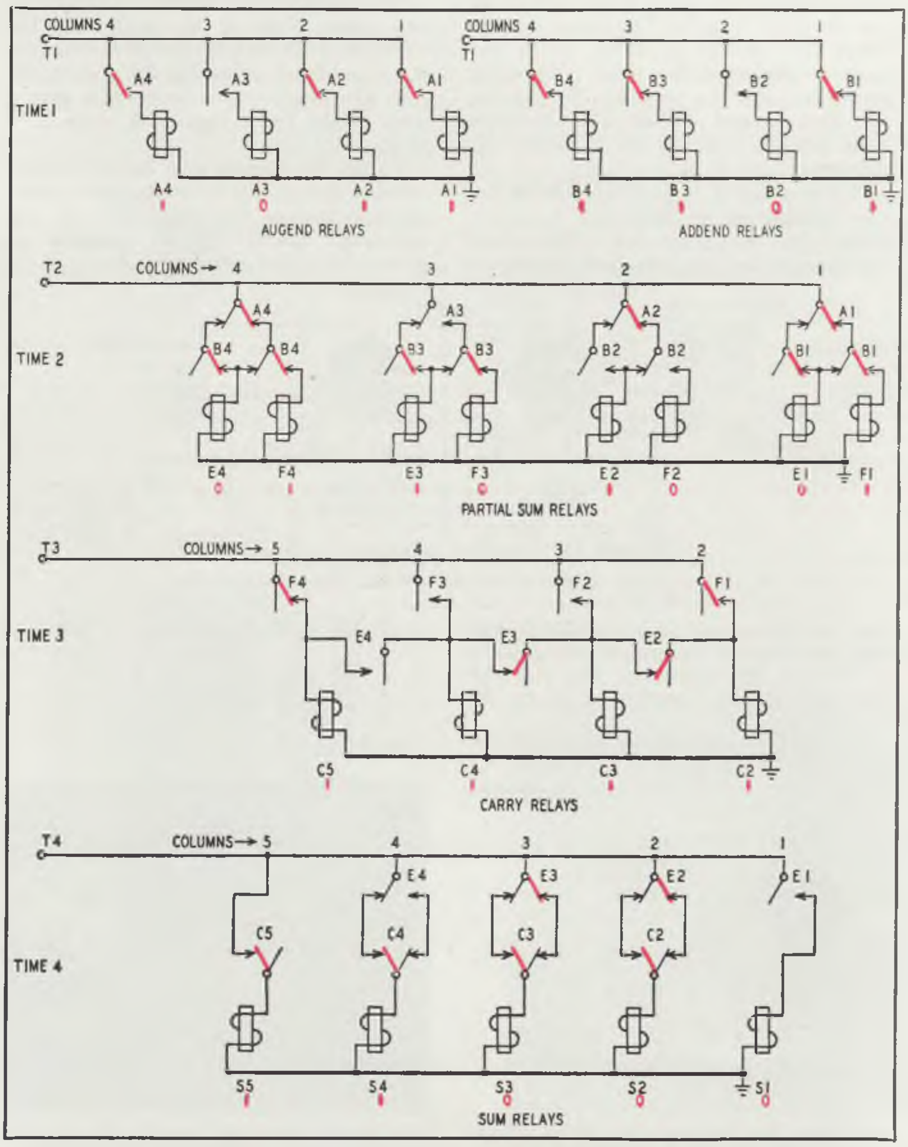


Fig. 1—A binary adding circuit that is capable of adding two 4-column numbers.

that are interesting and useful are the notations called *biquinary* (pronounced "by-kwy′nerry") and *binary* pronounced "by′-nerry").

Biquinary means counting by 2's and 5's; the "bi" refers to two and the "qui" to five. It is the notation expressed in a limited way by the hands and feet of human beings, Roman numerals in ancient style, and the earliest of all calculating machines, the *abacus* or bread-counting frame. For example, in writing the number CCLXXXVIIII (two 100's plus one 50 plus three 10's plus one 5 plus four 1's), we are using biquinary notation. The more recent Roman numerals used IX in place of VIIII but this later style is not biquinary.

Two of the big electrical brains that have been finished by Bell Telephone Laboratories use the biquinary system. Seven relays are used for each decimal digit in the pattern shown in Table I.

Table I—Biquinary Relay System

Decimal Digit	Name of Relay: 00	5	0	1	2	3	4
0	1	0	1	0	0	0	0
1	1	0	0	1	0	0	0
2	1	0	0	0	1	0	0
3	1	0	0	0	0	1	0
4	1	0	0	0	0	0	1
5	0	1	1	0	0	0	0
6	0	1	0	1	0	0	0
7	0	1	0	0	1	0	0
8	0	1	0	0	0	1	0
9	0	1	0	0	0	0	1

1 means "relay energized"; 0 means relay "not energized.") The first two relays are called 00 and 5 and the others 0, 1, 2, 3, and 4. To represent any digit, either 00 *or* 5 must be energized, and exactly one of the other five relays must be energized. If either of these conditions fails, then the machine gives an alarm and stops at once. When the operator examines the trouble-light panel, he detects the circuit that failed. The machine acts as its own policeman. This machine in months of operation has not allowed a single wrong result, except those due to human errors in instructing it.

There are still simpler notations for representing information in circuit elements having just two positions, such as relays or tubes. One of these is binary notation and its relatives. In pure binary notation, the *base* is not 10 as in decimal or alternately 2 and 5 as in biquinary. The base is 2, and the digits —which are only 0 and 1—report the powers of 2: 1, 2, 4, 8, 16, etc. Numbers 1, 2, 4, 8, etc., are represented by 1, 10, 100, 1,000, etc. Fractions ½, ¼, ⅛, $\frac{1}{16}$, etc., are represented by 0.1, .01, .001, .0001, etc. For example, the figure 1,011 now means one 8 plus zero 4's plus one 2 plus one 1, and so equals 11.

Other samples of equivalents of decimal and binary numbers shown in Table II. The last binary number in the table is an example of a "recurring decimal" (or should we call it a *recurring binal?*) in binary notation.

Incidentally, if you, like the author, prefer to pronounce new words and signs under your breath, the binary numbers 10, 11, 1,000, and so on, should be pronounced "one-oh, one-one, one-oh-oh-oh" and so on. This way of referring to them gets away from confusion with decimal numbers.

Table II—The Binary Numbers

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
8	1,000
16	10,000
17	10,001
32	100,000
0.5	0.1
0.25	0.01
0.125	0.001
1/3 or 0.33333....	0.01010101
or $0.\dot{3}$	or $0.\dot{0}\dot{1}$

As shown by these examples, we can convert numbers completely from the decimal system into the binary system. Some electronic brains do just that, for so doing it simplifies the circuits of an electronic machine. For example, the electronic brain known as Binac (Binary Automatic Computer), completed in 1949, handles all numbers in pure binary notation.

Adding in Binary Notation

Before we proceed to other forms of notation, let us consider how adding in binary notation is carried out—on paper, and by a relay circuit.

Adding in binary is carried out by this addition table:

	0	1
0	0	1
1	1	10

To add with this table, select one of the numbers you wish to add from the top outside row and the other from the left outside column. Draw a vertical line (imaginary if you wish) from the number selected in the top row down through the column of numbers on the inside, and a horizontal line from the number in the left column across the row of numbers on the inside. You will find the sum of the two numbers where the two lines cross. Thus 0+0=0, 0+1=1, 1+0=1, and 1+1=10, (i.e., 2).

No one could ask for a simpler addition table! Instead of the 100 entries of the decimal addition table, the binary addition table has only four.

For an example, let us add 1,011 (8 plus 2 plus 1, or 11) and 1,101 (8 plus 4 plus 1, or thirteen):

```
 1,011
 1,101
------
11,000
```

The result is 11,000 (16 plus 8, or 24). We may obtain it by the following "schoolboy" routine: (column 1, starting from the right) 1 and 1 is 10, put down 0, and carry 1; (column 2) 1 and 0 is 1, and 1 to carry, is 10, put down 0, and carry 1; (column 3) 0 and 1 is 1, and 1 to carry, gives 10, put down 0 and carry 1; (column 4) 1 and 1 is 10, and 1 to carry is 11, put down 1 and carry 1; (column 5) put down 1.

Binary adding circuits

Now let us design a set of circuits which will carry out the operation of addition in binary notation. We shall begin with a preliminary circuit, preliminary because it uses many more relays and more successive impulses than are necessary. But it is useful because it shows the steps needed to reach a final circuit.

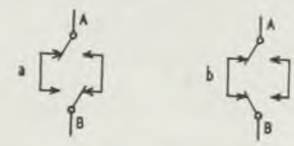


Fig. 3—A and B contact patterns that are used to simplify the adding circuit.

First, two binary numbers to be added, each of four binary digits, are stored in the two registers A and B (called Augend "to be increased" and Addend "to be added"). The relays marked A1 to A4 store the digits in the digit columns 1 to 4, counting from the right—first the one digit, then the twos, the fours, and the eights digits. (See time 1 of Fig. 1). At some previous time these relays have been picked up, and now terminal T1, through the hold contacts of these relays, holds them up while the later circuits operate. For example, suppose 1,011 is stored in the A or Augend register. Then relays A4. A2, A1 are

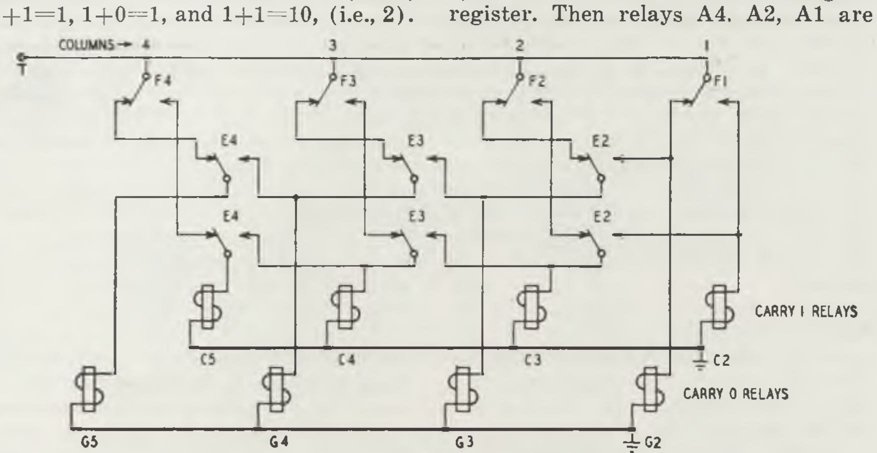


Fig. 2—A 4-column binary carry circuit that will carry 0 as well as carry 1.

energized and relay A3 is not energized. Suppose 1,101 is stored in the B or Addend register. Then relays B4, B3, B1 are energized, and relay B2 is not energized. In the diagram the relay contacts that will be closed for this particular state are shown in red. The energized coils are marked with a 1 and the unenergized coils with a 0.

Let us stop for a moment to explain the convention used in the figures. The coils of relays are *not* drawn next to their corresponding contacts, as is common in radio work, but follow a practice common in electrical diagrams: The contacts of a relay are labeled with the label of the coil. For example, in time 2 of Fig. 1 more contacts of the A and B relays of time 1 appear, all drawn in the unenergized state. To find which of the partial sum (E,F) relays will be energized corresponding to A 1,011 and B 1,101, follow the red contacts.

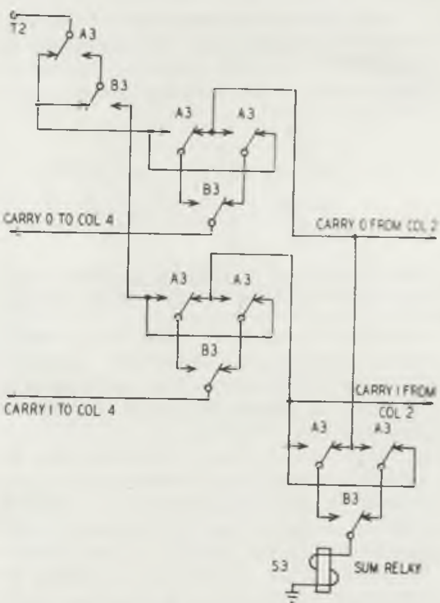


Fig. 4—The final adding circuit after simplification. The four parts of Fig. 1 are all combined in this one circuit.

Returning now to the discussion of the circuit of Fig. 1, at the second time (time 2) terminal T2 is energized, and the E and F relays are picked up and held. The E relay is picked up if there is just a 1 and a 0 in the digit column, and the F relay is picked up if there are two 1's in the column, causing the sum (without carry) to be 10. In other words, the F relay of a column signals that a carry originates. The E relay of a column reports two facts: (1) the sum of the digits without carry in that column is 1; (2) if there is a carry into this column, it should be transmitted into the next column.

Third, at time 3, terminal T3 is energized, and the C relays (C for carry) are energized. Contacts from the F relays report the originating of a carry, and contacts from the E relays do the transmitting of the carry.

Fourth, at time 4, terminal T4 is energized, and the binary sum is recorded in the S (S for sum) relays. Any sum relay will be energized under two conditions: (1) the sum without carry in a column is 0, and there is a carry of 1 arriving in that column; (2) the sum without carry in a column is 1, and there is no carry of 1 arriving in that column.

Some circuit tracing

To test the circuit of Fig. 1, let us follow through the addition of the first two columns of the two numbers we are adding (1,011 and 1,101). These two columns will have 11 as the augend and 01 as the addend. To begin with, A1 and A2 are both energized to indicate 11; and B1 is energized and B2 is not energized to indicate 01. When terminal T2 is energized, F1 is picked up to indicate that the sum of the first column is $1+1=10$ and that the 1 in this sum must be carried to the next column. E2 is picked up to indicate that the sum of the second column is $1+0=1$. So far we have added the two columns separately and have indicated where numbers are to be carried.

In the next step terminal T3 is energized. The carry from column 1 now takes place and C2 is picked up because contact F1 is closed. Adding the carry from column 1 to the 1 already in column 2 makes the sum of column 2 equal to 10 and the 1 of this sum must be carried to the third column. This carry takes place because E2 is closed to indicate a 1 already present in the second column, and the new carry is transferred to column 3.

In the final step T4 is energized. Now S1 is not picked up because E1 was not picked up when T2 was energized. S2 is also not picked up because E2 was energized, but so was C2 when we had to carry the 1 from column 1. So the first two columns of our addition will be 00, which we know checks with the sum of the two numbers we started with. By following through the circuit we can see that the rest of the circuit operates in exactly the same way.

A simplified circuit

The group of circuits in Fig. 1 can be considerably improved and condensed into a single circuit if we introduce the concept "carry 0" as well as "carry 1". In general, we carry 0 into a column if, and only if, the preceding column contains either both digits 0 or just a 1 and a 0 and is not receiving a carry itself. A circuit that will establish both carry-0 and carry-1 conditions is shown in Fig. 2. This circuit is the same as part 3 of Fig. 1 except that both make and break contacts of the E and F relays are used and an extra set of E relay contacts is needed to perform the carry-0 operation. The G relays record the condition of carry 0.

We can now replace the F relay make contacts with a pair of A and B relay make contacts in series. When this is done, the C relays are picked up only when both A and B relays are energized, which is the condition we need to indicate a carry.

We further simplify by replacing the E make contacts by the pattern of A and B contacts shown in Fig. 3-a and the E break contacts by the arrangement of Fig. 3-b. Now the G relay is energized through the A and B contacts only for the carry-0 condition and the C relays for the carry 1.

Instead of allowing the lines to lead down to the G and C relay coils we insert the E and C contacts of part 4 of Fig. 1, but immediately replace these with a pattern of A and B contacts. The circuit then looks like Fig. 4, which shows the relays only for column 3. (Note that the adjacent B contacts are combined in a single contact.)

There are four conditions when the S3 relay must be energized:

1. A3=0, B3=0, carry 1 in;
2. A3=1, B3=0, carry 0 in;
3. A3=0, B3=1, carry 0 in;
4. A3=1, B3=1, carry 1 in.

For condition 1, the S3 relay is picked up directly through the carry-1 line from column 2 and the carry 0 is applied to column 4. For conditions 2 and 3, S3 is energized directly by the carry 0 from column 2 and carry 0 is also fed to column 4. For condition 4, S3 is operated by the carry 1 coming in, and a carry 1 is fed to column 4. Under these conditions, but no others, will the S3 relay be energized. By similar reasoning, we can list the condition for which we want S3 to remain unenergized and whether we want a carry 0 or carry 1 to go to the next column. Then with a little more tracing we will see that the circuit of Fig. 4 will meet all these conditions. Doubtless this circuit also can be improved and simplified.

ENGINEERS WANTED

Electronics engineers specialized in the communications field are urgently needed to fill jobs in the Signal Corps research and development laboratories at Fort Monmouth, N. J. Salaries range from $3825 to $5400 a year.

Also in critical demand are experienced technical writers and training instructors with extensive electronics or communications equipment background at salaries from $3825 to $4600.

To qualify for the basic electronic engineering positions, applicants must have a Bachelor's degree from a recognized college or equivalent education or experience plus one year of professional electronic engineering experience or study for a Master's degree.

Technical writers must have at least 3½ years experience in preparing reports, manuscripts or manuals concerned with electronics, radio, radar or communications. Additional experience is required for the higher grades.

Four years of technical experience in such fields as microwave radio relay, radar, radio electronics and central office techniques, plus one year of teaching experience is needed to qualify for the training instructor jobs.

These jobs may be applied for by mail or in person at the Civilian Personnel Branch, Building 530, Fort Monmouth, N. J.

Relays Do Simple Arithmetic

Part III—How to use relay adding circuits for subtraction and multiplication in the binary system

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

IN THE two earlier articles we have seen how an electric brain can:

1. store information in a register;
2. transfer information from one register to another; and
3. add two numbers expressed in binary notation (the scale of two).

Being interested in constructing a relay calculator, in this article we shall consider subtraction and multiplication using relays.

We shall keep to binary numbers for the present for three reasons: It is easy to carry out the operations we are interested in. Also, binary notation is good for electron-tube calculating circuits as well as relay calculating circuits. Finally, it is a good introduction to the circuits needed for calculating in the decimal scale.

Suppose we wish to subtract the binary number 101 (read "one-oh-one," meaning one 4 plus no 2's plus 1, or 5) from the binary number 1110 (read "one-one-one-oh," and meaning one 8 plus one 4 plus one 2 plus no 1's, or 14). We write 101 under 1110 and subtract:

```
 1110
  101
 ----
 1001
```

How do we manage to subtract? We recall the binary addition table:

	0	1
0	0	1
1	1	10

Then we say under our breath, for the first column at the right: "1 from 0 does not go, borrow 1; 1 from 10 (read "one-oh" not "ten") is 1, write down 1." For the next column, we say, remembering the borrow: "0 from 0, write down 0." For the third column: "1 from 1 is 0, write down 0." For the last column: "nothing from 1 is 1, write down 1." The result is 1001 (read "one-oh-oh-one," meaning one 8 plus no 4's plus no 2's plus one 1, or 9) just as we would expect it to be.

We could set to work and design a circuit which would reproduce this process and give the precise result we desire. But isn't there an easier way?

There is an easier way to subtract—by using the addition circuit shown in the last article, and using the mathematical fact that subtracting a number is the same as adding the complement.

To make the idea of complement clear, let us return for a moment to decimal notation (the scale of 10) and consider a desk adding machine having just five columns. Suppose we consider a number 864 (eight 100's plus six 10's plus four 1's). Suppose we set the machine at 0 and subtract 864. We will obtain 99136. This is called the *complement* of 864 (also called the *tens complement* of 864). For, if we take 864 and 99136, and add them, we get 100,000; but the extreme left-hand digit (the 1) being beyond the capacity of the five-column adding machine, it vanishes and the result is 00,000 or zero. (In a machine of ten columns instead of five the complement would be 9,999,999,136, correspondingly.)

Now to subtract 864 from any number—suppose it is 3,145—we add the complement:

```
 3145        3145
- 864     + 99136
 ----      ------
 2281      102281
```

The extreme left-hand digit in 102,281 will disappear off the machine, giving 2,281 as the result, which is correct.

The complement (such as 99,136) is easily found for any number (such as 00,864) by two rules:

1. take each digit away from 9 (obtaining what is called the *nines complement*, in this case 99,135);
2. add 1 to the result (obtaining in this case 99,136, the *tens complement*).

What is the analogue in binary notation to these complements in decimal notation? In decimal notation we have a *nines complement*, nine being one less than ten, the base of the scale of notation; so, in binary notation, we shall have a *ones complement*, since one is one less than two, two being the base

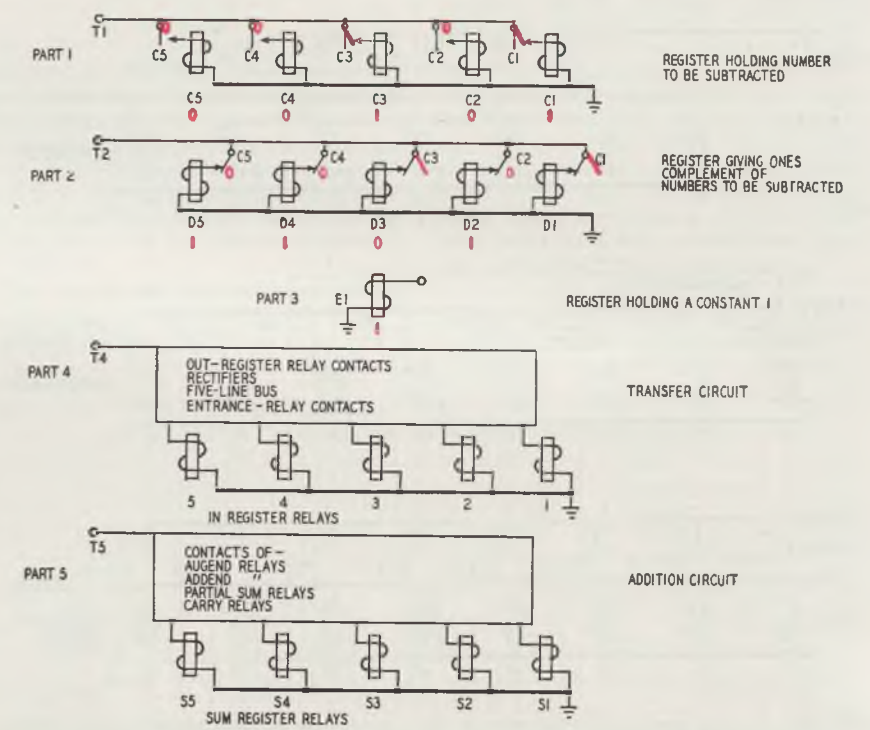


Fig. 1—A relay circuit that subtracts by first obtaining the twos complement.

* Author: Giant Brains

of the scale of notation. We obtain the *ones complement* of a binary number by taking each of its digits away from 1. The *twos complement* is then equal to the ones complement plus 1.

So to subtract in binary:

1. find the *ones complement* of the number, by taking each digit away from 1;
2. obtain the *twos complement* by adding 1 to the ones complement;
3. add the *twos complement*, and discard the extreme digit on the left.

Suppose we do this arithmetically first, and then examine a series of circuits which will produce the same result. Let us return to the example of subtracting 101 from 1110. Let us assume that we have a five-column binary calculator. The *ones complement* of 101, then, is 11010; adding 1, we get the twos complement, 11011. We add:

```
   1110
+ 11011
-------
 101001
```

The extreme left-hand digit 1 vanishes ("goes off the keyboard") and the result is 1001, the same result as before, as would be expected.

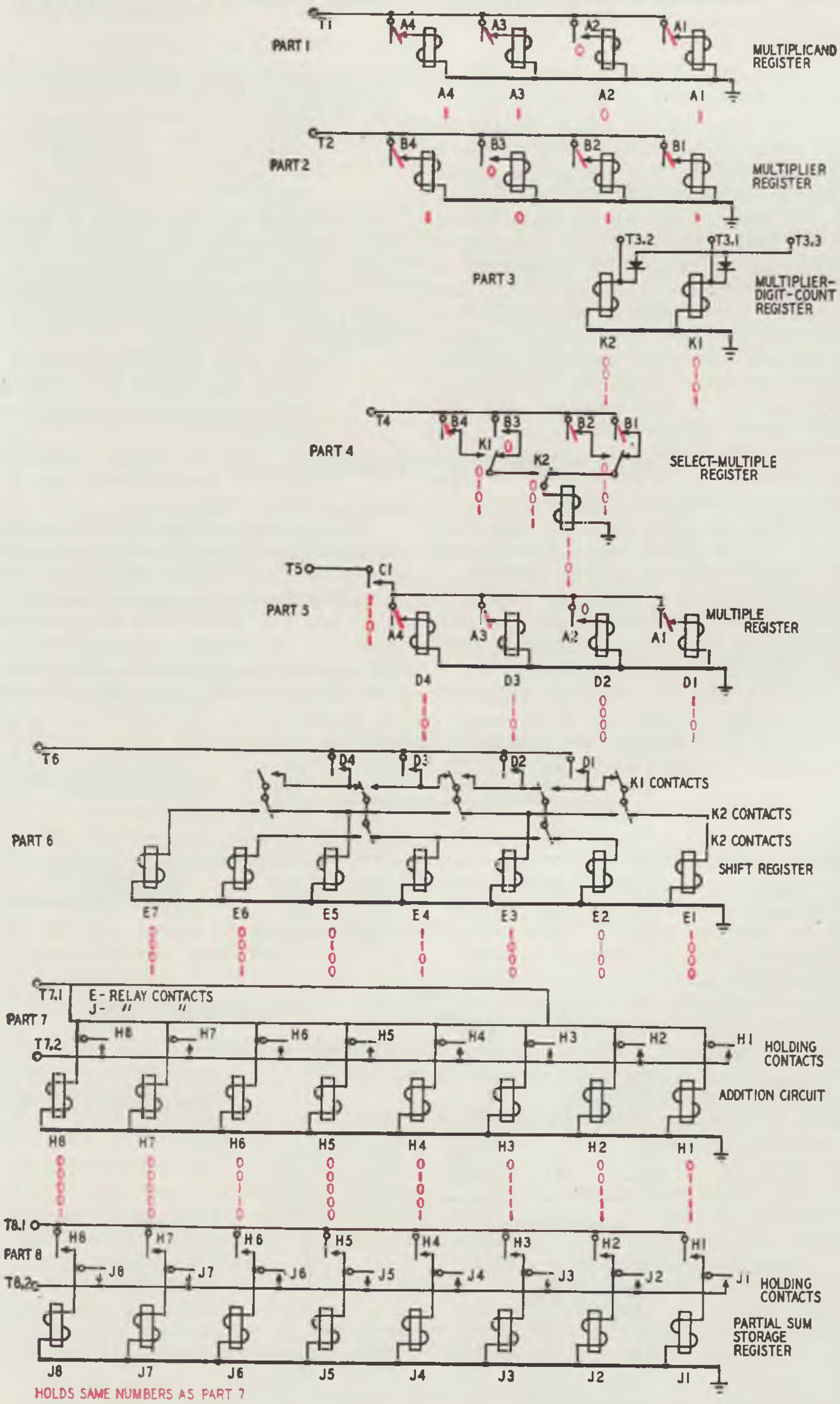


Fig. 2—A relay multiplication circuit. It is really an adding circuit which shifts successive multiples of the multiplicand to the left and adds them.

Subtraction circuits

A series of circuits for subtraction is shown in Fig. 1. First (see part 1), the number to be subtracted is stored in the C register in relays C5 to C1. The example shows 101 stored (or 00101) where the first 0 tells us the number is positive. The C3 and C1 relays only are energized as shown in red. At Time 1, terminal T1 holds these relays energized through their hold contacts.

At Time 2 (see part 2), the ones complement of the C number is obtained, by reading out from terminal T2 through normally closed contacts of the C relays, into the D relays. Hence, the number stored in the D relays is 11010, precisely the ones complement of 00101.

All the rest of the calculation is "reduced to the previous case," as the mathematicians say: reduced to transfers and additions, which have been described in earlier articles but are as follows:

The D number—the ones complement —(see part 2) is transferred via the transfer circuits of part 4 into the augend relays of the addition circuits (see part 5). A constant 1 stored in the relay E1 (see part 3) is transferred (see part 4) into the addend relays of the addition circuits (see part 5). The sum obtained in the sum register, the S relays (see part 5), is routed back via the transfer circuits of part 4 (and perhaps via an extra temporary storage register) into the addend relays of the addition circuits. So at this point we have the twos complement of the number to be subtracted, stored in the addend of the addition circuit.

Next, the number to be diminished is transferred from whatever register it is stored in via the transfer circuits of part 4, into the augend relays (see part 5) of the addition circuit.

Then finally we pulse the addition circuit a second time, and the result of the subtraction is produced in the sum register of the addition circuit.

Note that the transfer circuits of part 4 are here assumed to transfer a five-digit binary number from a five-relay register via a five-line bus into another five-relay register. The transfer circuits of Article I of this series show only two-relay registers and a two-line bus; but the extension should give no difficulty.

Plus and minus numbers

At this point we start thinking again and say to ourselves, "It is foolish to have a calculator that can only handle positive numbers. Our calculator should handle both positive and negative numbers. How shall we arrange that?"

A good answer, though not the only one, is to agree that the extreme left-hand digit of the number will tell the sign of the number, whether *plus* or *minus*. For example, in decimal notation, with a five-column calculator, we would stop using the fifth column from the right for showing digits. Instead we would say: "If it holds 0, the other four digits are a positive number; if it holds 9, the other four digits are the complement of a negative number." For example, 09136 would mean +9136, but 99136 would mean –864. As a result, the machine would be unable to express any number greater than +9999 or less than –9999.

In binary notation we can do almost the same thing. We say, "If the extreme left-hand digit is 0, the remaining digits make a positive number. If that digit is 1, the remaining digits are the complement of a negative number."

We need to adjust our calculating circuits for adding two numbers which are both positive, or both negative, or one positive and one negative. We also need to adjust our calculator to ring an alarm in cases where the result is beyond the capacity of the calculator—that is, beyond +9999 or –9999 in terms of the example given above. A third consideration is the decimal point or, in binary notation, the "binal" point. In fact, there are a number of little adjustments needed. But it is probably better to neglect them at this stage, and go on to the next main process, multiplication.

Multiplication

In binary notation, the multiplication table becomes simply:

	0	1
0	0	0
1	0	1

or, in other words, 0 times 0 is 0, 0 times 1 is 0, and 1 times 1 is 1. Multiplication becomes either adding or not adding, and shifting.

For example, let us multiply two binary numbers, 1101 (one-one-oh-one: 8 plus 4 plus 1, or 13) and 1011 (one-oh-one-one: 8 plus 2 plus 1, or 11):

```
    1101
    1011
    ----
    1101
   1101
  0000
 1101
 --------
10001111
```

The result is 10001111 (one-oh-oh-oh-one-one-one-one, or, one 128 plus no 64's plus no 32's plus no 16's plus one 8 plus one 4 plus one 2 plus one 1, or 143), which is of course what we would expect from ordinary multiplication of 13 and 11.

Fig. 2 shows circuits with energized relay contacts in red. Fig. 3 is the timing chart, showing how the circuits are to operate one after another, and over again, for successive digits of the multiplier. These circuits are preliminary, and not final.

We have assumed that the multiplicand (the number to be multiplied) is the four-digit binary number 1101, and the multiplier is the four-digit binary number 1011. They are stored in the A register (see part 1) and the B register (see part 2).

The general method we have used for obtaining their product is: choose multiples of the multiplicand, either 1101 or 0000, according to the successive digits of the multiplier 1011 taken from right to left 1, 1, 0, 1 (see parts 3, 4, 5); shift these multiples over to the left (see part 6) according to the successive positions of the multiplier digits (0, 1, 2, 3, or in relay language, 00, 01, 10, 11); with an addition circuit and storage register, add successively the shifted multiples (see parts 7, 8).

Proceeding now to examination of the parts of the circuit in detail, let us begin with a look at part 4. The select-multiple circuit (controlling relay C1 in part 4) operates on the multiplier 1011. This circuit yields at different times the successive digits 1, 1, 0, 1 that select the multiple of the multiplicand. How is this made to happen? In part 3, the K relays are energized in the pattern 00, 01, 10, 11, (0, 1, 2, 3 in binary) at successive times.

In part 5, the multiple of the multiplicand is selected. Its successive values are 1101; 1101; 0000; 1101. The contacts used are a contact of the C relay (see part 4) and four contacts of the A relays (see part 1); the multiple is recorded in the D relays.

In part 6, the selected multiple, recorded in the D relays, is shifted 0, 1, 2, or 3 spaces over, according to the position of the multiplier digit, recorded in the K relays (see part 3). The numbers produced by the E relays accordingly are:

0 0 0 1 1 0 1,
0 0 1 1 0 1 0,
0 0 0 0 0 0 0, and
1 1 0 1 0 0 0.

In Part 7, the addition circuit is indicated in block diagram, since it was treated in full in Part II, of this series. The storage register (part 8) is needed to transfer the result of the addition from the output of the addition circuit back into one of its inputs. Using these two circuits, the shifted multiples are added one after another as indicated in Table I.

The timing of the circuits is shown in the timing chart of Fig. 3. A good deal of useful information is summarized in this chart.

Successive time intervals, 1, 2, 3, 4, etc., are shown from left to right. The different terminals are shown from top to bottom, together with the relays they energize.

Opposite each terminal, the horizontal line begins at the time when the terminal is energized and stops when the terminal ceases to be energized. For example, the three terminals that energize the K relays T3.1, T3.2, and T3.3 are energized from time 6 to 9, from time 11 to time 14, and from time 16 to time 19, respectively. The following parts then complete the multiplication in four similar cycles.

Sections of different horizontal lines are connected by vertical lines showing X's and O's. These vertical lines with their marks summarize the functional relation of circuits. X marks the relays that are energized at a certain time, and O's mark the relay contacts through which these relays are energized. For example, at times 3, 8, 13,

Table I — Multiplication Sequence

Name	Number	Relays
Partial sum	0000000	H to J
1st multiple	0001101	D
New partial sum	0001101	H to J
2nd multiple	0011010	D
New partial sum	0100111	H to J
3rd multiple	0000000	D
New partial sum	0100111	H to J
4th multiple	1101000	D
Final sum	10001111	H to J

and 18, the D relays of circuit 5 are energized by current flowing from terminal T5 through contacts of the A relays and the C relay. For another example, at time 2, terminal T4 is energized, and the C relay is picked up, reading through B relay and K relay contacts. The K contacts at this time have not been energized; but this is correct, because the first multiplier digit has the position 0.

It should be emphasized once more that there are many ways of condensing and improving these circuits. For example, parts 4, 5, and 6 can be combined, and the C and D relays eliminated. But the resulting circuit would have been harder to understand than the separate circuits here shown.

(To be continued)

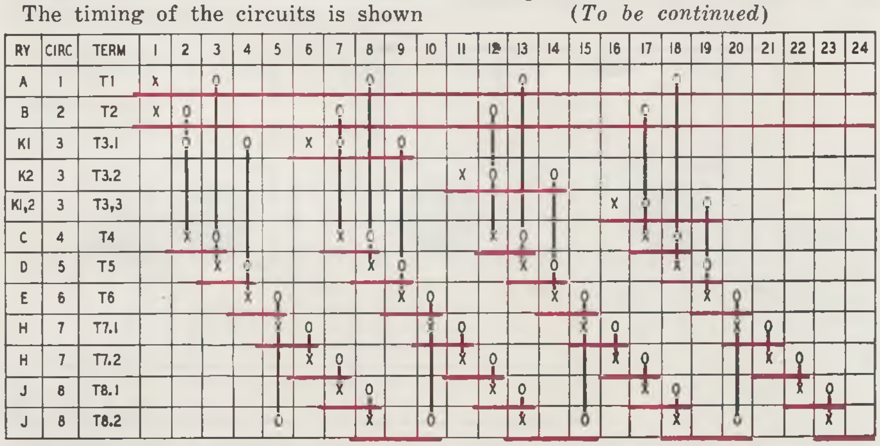


Fig. 3—Multiplying circuit timing chart which shows the sequence of operation.

How an Electric Brain Works

Part IV—Long division with relays—our little electric brain learns how to divide and to convert decimal numbers to binary and back again. Simon is getting an education

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

PREVIOUS articles of this series have shown how an electric brain made of relays can add, subtract, and multiply.

Now we shall carry out division. As before, we shall consider the process in *binary notation,* the scale of two.

As a second topic, we shall consider how to make a relay calculator convert a number from decimal notation to binary notation, and back again. There is every reason in the world why the machine itself should convert any decimal number, say 23, into the corresponding binary number (in this case 10111, one-oh-one-one-one, or one 16 plus no 8's plus one 4 plus one 2 plus one 1).

Addition, subtraction, and multiplication turned out to be very simple in binary notation as compared with decimal. The same is true with division: binary division is simple as can be.

Suppose we divide 1101 (one-one-oh-one, or 8 plus 4 plus 1, or 13 in decimal) into 10000101 (one-oh-oh-oh-oh-one-oh-one, or 128 plus 4 plus 1, or 133.

We do this in the same general way as we do in decimal division, except that we act as if we knew only the two digits 1 and 0:

```
                  01010     (Quotient)
(Divisor)  1101)10000101    (Dividend)
                0000
                10000 (1st  Partial
                 1101       Remainder)
                  0111 (2nd Partial
                  0000      Remainder)
                   1110 (3rd Partial
                   1101     Remainder)
                    0011 (4th Partial
                    0000 Remainder)
                     011 (Remainder)
```

Only two multiples of the divisor are used, one times the divisor, and zero times the divisor—and the latter is of course zero in every digit. No other multiples of the divisor are needed. If we simply compare the divisor with the partial remainder at any point in the division, we can tell whether the digit of the quotient is 1 or 0.

Circuits for division

As before, to keep the circuits simple, let us ignore a number of fine points, such as: fractions; the *binal* point (the analogue in the scale of two of the decimal point in the scale of ten); positive and negative numbers; size of numbers; etc. Suppose that we have an eight binary digit dividend, and a four binary digit divisor.

The circuit is on the opposite page. In part 1, terminal T1 is energized at the start, and holds up the relays storing the dividend through their hold contacts. (All current-carrying circuits and relay contacts in the energized state are in red.) The actual number which these relays store, of course, depends on something that happened before the time at which we begin. In the same way, the divisor is stored in relays of part 2 of the circuit, and terminal T2 holds them up.

Now different things have to happen at different stages during the division. So we want to have some relays that will tell us at what stage we are during the process of the division. This is the function of the K relays of part 3 of the circuit. The stages that they detect and report are 0,1,2,3,4. The time chart in Fig. 1 shows that stage 0 lasts from time 1 to time 8, stage 1 from times 9 to 16, stage 2 from times 17 to 24, etc. At stage 0, we attend to the first quotient digit; at stage 1, we attend to the second quotient digit; etc. The red parts of the circuit apply to the first stage of the division only.

We have to start off the divisions by selecting some digits, which we can call a *partial remainder* (see part 4). At stage 0, this is the first four digits of the dividend; but at later stages this is the result of a subtraction together with "bringing down" one more digit of the dividend. The circuit of part 4 shows that at each stage of the division, we have just the partial remainder that we desire stored in the E relays. We have to look ahead to part 8, of course, and take on faith that the G relay contacts in part 4 will express the result of a subtraction that we want.

The next thing that we must do is decide whether the divisor "goes" into the partial remainder, or whether it "doesn't go". To make this decision, we must compare two numbers and decide which is the larger. The divisor "goes" and yields 1 as a digit of the quotient if, and only if, the partial remainder is larger. A circuit that does exactly this is shown in part 5. The red contacts show the original partial remainder (stored in the E relays) and the divisor (A relays). We see that there is no path for the quotient relay Q to be energized, and so the first "quotient digit" is 0.

Before we go any further, we want to store that quotient digit, so that we shall know the whole quotient when we get through with the division. This duty is performed by the circuit of part 6, which shows how the digit quotient is routed, according to the time it is obtained, into the right C relay.

We now want to determine the multiple of the divisor that depends on the quotient digit and the divisor. This is the function of part 7 of the circuit, which will give us the divisor itself if the quotient digit is one and zero in all digits if the quotient digit is zero.

In part 8 of the dividing circuit, the subtraction of the divisor multiple from the partial remainder is indicated schematically, because actual circuits for subtraction were discussed previously.

The timing of the circuits, up to the end of the first two quotient digits, is shown in the timing chart of Fig. 1. The same conventions are used here as in the time chart for multiplication in the previous article. Successive time

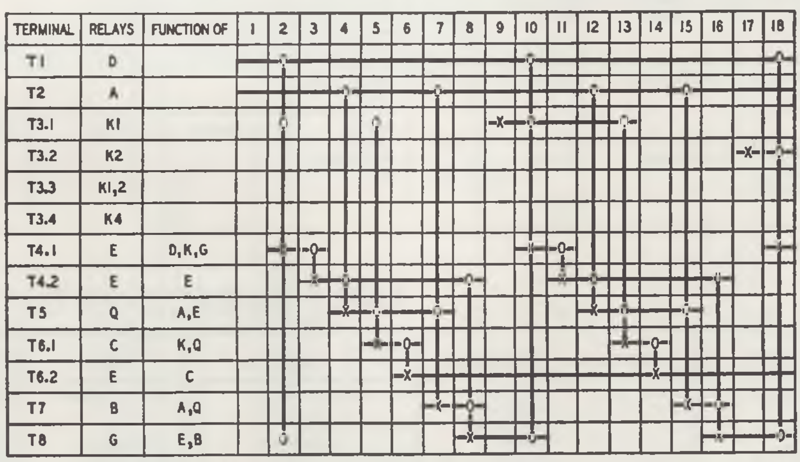
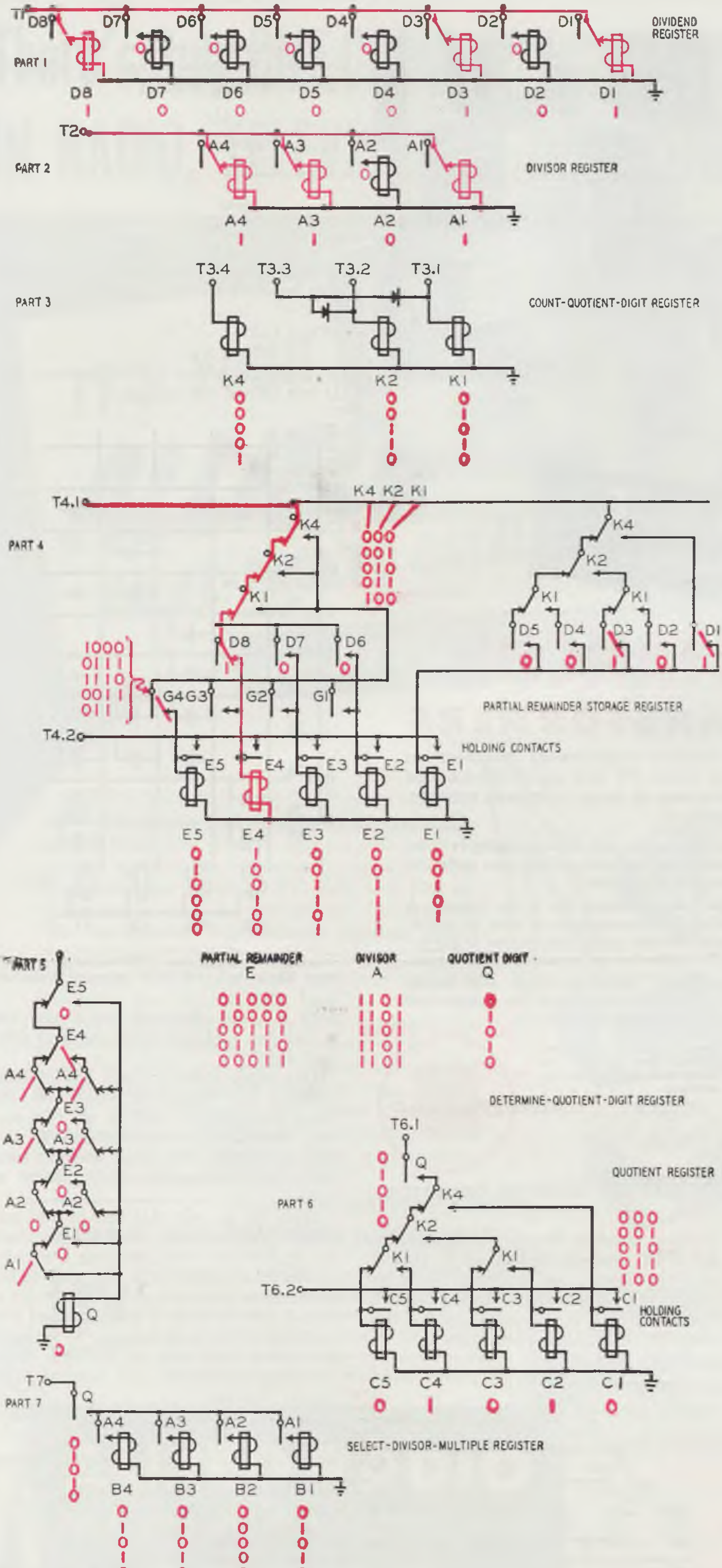


Fig. 1—Timing chart which shows the sequence of operation for the first two stages of the division with binary numbers performed by the circuit of Fig. 1.

*Author: *Giant Brains*



The circuit for doing long division with relays. Binary numbers are used for the process, and the circuits that carry current are shown in red.

intervals 1,2,3,4, are shown from left to right. In the first column, the different terminals are shown from top to bottom; in the second column, the names of the relays which the terminals energize; in the third column the names of the relay contacts through which the relays are energized. Each horizontal line begins when its terminal is energized, and stops when its terminal ceases to be energized. There are some vertical lines showing X's and O's. X marks the relays energized at a certain time, and the O's mark the contacts through which they are energized.

Now, you may say, it is all very well to be able to add, subtract, multiply and divide in binary notation, but how do we go from decimals to binaries?

In fact, even before we ask this question, we have to ask: how will the machine take in a decimal number? In other words, how will the machine accept it, record it, and store it?

Ordinarily a calculating machine (or some auxiliary part of it) will have a keyboard, containing keys numbered 0,1,2 up to 9. Often the keyboard will have a different column for each column of the number to be inserted in the machine. To put in a number like 593, we press down the 5 key in one column, the 9 key in the next column, and the 3 key in the third column.

In many calculating machines, the result of pressing down a key, say 3, is to turn some little counter wheel 3/10 of one complete turn. But in our machine we want the result of pressing down the 3 key to be the energizing of certain relays, so that we can use the information later in the machine.

We would reasonably desire to convert any one of these ten decimal digits 0 to 9 into a pure binary number according to Table I.

Fig 2 is a circuit which will do this (using 15 rectifiers and 4 relays).

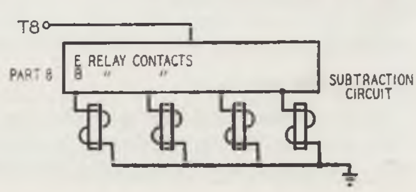
For example, if we press the 3 key, relays A2 and A1 are energized, but not relays A8 and A4, and so the information produced in the relay register is 0011, which is the binary number three.

In this way the decimal number 593 can be converted into 0101 1001 0011 stored in 12 relays. This form of representing a decimal number by a "code" for each digit is *coded decimal* rotation.

Now how do we go from 0101 1001 0011 to what this number is in pure binary notation? 593 of course is 5 times 10 times 10, plus 9 times 10, plus

Table I—Decimal to Binary Conversion

Decimal	Binary	Decimal	Binary
0	0	5	101
1	1	6	110
2	10	7	111
3	11	8	1000
4	100	9	1001



3, and all we have to do is write this in binary and tell our machine do it:

0101 times 1010 times 1010 plus
1001 times 1010, plus
0011.

And this our machine can do because it has an addition circuit and a multiplication circuit.

It will be neater to program this operation with:

5 times 10, plus 9,
all times 10, plus 3.

Thus for a ten-digit decimal number, we shall only need nine multiplications.

Binary to decimal

Now suppose that we have the opposite problem. Given a binary number, we want to find the corresponding decimal number. We divide this number by 1010 (one-oh-one-oh, or 8 plus 2, or 10 in binary) and find the remainder, which will be less than 10, and store it. Then we take the quotient, and divide that by 1010, and store the new remainder. And so on.

For example, suppose we desire to convert the binary number 10000101 into a decimal number.

```
              1101
1010 ) 10000101
         1010
          1101
          1010
           1101
           1010
             11, which is 3 in decimal,
```

and becomes our first decimal digit.

```
           1
1010 ) 1101
       1010
         11, which is 3 in decimal, and
```

is our second decimal digit

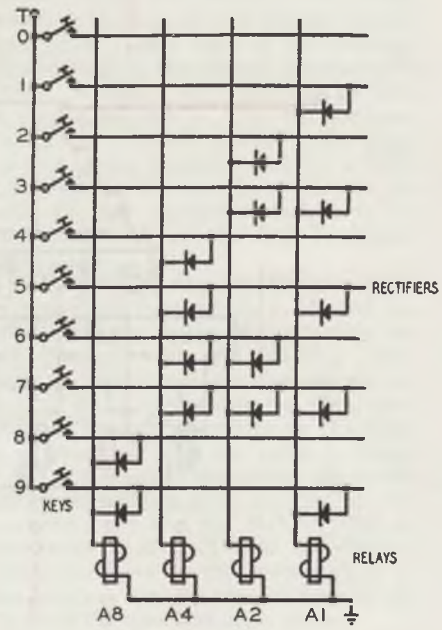


Fig. 2—Circuit for converting a decimal digit to a 4-digit binary number.

```
         0
1010 ) 1, our dividend being also our
```

remainder, and becoming the first decimal digit.

Our relay electric brain has division circuits and registers where we can store remainders; and so we can convert from binary into decimal. In this case we obtain the coded decimal form 0001 0011 0011 which is the same as 133.

How do we get this out of the machine? For example, suppose we have ten typewriter keys, bearing the characters 0,1,2,3,4,5,6,7,8,9. We wish to impulse these keys in order. The circuit in Fig. 3 will do this. When terminal T is energized, the appropriate K relay is energized, depending on the state of the A relays which hold the corresponding binary number.

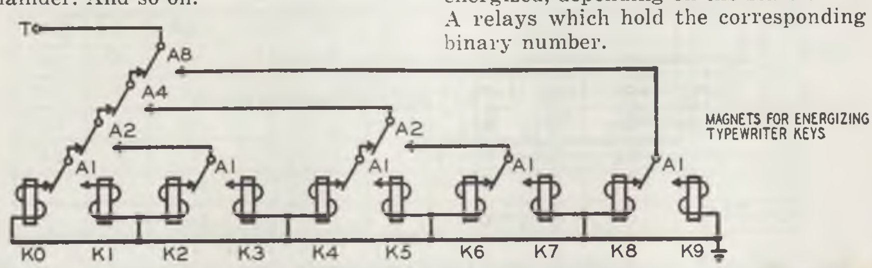


Fig. 3—Circuit for converting 4-digit binary system digits to decimal digits.

How an Electric Brain Works

Part V—Having learned his arithmetic, Simon is ready to be coordinated by automatic control so that he can learn to follow instructions and find the answer to a problem

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

IN previous articles of this series we have shown how an electric brain made of relays can:

1. Store information in registers;
2. Transfer information from one register to another;
3. Add, subtract, multiply, and divide;
4. Convert from decimal to binary notation, and back again.

These operations are the same as those of an automatic relay calculator. However, we have not yet completed the electric brain, for we have not yet arranged for these operations to be carried out in sequence one after another *under automatic control.* This we shall now set out to do.

When an electric brain operates under automatic control, it carries out instructions one after another in some planned sequence. In one important type of electric brain, the instructions are written out on a long piece of tape, in a language the machine can read. The tape may be made of paper punched in a pattern of holes, or of plastic impregnated with magnetic particles, magnetized in a pattern of spots, or of other recorded material. As each section of the tape comes to the input reading device of the machine, the instructions are read by the machine and carried out.

* Author: Giant Brains

For example, we might desire to give to a machine instructions such as these:

1. Take 24.
2. Take 13.
3. Add them.
4. Store this result (result No. 1).
5. Take 45.
6. Take 31.
7. Subtract the latter from the former, and store the result (result No. 2).
8. Compare these results and record 1 if result No. 1 is greater, and 0 if result No. 1 is not greater.
9. Store this result (result No. 3).

When a machine can carry out a set of instructions such as this under automatic control, it is an electric brain. The capacity to store and refer to information (as these instructions imply) and to carry out a chain of operations is the essential capacity of a brain, mechanical, animal or human.

But how do we get a machine to do this?

Commands

In the first article of this series we noted that a mechanical brain was like a telegraph system with many stations, where information could be telegraphed from one station to another. Accordingly, the key to getting a machine to carry out a sequence of instructions is:

1. Organize each instruction in the form of a command involving two registers or stations (such as Albany and Boston), connect them to the main telegraph line, specify the direction of the message between them (for example, from Albany to Boston), and then transfer the information.
2. Give the computer a long series of successive commands, each of this same standard form.

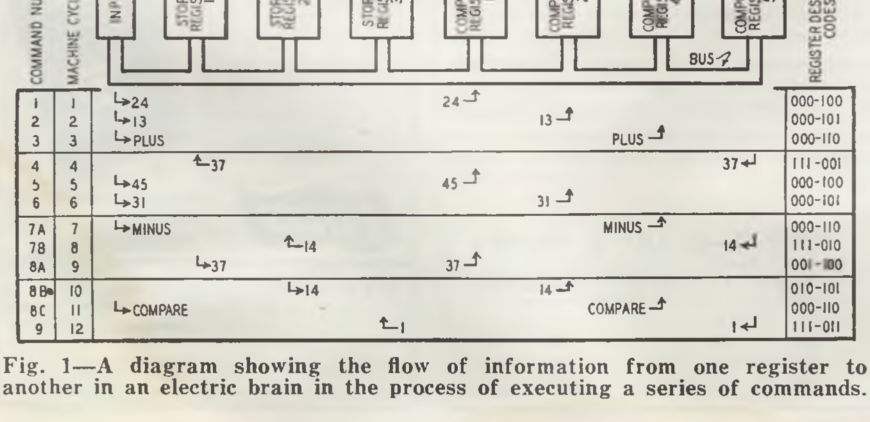


Fig. 1—A diagram showing the flow of information from one register to another in an electric brain in the process of executing a series of commands.

It is a remarkable fact that almost all numerical and logical handling of information can be reduced to a series of identified commands (i.e., commands that are identified by numbers or labels): "Command No. ....: Transfer information out of register .... into register ...., and then proceed to Command No. ...." This fact becomes easier to see when we remember that some registers are factories, and manufacture new information out of old, and so put out different information from that which went into them.

Yes, yes, you may say, that may all be very true, but up above you put down a series of instruction for an example: now how do you make a machine carry out that series of instructions—how do you convert them into a series of commands of the kind you speak of?

Here is what we do, supposing that we are to give the machine instructions from a tape (see Fig. 1):

1. Transfer information from INPUT station (where the tape is read, and at this time has 24 at the reading point) into COMPUTER REGISTER NO. 1 (the first register in the computer).
2. Transfer information from INPUT station (the tape has moved along, and now has 13 at the reading point for reading by the machine) into COMPUTER REGISTER NO. 2.
3. Transfer information from INPUT station (the tape has again moved along, and now has at the reading point a signal which means addition) into COMPUTER REGISTER 4 (which, we shall suppose, is a special register enabling us to make the computer add or do some other operation).
4. Transfer information from COMPUTER REGISTER NO. 5 (which we shall suppose is the output register of the computer) into STORAGE REGISTER NO. 1.
5. Transfer information from INPUT station (the tape has moved along, and now holds 45) into COMPUTER REGISTER NO. 1.
6. Transfer information from INPUT station (the tape now has 31 at that point ready to be read) into COMPUTER REGISTER NO. 2.

7-a. Transfer information from INPUT station (the tape now presents a code or signal that means subtraction) into COMPUTER REGISTER NO. 4 (the operation-accepting register); and

7-b. Transfer information from COMPUTER REGISTER NO. 5 (the result register) into STORAGE REGISTER NO. 2 (We see that Instruction No. 7 in the first list breaks into two commands).

8-a. Transfer information from STORAGE REGISTER NO. 1 into COMPUTER REGISTER NO. 1.

8-b. Transfer information from STORAGE REGISTER NO. 2 into COMPUTER REGISTER NO. 2.

8-c. Transfer information from INPUT station (the tape now presents a signal which when read will mean comparison) to COMPUTER REGISTER NO. 4. (When the computer is called on to compare *a* and *b* we shall suppose it is able to produce 1 if *a* is greater than *b*, 0 if *a* is not greater than *b*. We note that Instruction No. 8 breaks up into three commands.)

9. Transfer information from COMPUTER REGISTER NO. 5 (the result register) into STORAGE REGISTER NO. 3.

Each of these commands will take place during a complete operating cycle of the machine (see the numbers of the machine cycles in Fig. 1). For each cycle, naturally, the machine requires not only the number and operation

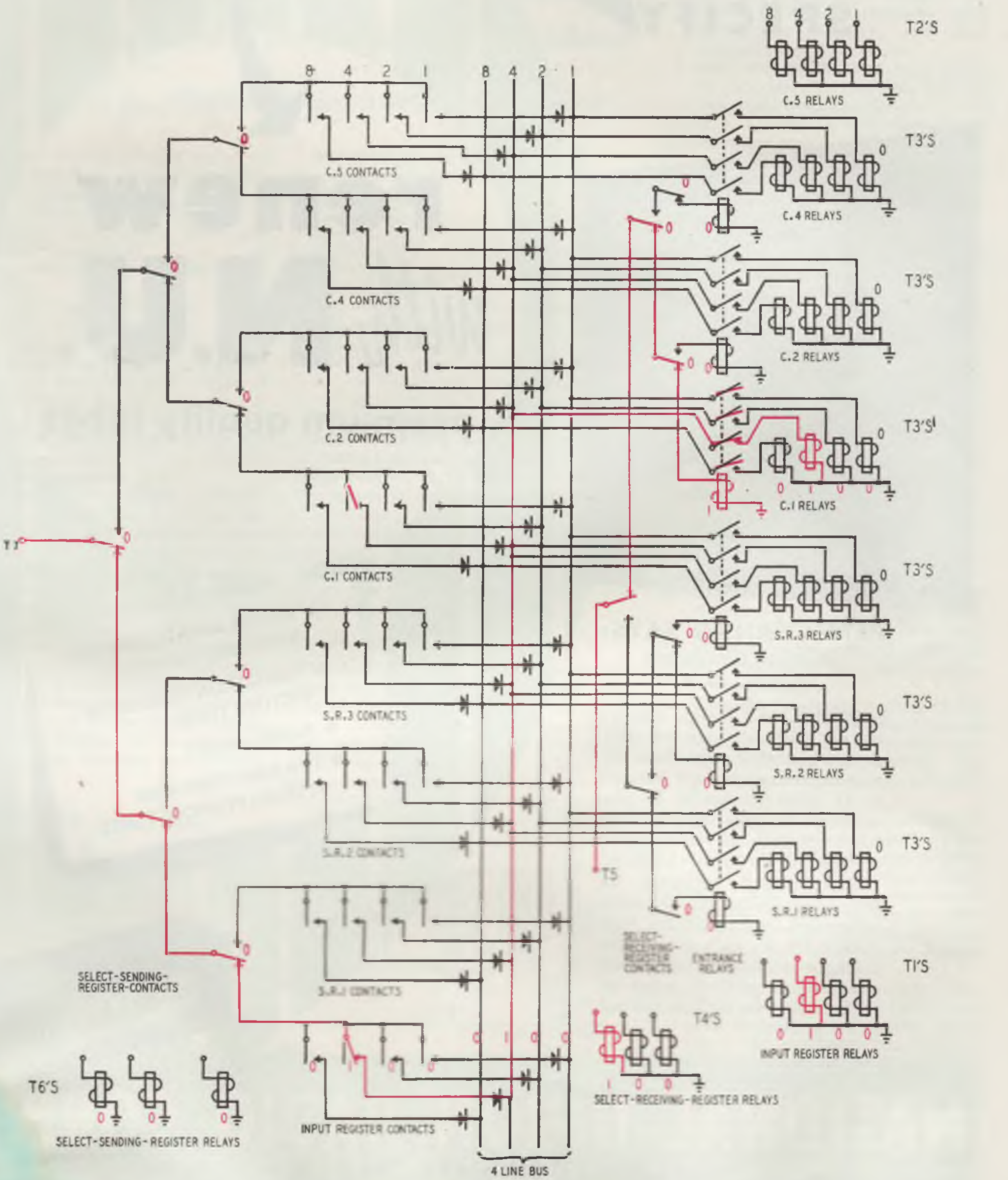


Fig. 2—The main transfer circuit of the electric brain. The 4-line bus is sufficient for transferring one decimal digit coded into binary notation. The circuits shown in red carry current during the first step of the problem.

information from the input station, but also knowledge of the designated registers. So at each cycle we must also give the machine the signals that call for the designated two registers at each cycle. Computer register No. 3 is omitted from this diagram because we do not need it for this problem, but we shall have occasion to use it later for other problems.

So, for our particular series of commands, we shall suppose that the codes of the registers are 000 to 111 as shown right above them in Fig. 1, and that these codes are also given to the machine, cycle by cycle. The machine is constructed to know that the first code refers to the sending register, and that the second code designates the receiving register.

What information then will be on the tape, cycle by cycle? The tape will contain the following information:

Cycle	*Information*
1	24-000-100
2	13-000-101
3	PLUS-000-110
4	—-111-001
5	45-000-100
6	31-000-101
7	MINUS-000-110
8	—-111-010
9	—-001-100
10	—-010-101
11	COMPARE-000-110
12	—-111-011

Of course, just as the registers are represented by codes such as 000 or 100, so the numbers like 24 and 13 and the operations PLUS, MINUS, COMPARE will be represented by codes. Certainly, the machine will not be required to translate a collection of alphabetical letters into an operation indication. Probably, all of these numbers and operations will be translated into sets of 1's and 0's corresponding to equipment that has two different stable conditions.

Circuits

Fig. 1 is a schematic diagram of the flow of a problem through an electric brain. But what is the diagram of the actual circuits?

In Fig. 2 is shown the main transfer circuit for the electric brain of Fig. 1. By looking at Fig. 2, we can see most, though not all, of the essential electrical network for making the events of Fig. 1 happen.

Down the center is a four-line bus; this is sufficient for transferring one decimal digit coded into binary notation. The second decimal digit could be provided for by another four lines of bus, but these have been omitted for the sake of simplicity.

On the right side of Fig. 2 are the eight sets of relays which correspond to the eight registers of Fig. 1. We note that we cannot read in from the bus to either the INPUT REGISTER or the COMPUTER 5 REGISTER. This is as it should be, because COMPUTER 5 REGISTER is filled from the information in COMPUTER REGISTERS 1, 2, and 4 and the INPUT REGISTER is filled from the tape.

The sets of terminals marked T1's,

T2's, and T3's are energized at an earlier time. Their hold contacts hold a pattern of information in the relay registers. There are, however, exceptions: for example, provision must be made for resetting any register before information is read into it. But as usual, we will not consider complications until a later stage.

On the left side of Fig. 2, we see mainly the "read-out" contacts of the relay registers on the right side of Fig. 2. Since any one of the eight registers can be read out of, all eight of the registers are represented.

The very first transfer is of the decimal digit 4 (in binary, 0100) of the first number, 24. Before this transfer begins, we see the binary number 0100 stored and held in the INPUT REGISTER RELAYS, recorded there in red 0100, the 0's saying that relays 8, 2, 1 are not energized and the 1 saying that relay 4 is energized. Consequently, on the left side we see the number also stored in the INPUT REGISTER CONTACTS.

Now, we want to transfer out of the INPUT REGISTER, code 000, into COMPUTER REGISTER NO. 1, code 100. Accordingly, information in the tape energizes the SELECT-RECEIVING-REGISTER RELAYS in the pattern 100, and the SELECT-SENDING-REGISTER RELAYS in the pattern 000 (using the terminals T4's and T6's, respectively), and affecting the positions of their contacts.

We now pulse terminal T5. This picks up only the ENTRANCE RELAY for COMPUTER REGISTER NO. 1, and connects only the pickup coils of COMPUTER REGISTER NO. 1 to the bus.

We are now ready to pulse terminal T7 and read through the whole circuit, the main transfer circuit. This we do. As soon as we do it we transfer the pattern 0100 into COMPUTER REGISTER NO. 1. All the circuits that carry current during this process are shown in red. This same general process repeats once each cycle, and again and again, carries out the automatic operation of the electric brain according to the instructions on the tape.

How many tapes?

One of the questions considered in designing an electric brain is whether it should have one tape, several tapes, or no tapes.

For a small machine such as Simon there is an advantage in having one tape, because one tape-reading mechanism is cheapest. In each cycle of such a machine, there can be three times: at two of the times, the machine reads from the tape the codes of the receiving register and the sending register, and at a third time the machine reads from the tape the number or operation information.

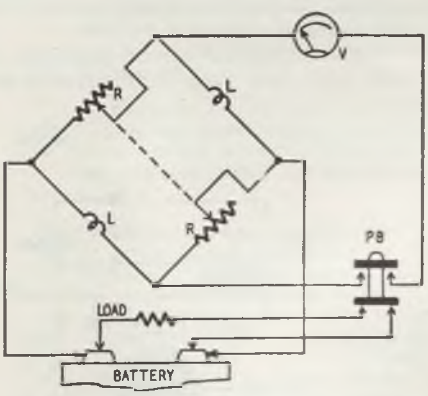
It is often more efficient to have two tapes. Usually they will consist of a *problem tape*, containing the numbers belonging to a particular problem, and a *program tape*, containing the transfer commands, operations, and constants (the numbers that do not change from problem to problem).

It is still more efficient to put almost all of the program into the storage or *memory* of the computer. This often becomes possible when we go to electronic computers with 1,000 or more registers in the *rapid memory*, where the rapid memory is the part of the storage registers of the machine and any specified piece of information can be gotten at very quickly. This is in contrast to the *slow memory*, where many seconds or some minutes may be needed to get at a specified piece of information. For example, we can store dozens of useful common routines, such as a procedure for getting square root, in the rapid memory of the machine, and then tell the machine to pick up any one of them and use that procedure when a given indication in the problem occurs. This has been done with the Model 6 relay computer in Bell Telephone Laboratories in Murray Hill, N. J., which is a particularly well-educated electric brain; it has routines and subroutines which are numbered in the hundreds; they belong to half a dozen levels of "intelligence," and can call for each other.

EXPANDED SCALE METER

Designed especially for testing storage batteries, this meter has a range limited to a minimum of 5 and a maximum of 6 volts. Voltages below 5 cause no deflection. The unit is described in patent 2,509,486, issued to C. W. Danzell.

The heart of the circuit is the non-linear bridge shown. This contains two ganged arms R of equal resistance and two identical low-voltage lamps L. The lamps have nearly zero resistance when



cold. This resistance rises rapidly with filament temperature. R is chosen to equal the resistance of L when the bridge input is 5 volts. Since this balances the bridge, there can be no deflection at that value. Full-scale meter reading occurs at 6 volts.

The meter needle moves backward when the bridge voltage is less than 5. This reverse reading is suppressed as follows: first, all connections are made to the battery under test. Then the push-button is depressed. The lower pole connects a load resistor. An instant later the upper pole closes the meter circuit. By the time the meter is in the circuit the lamps are warm. When the button is released, the meter circuit is interrupted at once. Therefore the reverse deflection is not observed during test.

How an Electric Brain Works

Part VI—Although no genius himself, Simon now helps us to understand how an electric brain is put together

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

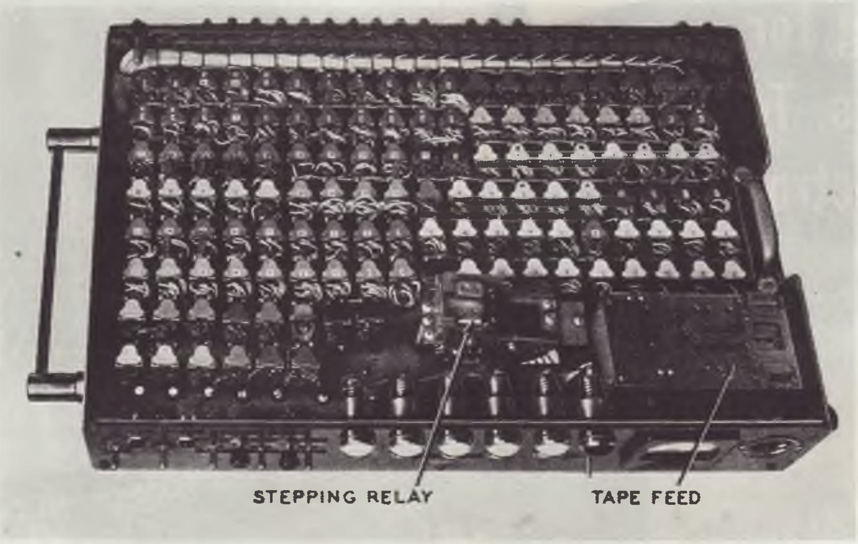


Fig. 1—A photo of Simon with his top cover off. He has 120 relays that make up his gray matter, as well as a few spares for development in the future.

MOST of the operations that are essential for an electric brain have now been explained and illustrated. In previous articles, we have covered relay circuits for:
 storing and transferring information;
 performing arithmetical operations;
 arranging automatic control.

Now, how do we put all these operations together, so that we actually succeed in making a complete electric brain that will work successfully?

As usual in this discussion, we shall keep to a simple example and leave out the more complicated sides of questions, so that principles may be made clear. But instead of having to talk about hypothetical examples, this time we can talk about an actually existing example—the baby mechanical brain Simon that we have described previously.

The machine Simon was pictured in the October, 1950, issue of RADIO-ELECTRONICS, and some more pictures of it are given here. A top view is shown in Fig. 1, and a bottom view in Fig. 2. Because Simon does not have covers in these pictures, some idea of what the machine is really like may be gained from them.

From the top view we can see that Simon has:
 a front panel, with lights, buttons, switches, and a meter;
 a tape feed, for feeding 5-hole paper tape;
 a stepping switch, for timing the machine; and—
 some 120 active relays, for operations.

The bottom view shows:
 some banks of small rectifiers;
 some capacitors, for spark suppression; and
 a lot of wiring.

When finished in May 1950 Simon knew only the numbers 0, 1, 2, and 3. As a result of changes of his circuits made in August 1950, however, Simon now can take in numbers from 0 to 15 and can report numbers from 0 to 31. With some more changes, Simon could handle bigger numbers still.

Earlier we said there were five parts to every mechanical brain: input, output, storage, computer, and control. Where are these various parts in Simon located?

Input

The *input* of Simon consists of 5-hole paper tape, the tape-feeding mechanism (see Fig. 1), and switches and buttons (see Fig. 3). These are all the ways in which you can give information to Simon so that he will know exactly what to do and in what order to do it.

Ordinarily, when you want to run a problem on Simon, you write out the commands (with 1's and 0's, using binary notation) cycle by cycle on a piece of paper. This is called the *coding* for the problem. Then you punch this coding onto a piece of tape, put the tape into the tape feed, and turn on the power. The tape runs, and Simon clicks away, "thinking," as he works out the answers to his problems.

You may, if you wish, give Simon general instructions applying to any one or more numbers, and then put the numbers into the machine by hand from

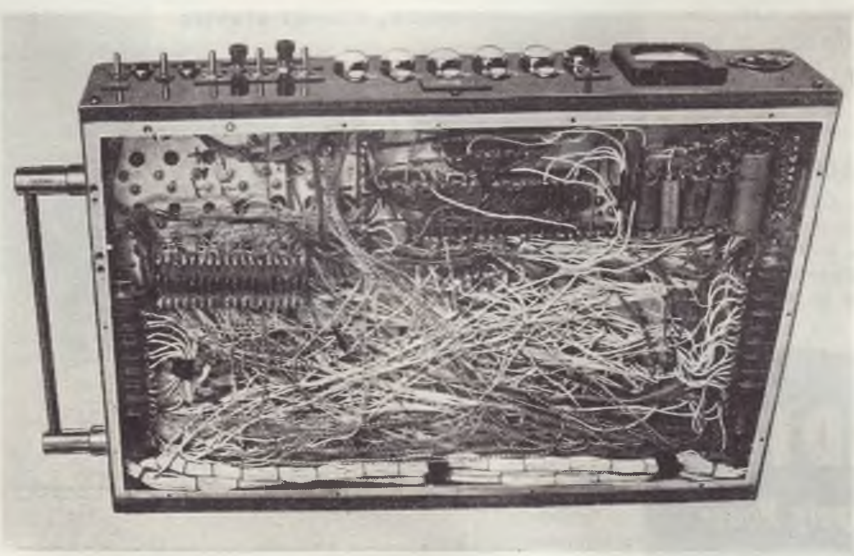


Fig. 2—Simon's underside is mostly a lot of wiring. He also has some small selenium rectifiers and some capacitors that are used for spark suppression.

* Author: *Giant Brains*, John Wiley & Sons, Inc.

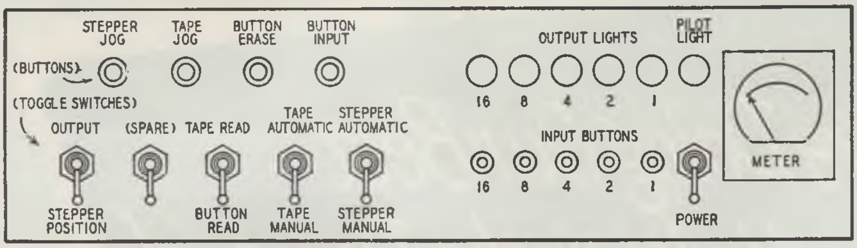


Fig. 3—Diagram of Simon's front panel showing his controls and indicators.

Table of Relay Abbreviations		
Abbreviation	**Name of Group**	**Purpose**
ASR	Auxiliary Stepping Relay	Slow down the stepping switch.
BR	Button Register	Temporarily record numbers in instructions from buttons.
CR	Computer Registers	Compute
ER	Entrance Relays	Allow information to enter registers.
IR	Input Registers	Temporarily record numbers from the tape or the buttons.
OR	Output Registers	Hold answers, to be shone in the output lights.
PR	Program Relays	Record programming information from the tape or from the buttons, and control Simon.
RR	Reset Relays	Reset, release, or clear registers, so that new information may be stored in them.
SPR	Step-Position Output Relay	Allows the position of the stepping switch to be read in the output lights.
SR	Storage Registers	Store information until used.
SYR	Synchronism Relays	Arrange that the tape and the machine cycles shall be automatically in synchronism.

time to time at appropriate points. You can tell Simon to stop at an appropriate place, by punching into the tape an indication for a "programmed stop," as it is called. When Simon makes a programmed stop, you press any one or more of the buttons numbered 8, 4, 2, 1 (see Fig. 3). This action inserts the number which is the sum of the figures selected, into the button relays (register BR in Fig. 4). Then to transfer the number from the button register into Simon's regular input register (register IR 2 and IR 1 in Fig. 4), you press a button called BUTTON INPUT (see Fig. 3). Finally, you press the STEPPER JOG button (see Fig. 3), and Simon runs on, automatically reading and obeying the tape.

Output

The *output* of Simon also is on the front panel (see Fig. 3). It consists of five lights, called OUTPUT LIGHTS 16, 8, 4, 2, and 1. In these lights any number from 0 (no lights shining) up to 31 (all five lights shining) can be indicated. For example, if OUTPUT LIGHTS 16, 4, and 1 are shining, the number indicated is 16 plus 4 plus 1, or 21. You program Simon so he will stop when a result has been delivered. You can examine and, if you wish, copy the result; and then, when you press the STEPPER JOG button, Simon will run on.

The sixth light on the front panel is a red pilot light which shines when the power is on; and the meter reads the voltage at which Simon is operating. Simon will operate at 20 to 35 volts d.c.

Storage, computer, and control

The functions of storage, computing, and control in Simon are carried out by relays. Of course, a good deal of the control is also expressed in the tape, but relays take the information from the tape and operate with it.

A diagram showing all the 129 relays of Simon is given in Fig. 4. Each relay is represented by a rectangle placed in the correct physical location (compare Fig. 4 with Fig. 1). There are 19 columns, corresponding to the columns shown in the photograph. Fig. 1, and 8 rows (the two bottom ones incomplete) corresponding to the rows shown in Fig. 1.

The relays in Simon may be designated in either one of two ways, according to location in the machine, and according to function.

For wiring purposes, the relays are designated by location, that is, by row (a single letter C, D, E, F, G, H, N, or P) and by column (a number 1 to 19).

But for purposes of understanding Simon, the functional designations are useful. To designate a relay by its function, each relay has an abbreviation that may have three parts. Part 1 is two or more letters, to tell the kind of register. Part 2, if any, is a number, used to number off registers all of the same kind. These numbers are not always consecutive, for reasons that will be explained later. Part 3, if any, is a number in parentheses, used to tell the binary digit being handled by that relay.

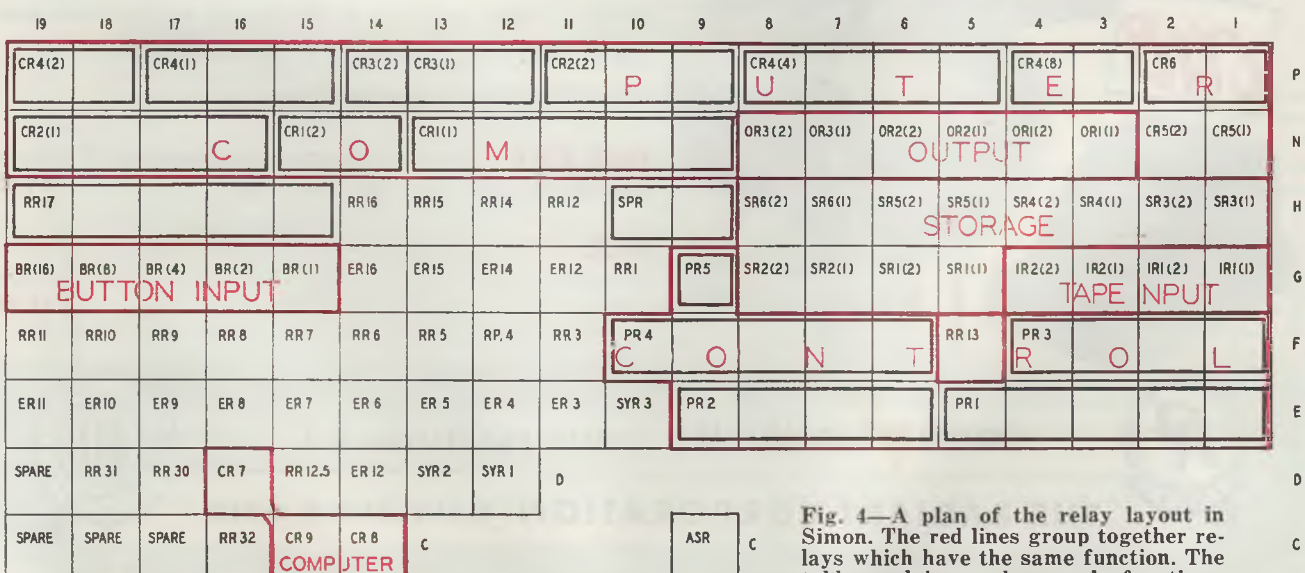


Fig. 4—A plan of the relay layout in Simon. The red lines group together relays which have the same function. The table explains each group's functions.

The numbers in part 3 may be 16, 8, 4, 2, or 1 corresponding to the digits in a five-digit binary number, for example, 10101 in binary, meaning one 16, plus no 8, plus one 4, plus no 2, plus one 1, or 21 in decimal. Now you may say: "But I thought Simon was a machine that handled only numbers of two binary digits." That is true at any one time; but it is quite possible to get Simon to take care of one pair of binary digits on one machine cycle, then on the next machine cycle to take care of a second pair of binary digits, and then on the third cycle to take care of a third pair of binary digits, and so on. This is one of the changes made in August, 1950.

The table on page 58 is a list of the part 1 or letter abbreviations in Fig. 4 and some explanation of them, giving the names of the group of relays the letters stand for, and the purpose of that group of relays.

Looking at the diagrams, we can see this same main grouping expressed roughly in the red boundary lines. The black boundary lines, on the other hand, group together relays that have exactly the same function, whose coils were wired in parallel to provide a sufficient number of contacts. At the time when we bought up 24-volt d.c. war-surplus relays to construct our little computer, it seemed the best we could do was to get relays that had one single-throw contact and two double-throw contacts. For example, in the upper left-hand corner, the diagram shows two paralleled relays to give enough contacts for computer register 4, and second digit (the digit expressing twos in binary notation). Naturally it would be much more efficient as far as space is concerned to use relays with the proper number of contacts in each case, but this would of course rule out the surplus market as a source of relays. In any case, Simon's layout might look rather different, depending on what particular relays happen to be most available.

(To be continued)

NBS SEEKS ENGINEERS

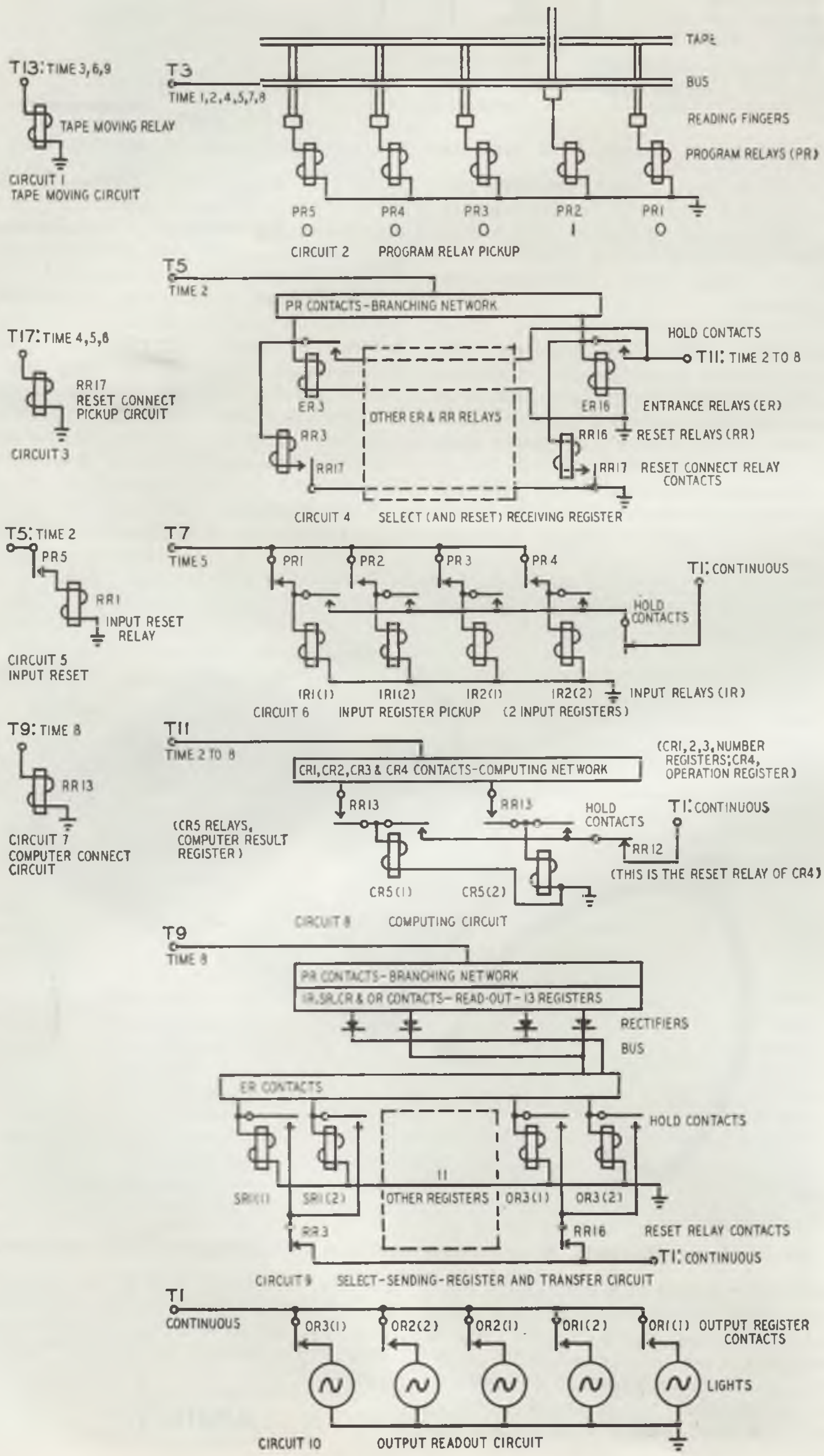
Competent electronic engineers and physicists are urgently needed by the National Bureau of Standards, Department of Commerce, for research and development work needed by the Department of Defense. The work offers opportunity for professional growth and may often form the basis for a doctor's thesis. The greatest need is for engineers and physicists at starting salaries ranging from $4,600 to $8,800 a year.

Applications for these jobs are being accepted by the U. S. Civil Service Commission, Washington 25, D. C., under Announcement No. 226 for Physicist, Chemist, Metallurgist and Mathematician, and its amendment for Electronic Scientist. Requirements for the jobs are given in the announcement. Applications may be obtained from most first- and second-class post offices, from Civil Service regional offices, or from the Commission's Washington office.

How an Electric Brain Works

Part VII—Analysis of Simon as a complete working unit. How the various sections are made to work together

By EDMUND C. BERKELEY and ROBERT A. JENSEN



IN THE last article, we began to talk about Simon, a complete baby electric brain, made of relays, a stepping switch, and a paper-tape feed. We anchored down the terms "input," "output," "computer," "storage," and "control," each into a particular set of relays that actually perform that function.

Now, what is the general scheme, and the circuit wiring, whereby this equipment works as a complete electric brain?

This may be seen in Fig. 1, which shows, sketchily and schematically, the 10 essential circuits of Simon, beginning with moving the tape, and ending with putting informaton out in the output lights.

To explain these circuits, we should start with their timing. Each of the terminals shown, T1, T3, etc., is energized at a certain time or times by means of the stepping switch as the calculation proceeds.

For Simon or any electric brain to operate, things have to happen precisely in succession, in sequence, one after another. This is the heart of automatic control. In Simon the timing is done by the stepping switch. The Clare Relay Co. stepping switch that we bought on the war surplus market when we were constructing Simon, had 20 timing points and 6 levels, but we found that it stepped too fast. The easiest change to make was to wire the points together in pairs, thus effectively giving the stepper 10 timing points. Also, we replaced the stepper's nonbridging wipers (which broke current between each point and the next) with bridging wipers, so that we would have uninterrupted current for holding up relays when desired. The switch was modified by installing a coil to operate on 24 volts d.c., the standard operating voltage for Simon.

As we worked out the circuits of the machine, the points were wired together to give seven terminals that were numbered T3, T5, T7, T9, T11, T13, and T17 (see Fig. 2), carrying currents at different times. The odd numbers were used to indicate that the current was of the same sign, positive, as the source,

Fig. 1—Simon's 10 essential circuits, showing operational sequence and connections to stepping switch terminals.

in contrast with negative, or the ground, terminal T2.

Timing of the machine

Now, how are these timed currents used to energize the relays and circuits of Simon? This is shown in the timing chart of Fig. 3; it is not altogether complete, but most of the operation of the machine is indicated there, and it will be understandable to readers who have followed the partial diagrams in earlier instalments of this series. Let

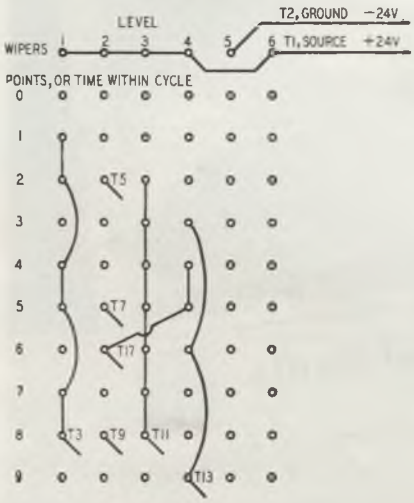


Fig. 2—How the 6-level stepping switch that activates the relay banks is wired.

us go through the timing chart of Fig. 3 and the diagrams of Fig. 1 and see what we can tell from them.

At time 1, there is a red O on the first row, and a red X on the second row, and a red line connects them. This means that at time 1, we read through holes in the paper tape (which at this time is still) and we pick up the corresponding program relays (see Circuit 2 in Fig. 1). At time 2, similarly interpreting the red line, we read through the positioned contacts of program relays to select the receiving register, and pick up its entrance relay, which is held up until time 8 (see Circuit 3). We also optionally (if there is a hole in position 5 on the tape) pick up RR1 (reset relay No. 1), which action resets the input register by interrupting its hold current (see Circuits 5 and 6). At time 3, all that happens is that we move tape (see Circuit 1), and drop out the program relays. At time 4, we read through tape again and pick up the program relays once more, for they are the relays in which the information from the tape is always immediately put (see Circuit 2). Also at time 4, we read through the hold contact of the selected entrance relay and pick up the matching reset relay, which resets the receiving register by interrupting its hold current (see Circuits 3 and 4). At time 5, we read through contacts of the program relays, and pick up the input registers 1 and 2, storing there a number or operation (see Circuit 6). At time 6, we move tape and drop out the program relays. At time 7, we read through holes in the paper tape, and pick up the program relays once more (see Circuit 2), this time to select the sending register.

At time 8, we transfer information (see Circuit 9). We read through, from the source:

1. contacts of the program relays which select a sending register (these relays were energized at time 7);
2. contacts of that sending register (held up by continuous current);
3. the bus;
4. the contacts of the receiving register's Entrance Relay (which have been closed and have been held from time 2 to 8);
5. and the coils of the selected receiving register, to ground.

And at time 9, we move tape preparatory to the next cycle, and drop out the program relays and entrance relay.

Independently of this main sequence of events, computing takes place in Circuit 8. The computer consists of three registers CR1, CR2, CR3 which take in numbers, and a fourth register CR4 which takes in an operation. Suppose that on previous cycles, these registers have been filled with the desired information, and that CR4 is the last one so filled. Then by means of T11, current is passed through the contacts of those four registers. To avoid back circuits, however, the computer is connected only at time 8 to the fifth computer register, CR5, which stores results (see Circuits 8 and 7)

Storing and transferring

The two things that are the first order of business in an electric brain are to store information and to transfer information. In Simon, information is stored in any one of 16 registers, each capable of holding two binary digits. Information is transferred as pulses of current along a two-line bus. The 16 registers of Simon and the codes used for "calling" them (either to transmit information they hold or to receive and store information) are:

Register	Code	Entrance Relay	Reset Relay
IR1	0000	none	RR1
IR2	0001	none	RR1
SR1	0010	ER3	RR3
SR2	0011	ER4	RR4
SR3	0100	ER5	RR5
SR4	0101	ER6	RR6
SR5	0110	ER7	RR7
SR6	0111	ER8	RR8
CR1	1000	ER9	RR9
CR2	1001	ER10	RR10
CR3	1010	ER11	RR11
CR4	1011	ER12	RR12
CR5	1100	none	RR12
OR1	1101	ER14	RR14
OR2	1110	ER15	RR15
OR3	1111	ER16	RR16

In the last article and in previous articles, we told most of the story about the input, output, and storage registers of Simon. The computer registers, however, require some more explanation here.

Computer registers

The computer of Simon consists of relays and wiring by means of which information is operated on and changed into other information. Simon's computer registers (abbreviation CR) are: CR 1 to 3, which take in numbers; CR4, which takes in the operation; CR5, which gives out the result; and CR 6 to 9, which help in the operations involving arithmetical carrying and were recently added.

Simon at present writing has nine operations built into it. For Simon to perform any one of these operations, he must be instructed. How do we instruct him? These operations also have certain codes, and here are the codes and the names of the operations:

No.	Code	Operation
1	0000	Add, No Carry;
2	0001	Negate, No Carry; Fours Complement
3	0010	Greater Than
4	0011	Selection
5	0100	Logical AND
6	0101	Logical NOT; Threes Complement
7	0110	Logical OR
8	1000	Add, Subject to Carry from Previous Addition
9	1001	Negate, Subject to Carry from Previous Negation

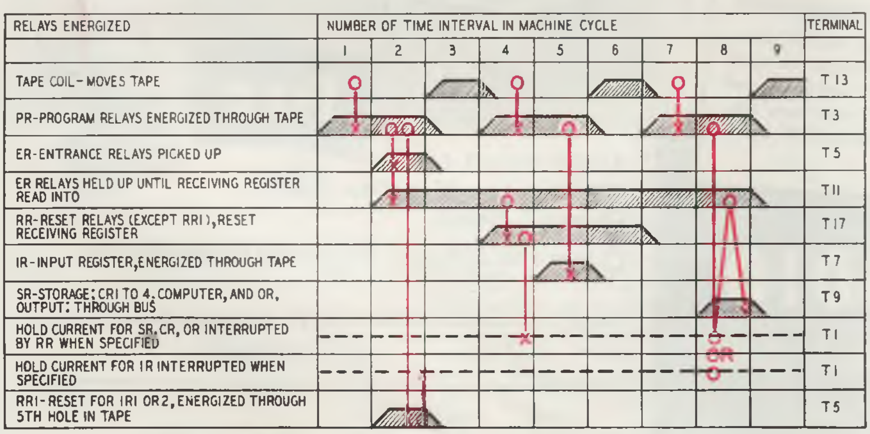


Fig. 3—Timing chart, showing how operations are carried through the machine.

Now, what do we mean by these operations? Regularly the complete meaning of an operation is described by giving a table of the outputs for some of the inputs, and then covering the remaining cases of output-input by general statements. This we have done in Chart 1, in rather condensed language. The lower case (not capital) letters a, b, c, refer to numbers that Simon knows, 0, 1, 2, 3. The lower case letters p, q, r, refer to *truth values:* the truth value of a statement is 1 if the statement is true, and 0 if the statement is false. A primitive way of indicating a truth value is with check marks (√) and crosses (×). But check marks and crosses are not numbers like 1 and 0, and cannot be combined like numbers, and there is no need to bother with them, for using them is like using a crystal set when you could use a console radio. The expression T(...), which is read: "T of ...", where ... is some statement, is a nice short way of writing "the truth value of ...". Truth values are becoming more important all the time as a means of designing and economizing electronic computer circuits. For example, the electronic computer Maddida was designed largely by truth value algebra instead of circuit diagrams.

The capital letters P, Q, R refer to statements of which the truth values are p, q, r, respectively. Statements have to be expressed usually with words, and sometimes may be expressed in other ways. But the little letters p, q, r are truth values and are always 1 or 0. It is readily understood that:

$p = T(P)$, $q = T(Q)$, and $r = T(R)$.

For example, let us consider the fourth operation, that of selection. The general statement for this operation is $c = ap + b(1-p)$.What will this rule give us in a concrete case? Suppose a is 2, and b is 3, and p is the truth value of the statement "0 is greater than 1."

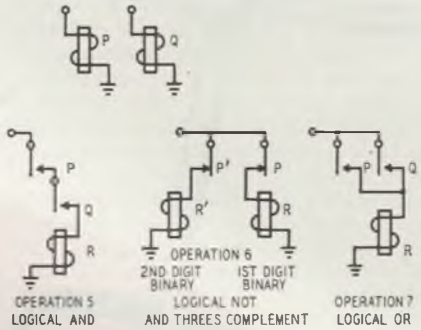


Fig. 4—Logical circuits. "And" circuit outputs a pulse only if both relays are closed; "not" circuit only if both are open; "or" if either or both are closed.

In other words, what we want to do is select 2 if 0 is greater than 1, and select 3 if 0 is not greater than 1. Here the statement that we are interested in is "0 is greater than 1"; this statement is false, and so p, its truth value, is 0. Putting $a = 2$, $b = 3$, and $p = 0$, in the rule,

$$c = a + b(1-p) = [2 \times 0] + [3 \times (1-0)].$$

The arithmetical result of this is 3, which of course agrees with what our common sense tells us. Why do we resort to all this roundaboutness? Because when you want to tell a mechanical brain to do something, you have to be very explicit.

Operations 8 and 9 need some comment. Their purpose is to enable us to handle numbers of more than two binary digits. If we are adding two four-digit binary numbers, for example, first we attend to the two binary digits at the right, using operation 1, Addition Without Carry. Second we attend to the two binary digits at the left, using operation 8, Addition, Subject to Carry, this process remembering whether or not there was a carry from that first step.

The circuits for operation 1 to 4 are given in Chapter 3 of *Giant Brains*, on pages 36 to 38. The circuits for operations 5 to 7 are given schematically in Fig. 4. We note that "logical NOT" and "threes complement" are the same operation, with the same code 0101. This is because if we use only the right-hand binary digit, we have the logical-NOT operation; and if we use both the left-hand and the right-hand binary digits, we have the threes complement. The circuits for operations 8 and 9 are not hard to design, in one way or another; there is no space to give them here.

Chart 1—Operations of Simon

Operation 1
Addition without carry
c = a .. b

a \ b	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Operation 2
Subtraction or negation without carry = fours complement
$c = 4p - a$
p T(a is 1, 2, 3)

a	p	c
0	0	0
1	1	3
2	1	2
3	1	1

Operation 3
Greater than
p = T(a is greater than b)

a \ b	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

Operation 4
Selection
$c = ap + b(1 - p)$

p	0	0	0	0	1	1	1	1
b	0	1	2	3	0	1	2	3
a								
0	0	1	2	3	0	0	0	0
1	0	1	2	3	1	1	1	1
2	0	1	2	3	2	2	2	2
3	0	1	2	3	3	3	3	3

Operation 5
Logical "and"
r = T(P and Q)

p \ q	0	1
0	0	0
1	0	1

Operation 6

Logical "not"
r = t(not P)

p	r
0	1
1	0

Threes complement
$c = 3 - a$

a	c
0	3
1	2
2	1
3	0

Operation 7
Logical "or"
r = T (P or Q)

p \ q	0	1
0	0	1
1	1	1

Operation 8
Addition, subject to carry
$c = a + b + p$
p T(previous addition was 1 + 3, or 2 + 2, or 3 + 2, or 3 + 3)

p	0	0	0	0	1	1	1	1
b	0	1	2	3	0	1	2	3
a								
0	0	1	2	3	1	2	3	0
1	1	2	3	0	2	3	0	1
2	2	3	0	1	3	0	1	2
3	3	0	1	2	0	1	2	3

Operation 9
Subtraction or negation subject to carry
$c = 3 - a + q(4p - 3)$
p = T(a is 1, 2, or 3)
q = T(previous negation without carry was of 0)

a	p	q = 0	q = 1
0	0	3	0
1	1	2	3
2	1	1	2
3	1	0	1

Programming and timing

The next thing we have to consider is the programming and timing of the machine as a whole.

In Simon, as constructed, we are using only one tape for both instructions and numbers. It proved to be convenient when constructing Simon to cause the tape to be read three times in each complete machine cycle, at time 2, time 5, and time 8. We call these three readings three *entries* of information into Simon. The first entry, at time 2, specifies the register which is to receive the information. When this entry is read, it causes the entrance relay of the receiving register to be energized, and so connects the coils of the receiving register to the bus, and clears out any information previously in it. The second entry, at time 5, puts any number or operation from the tape into input registers 1 and 2, so that the number or operation can be made use of in the machine. The third entry, at time 8, specifies the sending register, i.e., the register from which information is to be transmitted, and at the same time causes the information in that register to pass through the bus into the receiving register.

In some electric brains, we would have to consider carefully the subsequences of the timing of the routines of such computing operations as multiplication and division. This is not the case however in Simon, because every computing operation is completed in the machine cycle following the designating of the operation in computer register 4. If we should wish to do multiplication or division on Simon, we would need to program it by means of the instruction tape, and the use of arithmetical and logical operations.

For example, what would we punch in the tape if we wanted the operation "selection" (code 0011) to go into computer register 4 (code 1011), the register which chooses the operation the computer is to utilize? We would punch, in three successive lines of tape:

Entry	Punched Holes	Meaning
1	11011	Get ready to receive in CR 4, and clear IR 1 and 2.
2	00011	Put operation selection in IR 1.
3	00000	Transfer out of IR 1.

This description of Simon and its operations is incomplete. We have not touched on the wiring of the front panel, so that different types of automatic or manual operations are possible. We have not covered the two other uses of the 5th hole in the program tape, only mentioning that at time 2 in the machine operation it is used for optional reset of the input register. We have not given the specifications of the rectifiers, capacitors, and the rest of the parts list. The basic relay used is Allied Control or close 24-volt, 300-ohm, d.c., airplane-type, bought on war surplus, with either 4 poles double-throw or as many poles as can be found. We have not discussed the subject of coding problems for the machine, nor the range of problems that the machine can do. We have not discussed the ways in which this machine can be further expanded to do useful work. If there is sufficient interest, these matters may be covered to some extent in the final article of this series.

Construction of Simon

Simon as an idea came into existence at the end of 1947, when, at a meeting of the Association for Symbolic Logic in New York, Simon was discussed by Edmund C. Berkeley, one of the two joint authors of this series of articles. Next Simon became the third chapter in Berkeley's book *Giant Brains or Machines that Think* (John Wiley and Sons, 1949), with the purpose of being a simple introduction on paper to the same type of computing circuits used in the big mechanical brains.

Simon as a real machine was begun in November, 1949, and was finished in April, 1950. The cost of materials was about $270, and the labor for wiring actually paid for amounted to another $270. The balance of the labor, design, engineering, mechanical work, etc., was contributed; if it had been paid for, it would have amounted to about $3,000. Simon was actually constructed by three men: William A. Porter, a skilled technician who had much to do with the construction at Harvard University of two big mechanical brains built there, Mark II and Mark III, and Robert A. Jensen and Andrew Vall, two Columbia University graduate electrical engineering students. Jensen is the joint author of this series of articles.

In the next article we shall begin the discussion of electronic brains. **—END—**

"John had to be converted too. He's color blind."

How an Electronic Brain Works

Part VIII—The flip-flop circuit and other methods used to store information in electronic computers

By EDMUND C. BERKELEY and ROBERT A. JENSEN

IN THE previous articles of this series, we have described a simple example of an electric brain made up of relays.

We have shown that we can have a complete, and rather interesting, miniature electric brain made up essentially of the following: 16 registers, each consisting of two relays, which may store

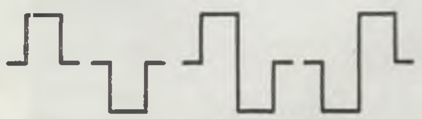


Fig. 1—Pulses like these carry information in electronic computer circuits.

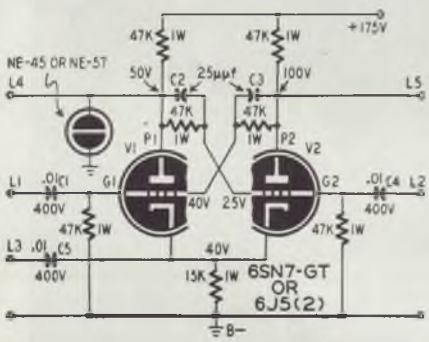


Fig. 2—A flip-flop circuit. V1 is conducting when the voltages are as shown.

numbers 0, 1, 2, 3, (in binary, 00, 01, 10, 11), or operations "addition," "negation," "greater-than," "selection" (codes 00, 01, 10, 11); and 1 register, consisting of 5 relays, which stores instructions (codes 00000, 00001, ... 11111).

There are many problems which require such vast amounts of computation that they have never been attacked by human mathematicians. Relay brains have been able to handle some of these problems. But even a relay brain is too slow for the biggest problems, such as computing the aiming direction of a missile that will intercept another one (like a buzz-bomb) in time to shoot it down. The fastest that an ordinary relay can operate is about 5 or 10 milliseconds. However the fastest that an electronic tube can operate is better than a microsecond.

So, with our background of understanding how a relay automatic computer operates, we can now set out to see how an electronic brain can operate that would compute a thousand times faster than a relay brain. We must translate the ideas we have been dealing with out of the language of relays into the language of electronic tubes.

It must be remembered that no one has yet constructed a complete operating miniature electronic brain. Consequently most of the information here given is derived from work that has been done with the giant electronic automatic computers.

Information

How shall we make electronic equipment express information? In electronic computers, just as in relay computers, the basic piece of information is a binary digit, a yes or a no, a 1 or a 0, a tube conducting or not conducting, the presence or absence of a certain change of voltage, etc. It is much easier and more direct to construct an electronic computer that operates in the pure binary system than it is to construct one that operates in the decimal system.

There are several main systems for representing information. The first system is that 1 is represented by a pulse of voltage (either positive or negative) at a certain time, and 0 is represented by the absence of a pulse at such time. A second system is that 1 is represented by a positive pulse of voltage, and 0 is represented by a negative pulse of voltage. Here, the absence of a pulse at a time when a pulse is expected becomes a useful indication that something has gone wrong. A third system makes use of a pair of pulses: a positive followed by a negative denotes a 1, and a negative followed by a positive denotes a 0. The second and third systems are more reliable, and for that reason are used in some automatic computers; but the first system is simpler and has the advantage that the presence of information is indicated by a pulse that may be either positive or negative. Fig. 1 shows the pulse arrangements. The minimum duration of a pulse depends on the time of operation of electronic tubes, which range in the neighborhood of 1 microsecond to 1/20 of a microsecond in most computer circuits.

Because of the speed of operation of an electron tube, many automatic electronic computers operate serially—that is with a bus consisting of just one line along which all pulses travel. One of those finished recently (the Bureau of Standards Eastern Automatic Com-

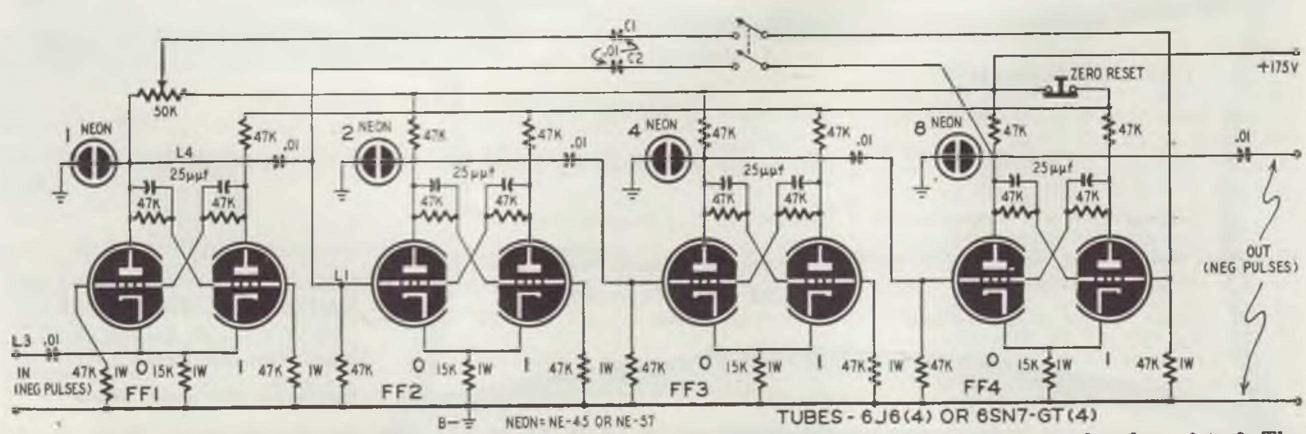


Fig. 3—A binary decade counter. This string of flip-flops is used to count up to and store any number from 0 to 9. Then it resets, passing an impulse to a similar unit, which acts as the "tens" bank, and so to any desired number of decades.

puter, in Washington, D. C.) is of this kind, and has a one-line bus along which all pulses travel in a series. It works with numbers of 45 binary digits, and its speed of operation is 1,100 additions or subtractions per second, or 330 multiplications or divisions per second.

But there is one electronic computer (Whirlwind I, built by the Servomechanism Laboratory at Massachusetts Institute of Technology) which uses a 16-line bus along which pulses travel in parallel. It works with numbers of 15 binary digits and an algebraic sign, plus or minus. Because of the parallel bus, this machine is able to reach the speed of 30,000 multiplications per second.

Flip-flop

As we have already seen, the first thing we have to do with a pulse of information is to store it—hold it in such a way that we can use it later. For example, in Simon, we would feed a number into a set of relays and hold it there till we had fed another number into another set. Then we could add the two numbers together, or compare their size, or do something else with them. Without this *memory*—this ability to store a number until it is needed—a computer would be so limited that it would be almost useless.

Large computers may use relays to store information till needed, or they may use electron tubes. One type of computer (such as the International Business Machines Electronic Selective Sequence Calculator, located at 57th Street and Madison Avenue, New York City), uses tubes for very short storage periods and relays for information that has to be stored for longer periods. Information that must be remembered indefinitely is placed on long punched rolls of paper.

If we can make the counter count up to 1,001 (9 in the decimal system) and then reset when the next pulse is received, we have a *decade* counter, or one that counts in tens. (The pulse that resets the first decade is fed to a second as an integer, so that two decades can count to 99.)

One of the standard ways of storing a pulse of information electronically is the *flip-flop* circuit (see Fig. 2). It consists of two triodes (in one envelope, a 6SN7-GT, for example; or in two envelopes, two 6J5's for example), and it has two stable states: (1) triode V1 conducting and triode V2 not conducting; (2) triode V1 not conducting and triode V2 conducting.

Now let us take a look at the operation of the flip-flop. Suppose we put a negative pulse (or voltage drop) on the input lead L1. Capacitor C1 transmits this pulse, and it goes to grid G1. The negative pulse reduces the current through triode V1, and so produces a rise in the plate voltage on plate P1. This positive pulse is at once transmitted through capacitor C2 to grid G2, and starts triode V2 conducting. As it starts to conduct, the current flow lowers the plate voltage on plate P2. This voltage drop is at once transmitted as a negative pulse through capacitor C3 to grid G1, and makes grid G1 even more negative, tending to cut off triode V1. This process continues and rushes to conclusion in less than a few microseconds. Triode V2 is then conducting (at saturation) and triode V1 is cut off. As long as the power supply is on, the flip-flop records and stores the fact that a negative pulse came along on lead L1. The neon lamp lights when triode V2 is conducting, and indicates a "1" stored in the flip-flop. The lamp is off when a "0" is stored.

Another consecutive negative pulse on L1 will now have no effect, but a positive pulse on lead L1 or a negative pulse on lead L2 will cause the flip-flop to change back to its original state. In regard to lead L2, we can see that a similar description applies symmetrically.

Now let us consider lead L3, which runs through capacitor C5 to the cathodes of both triodes. A negative pulse on this lead will cause a change of state in the flip-flop, no matter which of the two triodes is conducting. So the flip-flop will actually count, 0, 1, 0, 1, 0, 1, 0, 1, 0 . . . . . . , depending on the number of negative pulses that come in on this lead. A positive pulse will have no effect. However, the shaping of these pulses may require additional components, while the shapes of the pulses used on leads L1 and L2 are not too critical.

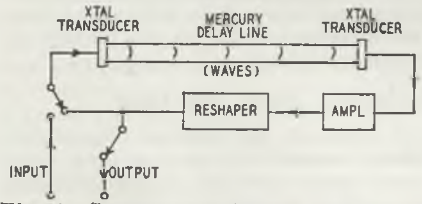


Fig. 4—Storage circuit with a mercury delay line and two crystal transducers.

This flip-flop is very similar to one actually used in the ENIAC, the first big automatic electronic computer. ENIAC was finished in 1946 at the Moore School of Electrical Engineering, and is now operating at the Ballistic Research Laboratories, Aberdeen, Maryland. (The ENIAC flip-flop is described further in a paper "High-Speed N-Scale Counters," by T. K. Sharpless, in *Electronics*, March 1948.)

Now suppose we hitch three more flip-flops in succession to this first flip-flop (see Fig. 3). We impulse FF1 by lead L3, so that (1) it changes state on every pulse, and (2) it puts out on output lead L4 alternately a positive pulse and a negative pulse. We connect lead L4 on flip-flop 1 to lead L1 on FF2; then only negative pulses on L4 cause flip-flop 2 to change state. We make similar connections between FF2 and FF3, and between FF3 and FF4. Then we have a binary counter that will count 0000, 0001, 0010, 0011, 0100, etc., up to 1111. Capacitors C1 and C2 are used to trip the counter back to 0000 after holding 1001. This is the principle used in the 4-tube counter decade described by John T. Potter in *Electronics* of June, 1944.

For storing one binary digit of information, a 1 or a 0, a flip-flop is decidedly expensive. Consequently it is used only in those parts of an automatic computer where a great deal of traffic with information requires the convenience and justifies the expense.

Delay line

Another scheme for storing information in an electronic computer is the *sonic delay line* (see Fig. 4). A sonic delay line consists of material which will transmit pulses as a series of molecular vibrations, more slowly than the usual wire conductor. It may be made of a solid, or of liquid in a tube, or air, in the case of an echo.

For example, think of a long rope, one end in your hand, the other end tied, and the whole rope pulled fairly taut. You shake your hand quickly, and a wave (or pulse) will travel down the rope. As soon as one wave (or pulse) has been started down the rope (or delay line), another can be started almost immediately after it, and the second will follow the first one without interfering with it. Thus what is basically needed for a sonic delay line is any medium down which a pulse may travel. As long as the medium is built suitably, the pulse will not die out until it is needed.

Now in the case of the rope, when the wave or pulse reaches the end of the rope that is tied, the pulse is reflected, and a wave of reversed phase travels back. Of course, reflected waves are not wanted, and a sonic delay line, contrary to the rope, is designed so that reflections are eliminated or rendered unimportant.

We therefore can see that information is stored in a sonic delay line as a series of pulses and absences of pulses, a pattern of 1's (the presence of pulses) and 0's (the absence of pulses).

The pattern is retained by sending it around and around a loop. How do we "write" information in a sonic delay line? We feed a series of pulses and absences of pulses into it. How do we "read" information from the line? We send it along two channels, one the channel back to the front end of the delay line, so that the information will circulate and be remembered, and the other the channel into the part of the computer where we want to use the information, say, into a bank of flip-flops. How do we "erase" information in the delay line? We interrupt the circulating loop long enough so that all the pulses are eliminated.

A type of sonic delay line that has been used quite widely in electronic computers is the *mercury delay line* or *mercury tank* (see Fig. 4). At each end of a long tube filled with mercury is a quartz crystal in contact with the mercury. When a voltage pulse is applied to the quartz, its shape changes (piezoelectric effect). The quartz agitates the mercury, and sends a ripple down the tank. The ripple is picked up at the far end by another quartz crystal and there converted back into an electric pulse. It is somewhat smeared, and so

is amplified and reshaped, in the loop shown. A mercury tank several feet long will store a pattern of about a thousand pulses about a microsecond apart. So it is equivalent to a thousand flip-flops.

Shorter delay lines are useful for remembering numbers in a computation. Very short delay lines—such as a "one-pulse delay line"—are not of the sonic type but of the electrical type, and consist of a network of capacitors and inductors and are an essential element of the computing circuits of some of the electronic brains. The delays obtainable with a small electrical delay line are from a fraction of a microsecond to a few microseconds.

Mercury delay lines for use in giant automatic electronic computers are being manufactured by the Eckert-Mauchly Computer Corp., Philadelphia, Raytheon Manufacturing Co., Waltham, Mass., and other organizations. The details of their operation in many cases involve trade secrets, although in the literature there is some information about operating details.

It is apparent that if the power goes off, or even flickers, the computer's memory of the pattern of the pulses may be erased in whole or in part. For this reason some of the builders of electronic computers do not like to rely on delay line storage.

Magnetic drum and tape

A compact and efficient device for storing pulses—and storing them even if the power goes off—is the *magnetic drum.* This is a rotating cylinder which may be of different sizes, but one size that has been used is about 12 inches long and 6 or 8 inches in diameter. It usually rotates at 1,800, or 3,600, or 7,200 r.p.m. The drum may be of brass or aluminum, and is coated with a compound containing magnetic particles. The compound is put on like paint, with a brush or spray gun.

Mounted almost touching the drum are small electromagnets or *magnetic heads*, very similar to magnetic tape-recorder heads. The separation between the two sides of the pole piece may be about 3/1000 of an inch, and the distance of the magnetic head from the drum may be perhaps 2/1000 of an an inch. A pulse of current passed through the magnetic head causes the recording of a magnetized spot on the rapidly rotating surface of the drum. A positive pulse will cause, say, a north-south magnetization, and a negative pulse vice versa.

The number of magnetized spots in an inch along the perimeter around the drum may be 20 to 60, and the number of channels side by side along the length of the drum may be 8 or 10 to the inch. So if we can get 400 magnetized spots to the square inch, and put them on 250 square inches of drum surface, the magnetic drum will be the equivalent of 100,000 flip-flops. Magnetic drum storage is being used in automatic electronic computers that have been made by the Harvard Computation Laboratory, Cambridge, Mass., Engineering Research Associates, St. Paul, Minn., Northrup Aircraft Co., Hawthorne, Calif., and other organizations. Here again very little if any information has been published revealing the exact know-how for reading, writing, and erasing pulses on drums, and the normal procedure has apparently been for each laboratory to work out its own technique.

In general, what is needed, of course, is something that will act like the holes in Simon's program tape to connect the pulses that the pickup head senses, into effective use in the computer at the exact instant the head passes over the point which holds the desired pattern of information.

To accomplish this, the information that is on a channel of the drum is always being "read" by the pickup head whose duty is reading; but the pulses are allowed into the computer only when the computer calls for admission, and an electronic switch allows them to come in. The timing is naturally very important, and is based on a series of permanently-recorded equally-spaced pulses on the drum, called the master clock channel.

Magnetic tape wound on a reel, such as is used in magnetic sound recording, has proved to be an important, useful, and reliable means for "slow storage". By this we mean storage of large quantities of information where a relatively long time (seconds) for access to the information is permissible. Six channels across a quarter-inch width of tape, and 100 magnetized spots to an inch of length seems to be a realizable objective.

Magnetic tape is well accepted as about the best device for input, output, and slow storage in an automatic electronic computer. Raytheon Manufacturing Co. is offering for sale multichannel magnetic heads for reading (sensing), writing (recording), and

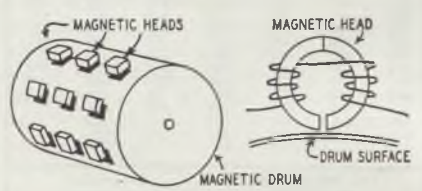
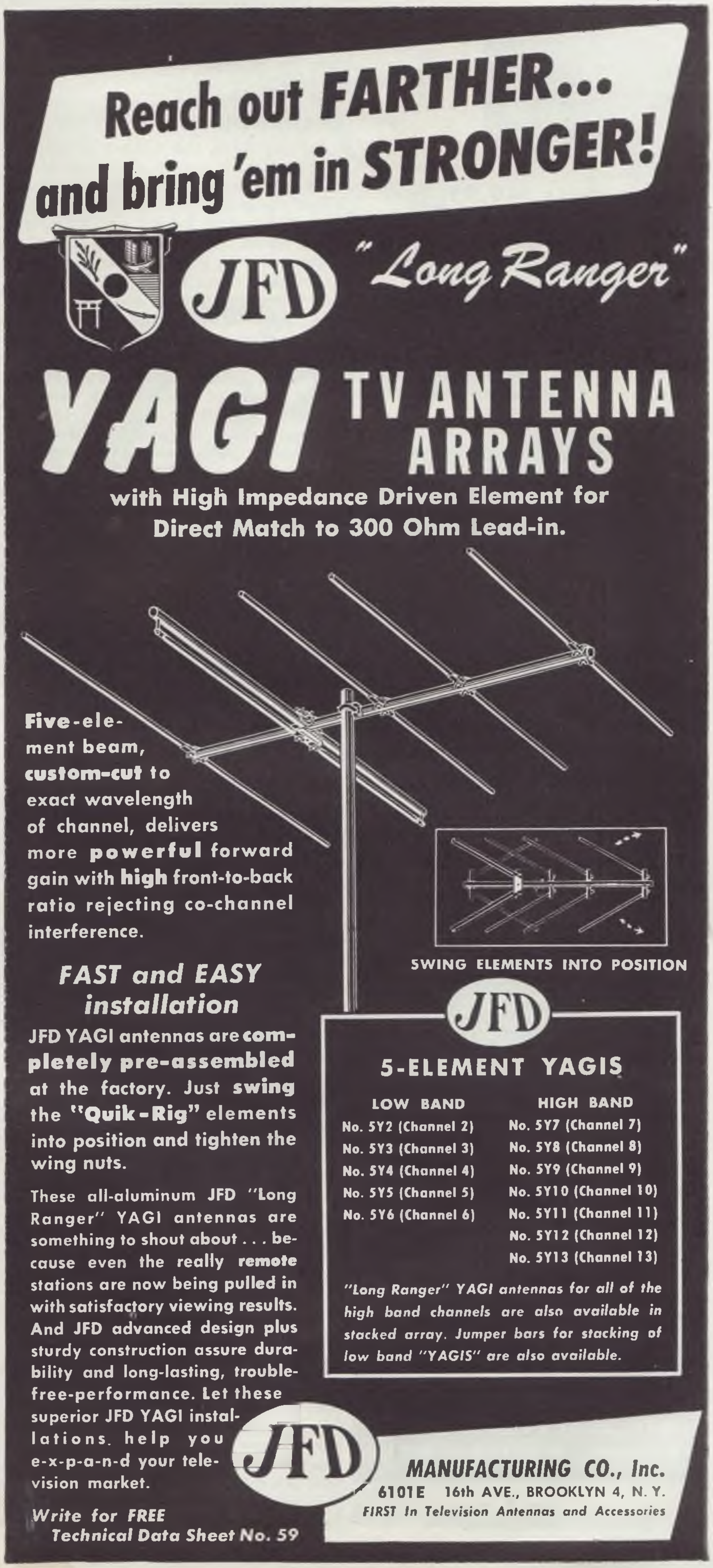


Fig. 5—Magnetic drums make one of the most efficient ways of storing information in smaller electronic computers.

erasing (eliminating) pulses on magnetic tape, and so undoubtedly would furnish details and know-how for using their heads.

Both the delay line and the magnetic drum have the disadvantages that the computer has to wait for the information to become available. If a delay line or a channel on a drum is storing 20 numbers, and the one you want has just gone by, you have to wait for the other 19 numbers to go by before you can pick up the number you want. In the case of a magnetic drum, you could of course put on additional reading heads and read out from that part of the periphery of the drum which is nearest to the location of the number



you are interested in. But that can be troublesome. Why not have all the numbers in storage accessible practically at once?

Electrostatic storage

In one form, the method of electrostatic storage uses a dielectric screen in a cathode-ray tube. The beam that scans the screen divides it into a pattern of, say, 32 by 32, or 1,024 separated spots. Information is stored on these spots as the presence or absence of certain electric charges. The spots are written on or read out or erased by one beam of electrons. The electric charges that have been recorded on the screen are held in their places and prevented from leaking away by another beam of electrons, a so-called "holding beam."

Electrostatic storage has proved to be rather a ticklish technique to master. F. C. Williams at the University of Manchester in England has succeeded in using electrostatic storage in the automatic electronic computer built there. Also, the Servomechanisms Laboratory at Massachusetts Institute of Technology is installing some electrostatic storage memory in Whirlwind I, which will still further raise its speed of 30,000 multiplications a second. Certainly, no miniature automatic electronic computer would be expected to make use of electrostatic storage, whereas a small magnetic drum would be a logical choice for its memory.

(continued next month)

LISTENERS BUY FM PROGRAM

To prevent their favorite program from being interrupted by commercials, listeners to an FM music series donated practically as much as the station would have received had the series been handled on the regular commercial basis.

The station with the music-loving audience was WABF (FM) of New York City, and the program the New Friends of Music concert series. The audience was informed that "it would be inconsistent with the tone and mood of the concerts" to have them sponsored by advertisers. Instead the listeners were asked to subscribe to the series by sending the station any sums they wished. During the first weeks of the program it was announced that more than 2,000 listeners had responded, sending WABF more than 90% of the amount which would have been realized had the program been sold in the regular manner to a commercial sponsor.

ELECTRONIC MONEY COUNTER

Eight tons of old currency—mostly one dollar bills—removed from circulation daily by the Department of Treasury is now being counted by an automatic electronic machine. Developing the counter was a difficult job as the old money is badly wrinkled and dog-eared.

How an Electronic Brain Works

Part IX—Some electronic circuits for computers and how they are used for adding and subtracting

By EDMUND C. BERKELEY AND ROBERT A. JENSEN

IN THE previous article we began the discussion of an electric brain to be built around electronic tubes instead of relays. We discussed the storage information in the form of the state of a flip-flop, or pulses circulating in a delay line, or magnetized spots on a magnetic surface, or charges on the screen of an electrostatic storage tube.

But how do we compute? As soon as we have arranged to read, write, and erase information at electronic speeds, we need to consider how to compute with electronic elements.

For computing purposes, a unit of information is represented as a pulse, either a rise and fall of an otherwise constant voltage, or else a fall followed by a rise. We will call the first kind a positive pulse or a 1, the second kind a negative pulse or a –1, and the absence of a pulse a 0. See Fig. 1.

In a computer, the pulses are usually of a standard duration, and may be for example ⅕ of a microsecond long and spaced ⅘ of a microsecond apart. In this case the pulse repetition rate would be 1 megacycle per second. In some computers, 1 and –1 pulses are both treated as the presence of information, the binary digit 1, the logical truth value 1, or "yes"; while 0 is treated as the absence of information, the binary digit 0, the logical truth value 0, or "no."

Phase inverter

The first computing element we need to consider is a *phase inverter*. In computer work, a phase inverter changes a positive pulse to a negative one, or a negative pulse to a positive one, that is, "inverts" the pulse. See Fig. 2. In this figure, and in Figs. 3 to 8, part *a* is the circuit diagram; *b* is its block diagram representation which we use for convenience; and *c* is a function table that indicates what the circuit does. Any grid-controlled electronic tube can act as a phase inverter.

Logical AND circuit

The next computing element we need to consider is called a "logical AND circuit." This is one of the meanings of the electronic term "gate." See Fig. 3. In this circuit, a pulse appears on the output line if, and only if, two pulses come in simultaneously on two input lines.

A tube with two grids, normally cut off with either one or no pulses, is one of the forms which a logical AND circuit can take. The reason for the word "and" is that we have a pulse on output line C if and only if we have a pulse on input line A *and* on input line B. This (with emphasis on the idea "both") is the regular meaning for "and" in logic. This type of circuit may take many forms with and without electronic tubes.

Logical OR circuit

Another computing element is called a "logical OR circuit", sometimes called "buffer." It allows a pulse on the output line if a pulse comes in on either one or both of the two input lines. See Fig. 4.

A tube with two grids, which is normally conducting, is one of the forms which a logical OR circuit may have, although there are others. The reason for the word "or" is that a pulse is on output line C if a pulse is on input line A or if a pulse is on input line B, *or both*. This nonexclusive meaning of the word "or" is its regular meaning in logic.

Logical EXCEPT circuit

Another computing element is called a "logical EXCEPT circuit," or inhibitory gate. In this a pulse is allowed out on the output line if a pulse comes in on a specified one of the two input lines *except* if a pulse comes in at the same time on the other input line. See Fig. 5.

The circuit shown in Fig. 5 will act as a logical EXCEPT circuit. Its constants are chosen so that when A is not pulsed, whether or not B is pulsed, still there is no output on line C. If A is pulsed and B is pulsed, the two pulses coinciding in time and of opposite phase eliminate the pulse on line C. If A is pulsed and B is not pulsed, then the pulse goes on through. Other circuits besides that shown in Fig. 5 are of course possible.

Electrical delay lines

The computing section of an electronic computer also uses an electric delay line of very short delay, such as one pulse time, or a few pulse times. A circuit that does this appears in Fig. 6. These are different from the long sonic

Fig. 1—Pulses are used in electronic computers to represent information. A positive pulse may mean +1, a negative pulse –1, and no pulse 0. Other pulse combinations are sometimes also used.

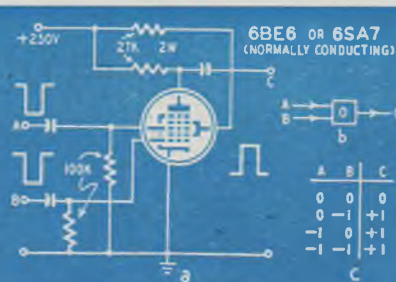


Fig. 3—A logical AND circuit, or gate. Output pulse appears at C only when inputs appear simultaneously at A and B.

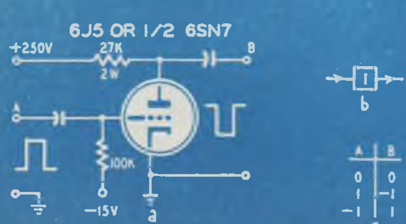


Fig. 2—A phase inverter circuit for computers. Part *a* is the circuit, *b* is its symbol for block diagrams, and *c* is a function table for this circuit.

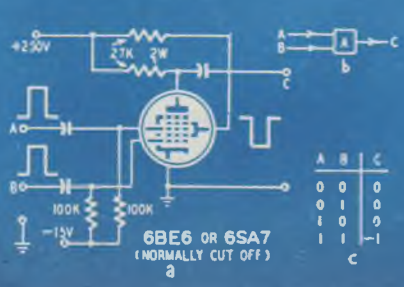


Fig. 4—Logical OR circuit, or buffer.

delay lines such as the mercury tanks described in the previous article, because the purpose of the short delay line is not storage but computation. Short delay lines are important because pulses sent into the various parts of an electronic computer must arrive at the various points just when they are needed. For example, in the Bureau of Standards Eastern automatic computer, delay times are figured to hundredths of microseconds and pulses are timed to be safely within the planned intervals.

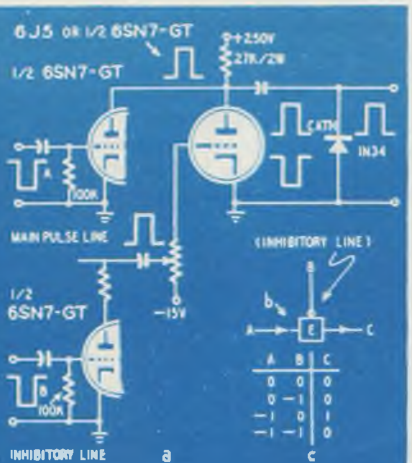


Fig. 5—A logical EXCEPT circuit. The pulse input at A appears at output C except if there is a pulse at input B.

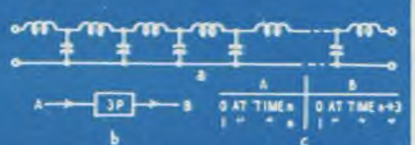


Fig. 6—Electrical delay line as used in computers. The 3 in the block symbol denotes a delay of three pulse times.

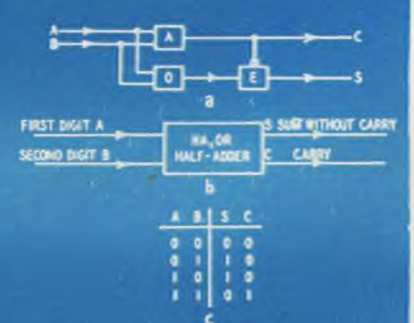


Fig. 7—This half-adder circuit gives the result of adding two binary digits.

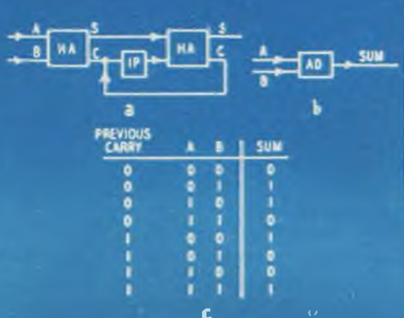


Fig. 8—An adder circuit is made up of two half-adders and a one-pulse delay.

Half-adder

Now how do we take these various computing elements and begin to do computing with them?

The first thing is to assemble these elements so that we can add two binary digits. Suppose there are two input lines A and B, and either one may bring in a binary digit that may be 1 or 0. Suppose that we have two output lines, one of them S, that will give us the sum without carry, and the other C, that will give us the carry. The function that we want to express is the result of adding two binary digits: A + B = C, S, where 0 + 0 = 00, 0 + 1 = 01, 1 + 0 = 01, and 1 + 1 = 10. See Fig. 7.

To make a half-adder circuit, one logical AND circuit, one logical OR circuit, and one logical EXCEPT circuit, combined as shown in Fig. 7-a, are sufficient.

Adder

But we are not finished, because a previous addition may have given a carry that has to be taken into account. The circuit which will perform complete binary addition is called an *adder*. See Fig. 8.

Now let us trace through the adder circuit with some numbers and see what actually happens in the sequences of pulses on the several lines in the circuit.

The digit 1 will represent a pulse (assumed to be positive or negative as the circuit requires), and the digit 0 will mean absence of a pulse at the proper time. At the same time the digits 1 and 0 will represent information that we desire to compute with.

Suppose we write a binary number (or more generally any set of binary digits) in the ordinary way (with the smallest ranking digit at the right) on any circuit line where the pulses are traveling from left to right. Then the binary number will be attended to as a pattern of pulses by the circuit in just the sequence from right to left that we ordinarily deal with in arithmetic. At the same time the number will show the sequence of pulses in the order that they are handled in the circuit.

As an example of using the adder, let us add 101 (one 4, no 2 and one 1 in binary, or 5 in decimal) and 1011 (one 8, no 4, one 2, and one 1 in binary, or 11 in decimal). We write the two numbers on the input lines A and B (See Fig. 9) and now we set out to see what happens.

At the first pulse-time, the pulse (the 1) on the A line and the 1 (another pulse) on the B line go into half-adder No. 1, and give rise to no pulse on the S line (sum without carry) and a pulse on the C line (carry). The 0 on the S1 line goes into the second half-adder without delay; but the 1 on the C1 line goes into the one-pulse delay and so it is held back one pulse-time. As a result, at the first pulse-time, 0 and 0 go into the second half-adder; and so its output is 0 for the first digit of the true sum, and 0 for the carry. The 0 for the carry circles round the loop and comes up to the entrance of the one-pulse delay.

At the second pulse-time, 0 and 1 go into the first half-adder, and give rise to a 1 on the S1 line and a 0 on the C1 line. The 1 on the S1 line goes into the second half-adder without delay. Now the delayed previous carry (with no conflict from the absence of pulse that came around the loop) now issues from the one-pulse delay. So 1 and 1 now enter half-adder No. 2, and from it issues a 0 on the sum line S2 and a 1 on the carry line C2 which circulates around the loop, and enters the one-pulse delay so it will be ready for the next pulse time.

At the third, fourth, and fifth pulse-times, each of the proper operations takes place similarly, and so we get out of the second half-adder exactly the sum that we desire.

Subtracter

Now how do we manage to subtract? A circuit that will subtract is shown in Fig 10, using the constituents of an adder, and a logical EXCEPT circuit. The word "minuend" means "the number to be diminished." The word "subtrahend" means "the number to be subtracted."

Let us test this circuit by subtracting five from eleven, or in binary subtracting 101 from 1011. The pulses appear in succession on each of the lines in the diagram, as shown. By following through the circuit, remembering what each stage does, we see that exactly the right answer, 0110 or six, appears on the output line marked "difference."

Acknowledgement is made to Henry W. Schrimpf for a number of the circuits and ideas in this article.

In the next article we shall take up the multiplication and division of binary numbers using electronic circuits and begin the discussion of the control of an electronic computer.

(continued next month)

Fig. 9—The numbers on this adder show how 0101 is added to 1011 to get 10000.

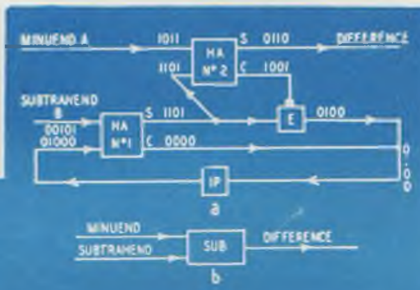


Fig. 10—Diagram of a subtracter. This one is taking 101 from 1011 to get 110.

How an Electronic Brain Works

Part X—The product of two binary numbers.

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

IN THE two previous articles we discussed how a brain built around electronic tubes can store information, add, and subtract; and we drew up schematics for an adder and a subtracter (RADIO-ELECTRONICS, June 1951, page 39).

The basic components which are required for an electronic brain have been labeled as we have indicated in the table below:

Symbol	Name
A	an AND circuit, or gate
O	an OR circuit, or buffer
E	an EXCEPT circuit, or inhibitory gate
nP	a delay line, delaying pulses for n pulse times
F-F	an electronic switch, or flip-flop

Now let us see how we can use these components to design an electronic multiplier, which can multiply two binary numbers together and give the product.

A circuit which will multiply is shown in Fig. 1. It contains 17 components, which are connected by lines bearing arrows, along which pulses travel in one direction. This circuit has seven input lines, numbered terminal 1 to terminal 7, and one output line, num-

* Author: *Giant Brains*, John Wiley & Sons, Inc.

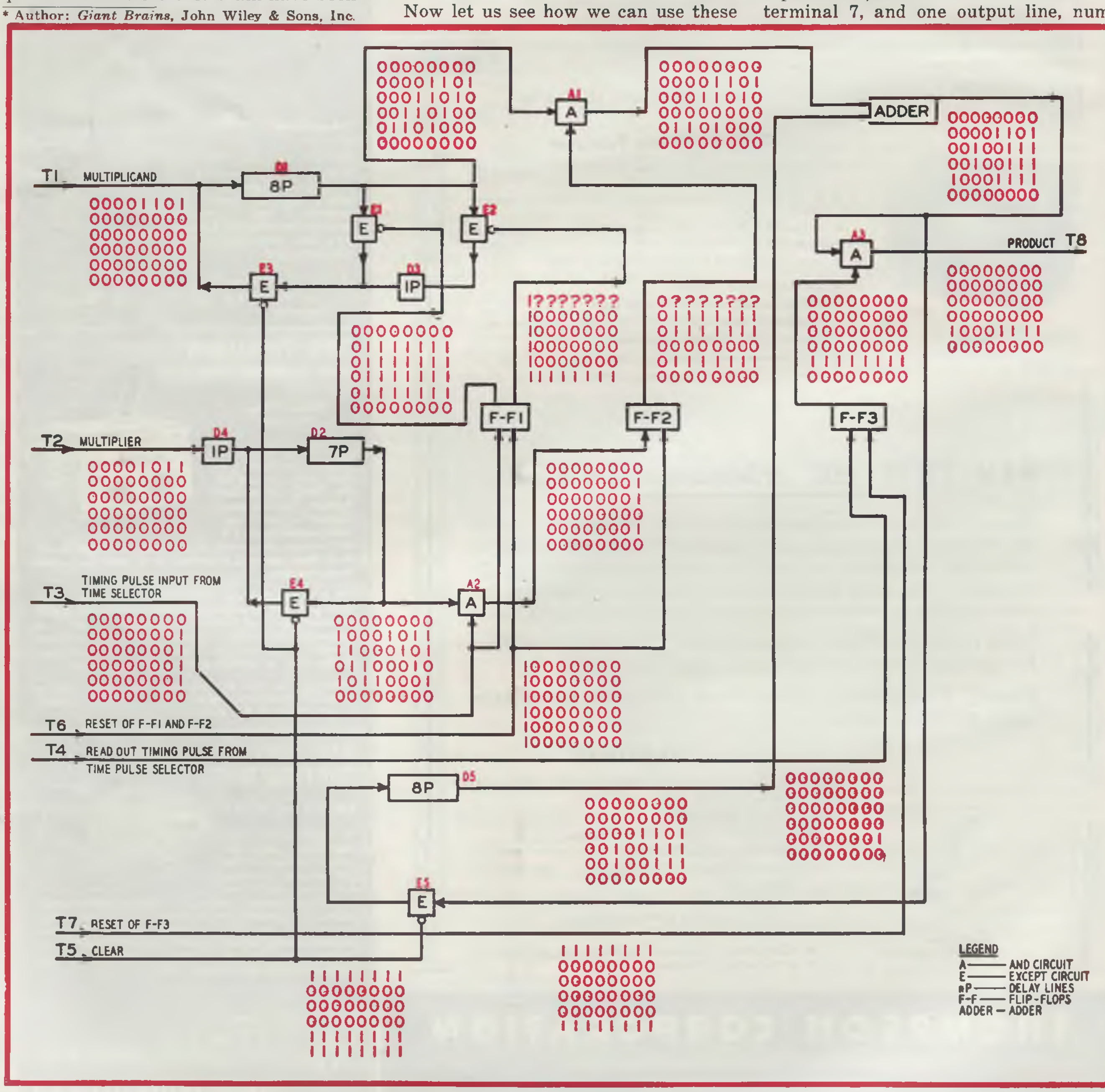


Fig. 1—This multiplier circuit has 17 components. There are 7 input lines and 1 output line where the product appears.

bered terminal 8. The result of the multiplication, the product, comes out on the output line, and the two factors of the multiplication, the multiplicand and the multiplier, come in on lines T1 and T2.

All information travels through this circuit in pulses serially along one line, and so the scheme of the circuit could operate for binary numbers as large as 50 or 100 digits, just by altering the length of the delay lines. To explain the circuit operation, we shall assume that we are going to multiply two four-digit binary numbers. We shall multiply binary number 1101 (one-one-oh-one, or one eight plus one four plus no two's plus one one, or thirteen) by the binary number 1011 (one-oh-one-one, or one eight plus no four's plus one two plus one one, or eleven). The answer will be 10001111 (the sum of 128, 8, 4, 2, and 1, or 143, which of course equals 13 times 11). Since the answer will be eight binary digits long it is convenient to specify that the multiplier circuit operates in six machine cycles, each handling eight pulses corresponding to eight binary digits. The reason for eight pulses is obvious; that for six cycles will appear later.

This example of multiplication, by the way, is the same example worked out with relays that was discussed in the December, 1950, article of this series, Part III. The comparison of the multiplication there using relays and the multiplication here using electronic tubes is instructive, if any reader cares to look it up.

In the first cycle, the multiplicand 1101, in the form of a series of eight pulses, or a binary number, 00001101, enters along line T1. The number shows the timing of these pulses *represented in order from right to left, the earliest pulse being at the right.*

Note well that on any line where the direction arrow points to the right, a regular binary number will represent pulses in the proper time sequence. We have avoided writing binary numbers, or series of pulses, on lines directed to the left because the sequence of pulses and the regular way of writing digits in the number would be opposite. For later cycles, we write the binary numbers of eight digits one under another. This helps to see without difficulty the pattern of as many as 40 or 48 consecutive pulses.

The series of eight multiplicand pulses go into the 8-pulse-time delay line D1, and they will accordingly issue in cycle 2 as 00001101, and apply for admission at the door of AND circuit A1, EXCEPT circuit E1, and EXCEPT circuit E2.

FLIP-FLOP F-F1 controls admission to the EXCEPT circuits E1 and E2. Because of the pattern of reset pulses (see later discussion) coming to this flip-flop, the EXCEPT circuit E1 is inhibited, while E2 is not inhibited. Consequently, the pulses go through E2, and through the one pulse-time delay line D3, through the noninhibited EXCEPT circuit E3, and round the loop through D1 again. As a result, *the multiplicand is shifted, to the left one digit* and the output from the delay line D1 of the numbers applying for admission to the AND circuit A1 is therefore as follows:

Cycle	Number
1	00000000
2	00001101
3	00011010
4	00110100
5	01101000
6	00000000

In the first cycle also, the *multiplier* 1011 enters as a series of pulses 0000-1011 into the one pulse-time delay line D4 and the seven pulse-time delay line D2, and goes round the loop through delay line D2 in such a way that there issues from the loop the following numbers:

Cycle	Number
1	00000000
2	10001011
3	01100010
4	11000101
5	10110001
6	00000000

The first digit is carried around to the eighth digit's place by D2 and E4 in each cycle. Because the delay line is only 7 pulses long it advances the pulses one step. In cycle 3, for instance, we no longer have the number 1011 at the right-hand side, but 101, the other one having been attached to the end (left side) of cycle 2, where it replaces the zero that would otherwise be there. The *first* digit in any one of these numbers, in the sequence in which they pass through the circuits, is at the *right*, and the *last* one at the *left*. This follows our regular practice in adding or multiplying on paper, but not in writing down figures. These numbers now continue and apply for admission at the door of AND circuit A2.

The other line going into AND circuit A2 is line T3, called "timing-pulse input from time-pulse selector," and the pulses (or numbers) that come in on this line are as follows:

Cycle	Number
1	00000000
2	00000001
3	00000001
4	00000001
5	00000001
6	00000000

These are standard pulses and would accordingly apply for any four-diget multiplier.

Their purpose is, we shall see later, to allow a single digit of the multiplier to go through the AND circuit, and the timing is such that just the proper digit of the multiplier is allowed to go through the AND circuit.

Now an AND circuit performs the operation of "both," i.e., logical multiplication, operating digit by digit, pulse by pulse. Consequently, the numbers that issue from the AND circuit are the logical product, according to the following very simple table, of the numbers coming in:

	0	1
0	0	0
1	0	1

So the numbers (or pulse series) coming out of AND circuit A2 are now:

Cycle	Number
1	00000000
2	00000001
3	00000001
4	00000000
5	00000001
6	00000000

The output of AND circuit A2 leads to the "set" side or left side of FLIP-FLOP 2. The "reset" side or right side of FLIP-FLOP 2 receives a series of reset pulses (see later discussion) as follows:

Cycle	Number
1	10000000
2	10000000
3	10000000
4	10000000
5	10000000
6	10000000

These are the two inputs of the flip-flop. So what comes out?

A flip-flop conducts on one side and not on the other side, depending on the last pulse that has come in on either side. Specifically:

Input: Last Pulse Received on		*Output:* Conducting on	
Left Side	Right Side	Left Side	Right Side
1	0	1	0
0	1	0	1

The operation of a flip-flop is ambiguous, not defined, if the last pulse comes in simultaneously on both left and right sides. So before any inputs are acceptable to a flip-flop there must be no cases of 1 and 1 occuring at the same time.

Applying this rule we can calculate the output of FLIP-FLOP 2 as follows:

	Input		*Output*	
Cycle	Left Side	Right Side	Left Side	Right Side
1	00000000	10000000	0???????	1???????
2	00000001	10000000	01111111	10000000
3	00000000	10000000	01111111	10000000
4	00000001	10000000	00000000	11111111
5	00000001	10000000	01111111	10000000
6	00000000	10000000	00000000	11111111

Wherever we see a zero on one side of the flip-flop output, we must see a one on the other side. This is merely equivalent to saying that one side or the other, but not both sides, of the circuit are conducting at any given time.

We may note that the output of the right side of FLIP-FLOP 2 is not used in this circuit, but is available and might be useful somewhere else in the computer. The output of the left side, however, goes to AND circuit A1 and is there very useful. The flock of 1's on cycles 2, 3, and 5 allows the shifted multiplicand to go through A1 in the case of the first, second, and fourth multiple, and this is precisely what we want. Furthermore, the fourth multiple of the multiplicand will never be more than 7 digits, and so the change of the flip-flop at time 8 in each cycle does not interfere with the multiples of the multiplicand.

ADDER at cycle 2 takes in the first multiple of the multiplicand, adds it to 0 coming in on the other line, and passes it around the loop through the noninhibitory EXCEPT circuit E5 to the partial sum register, the 8 pulse-time delay line D5. One cycle later this first multiple comes out, going to the adder. But now it is matched in time (cycle 3) with the second multiple, and the adder produces the sum of the two of them, and sends that around the loop to the partial-sum register again. In this way, the multiplication proceeds by successive addition of selected shifted multiples, and a breakdown of the problem will then appear as follows:

Multiplication	Partial Sum at Cycle
1101	
× 1011	
1101	2
+ 1101	
= 100111	3
+ 0000	
= 100111	4
+ 1101	
= 10001111	5

We must now get the final answer 10001111 out of the circuit into some other part of the machine where we can make use of it. We use FLIP-FLOP 3, operating it with a "readout timing pulse from the time selector" on the left side as follows:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	00000001
6	00000000

together with a reset pulse on the right side of the flip-flop will then be as follows:

Cycle	Number
1	11111111
2	00000000
3	00000000
4	00000000
5	00000000
6	11111111

Here is where—and why—we need a cycle 6 in the multiplier circuit: to be certain that no stray pulses come out on the output line after we have received the product. The output of FLIP-FLOP 3 accordingly will then appear as follows:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	11111111
6	00000000

This is the input on one side of the AND circuit 3. The other input coming from the adder is:

Cycle	Number
1	00000000
2	00001101
3	00100111
4	00100111
5	10001111
6	00000000

and as a result it may be seen the output is:

Cycle	Number
1	00000000
2	00000000
3	00000000
4	00000000
5	10001111
6	00000000

In order to clear the multiplicand, the multiplier, and the partial-sum registers, the EXCEPT circuits 3, 4, and 5 are used. The input to all of them comes in on line T5, and is in the usual listing, as follows, and after this the multiplicand, the multiplier and partial-sum registers are ready for new signals.

Cycle	Number
1	11111111
2	00000000
3	00000000
4	00000000
5	11111111
6	11111111

In the same way, to reset the flip-flops, the input of standard pulses from the time-pulse selector is as follows:

Cycle	Flip-Flop 1 and 2	Flip-Flop 3
1	10000000	11111111
2	10000000	00000000
3	10000000	00000000
4	10000000	00000000
5	10000000	00000000
6	10000000	11111111

The circuit is, of course, a block diagram. To convert it into working hardware, a good deal of attention will have to be given to the shape of pulses, the precise detailed timing of them, microsecond delays due to the length of cables along which they are traveling, etc. Descriptions or circuits of the components inside the blocks were given last month.

It should be emphasized once more that there are many ways in which the circuit can be improved, or otherwise designed. Some improvements will doubtless occur to readers of this article.

For example, F-F1, E1, and E2 can be avoided if we give up optional storage of the multiplicand unchanged in register D1; that is, if we make sure that the multiplicand and the multiplier do come in simultaneously to the multiplying circuit. Actually, in a serial computer it will be more convenient to have the multiplicand come in at one time and wait in the multiplicand register until the multiplier is ready and comes in at a later time. The circuit in Fig. 1 has provided for this second more convenient scheme.

Acknowledgment is made to Henry W. Schrimpf for many of the features of the multiplying circuit in this article.

Division

If we replace the adder by a subtractor, and put in some kind of circuit for comparing the divisor with the partial remainder from successive subtraction, and in still other ways modify the circuit of Fig. 1 so that we can use the method of dividing that was illustrated in Part IV (January, 1951) of this series for a relay computer, then we can design a division circuit for binary numbers.

But divisions do not occur as often as multiplications. Studies made some time ago in the Harvard Computation Laboratory indicated that a division occurred once for about every fourteen multiplications. So why spend money on equipment for division, which you may use less than 8% of the time, if we can get division some other way?

You can get the result of dividing the number M by the number K if you multiply M by the *reciprocal* of K, which is 1/K. And you can get the reciprocal 1/K by a process using multiplication. First, find some kind of reasonable first approximation X_1; second, use the following formula over and over:

$$X_n = X_{n-1} \times (2 - KX_{n-1}).$$

It can be shown that a reasonable first approximation X_1 is any number between 0 and 2/K.

For example, suppose that we want to find 1/3, which we know is equal to .33333 to five decimals, and suppose that we start off with a guessed first approximation .5. Then:

$X_1 = .5$ (guess)
$X_2 = .5\ [2 - 3(.5)] = .25$
$X_3 = .25[2 - 3(.25)] = .3125$
$X_4 = .3125[2 - 3(.3125)] = .33203$
$X_5 = .33203[2 - 3(.33203)] = .33333$

Thus we can get a fine result with just a few steps in multiplication. We can have our computer carry out division by programming our computer. We yield time; we save equipment. Over and over again this kind of compromise occurs in computer design. Division by successive multiplication, with special circuits for sensing good first approximations (which saved on the number of multiplications necessary), was the process built into the Harvard Mark II computer, now at the Naval Proving Ground, Dahlgren, Va.

This kind of scheme for use in automatic computers was first proposed, we believe, by Prof. Howard H. Aiken of Harvard.

(continued next month)

AUTOMATIC-FOCUS KINE

New picture tube with built-in-automatic focus has been announced by Du Mont. It eliminates the need for a focus coil, a focus control or other focusing methods used in magnetic and electrostatic kinescopes.

The new tube, the 17KP4, is an all-glass rectangular, magnetic- deflection, electrostatic-focus tube, with a gray-filter face plate and an external conductive coating. It is designed to work with an anode voltage of 16,000 and a maximum negative bias value of 125 volts.

"Left, I said! Wassamatter, aren't you tuned in?"

How an Electronic Brain Works

Part XI—Beginning the timing and control circuits

By EDMUND C. BERKELEY* and ROBERT A. JENSEN

IN the last three articles, we discussed series of pulses representing information traveling along lines (conductors) in an electronic brain (electronic digital computer). In Part VIII, we told how a series of pulses could be stored in a delay line until they were wanted somewhere else in the computer. In Part IX, we added two series of pulses or binary numbers, 0101 and 1011, in an adder (see Fig. 9 of Part IX). In Part X, we multiplied 1101 by 1011 in a multiplier. To do so we made use of (1) six machine cycles of eight pulse-times each, and (2) five different sets of control pulses.

The questions that arise now are:

1. Where do those series of control pulses for controlling the multiplier come from?

2. How can we obtain them and any other control pulses that we may need or want?

3. If we have two registers storing numbers, how can we take numbers out of those registers, put them into an adder or multiplier, and then store the results?

In this article, we begin the study of the answers to these questions; in other words the study of the timing and control of an electronic computer.

*Author: *Giant Brains*, John Wiley and Sons, 1949.

Timing pulse selector

In the operation of the multiplier one of the sets of control pulses which we used was the following:

Cycle	*Pulse Series*
1	0000 0000
2	0000 0001
3	0000 0001
4	0000 0001
5	0000 0001
6	0000 0000

The name of this particular set was the Multiplier Digit Timing Pulse, since it enabled us to select the desired successive digits of the multiplier at the time we wanted each one. Each of the eight-digit binary numbers in the pulse series is read from *right to left*, and the 1 indicates a pulse at the pulse-time corresponding to the position of the 1 and the 0 indicates no pulse at that pulse-time.

We obtain this series of pulses (and also any desired series of timing pulses) with a *Timing Pulse Selector*, an assembly of delay lines, flip-flops, and a single initial pulse. See Fig. 1, a diagram of a Timing Pulse Selector.

In the Timing Pulse Selector, the loop at the left side of Fig. 1 containing the seven-pulse delay line repeats a pulse pattern every eight pulses. If this loop were divided into eight 1-pulse delays, we could choose in every cycle a desired pulse in any one of the eight binary digit positions that we might be interested in. In our particular case, we are interested only in selecting the eight pulse-time in each cycle. So instead of installing seven 1-pulse-time delays (as would be more general), we have installed a single 7-pulse-time delay. A lead from the output of this delay gives us the pulse series for the Reset Timing Pulse for F-F1 and F-F2 in the multiplier of Part IX.

The six 8-pulse-time delay lines (marked 8P) in the loop across the top of the diagram of Fig. 1, enable us to choose the first pulse-time in any one of the six multiplier cycles that we may be interested in. At the start of each successive cycle No. 1, 7, 13, and so on, a pulse appears on line L1. At the start of each cycle No. 2, 8, 14, and so on, a pulse appears on line L2. At the start of each cycle No. 3, 9, 15, and so on, a pulse appears on line L3, and so on.

To construct the desired series of control pulses for the Multiplier Digit Timing Pulse we connect output lines L2, L3, L4, L5 from this loop (through crystal diodes to prevent back signals) to the outgoing line T3. The only times when a pulse is allowed out on L2, L3, L4, and L5 is at the start of cycles 2, 3, 4, and 5, so that we obtain just the timing pulses which we desire, the Multiplier Digit Timing Pulse. Similarly, we can obtain the Readout Digit Timing Pulse on line T4.

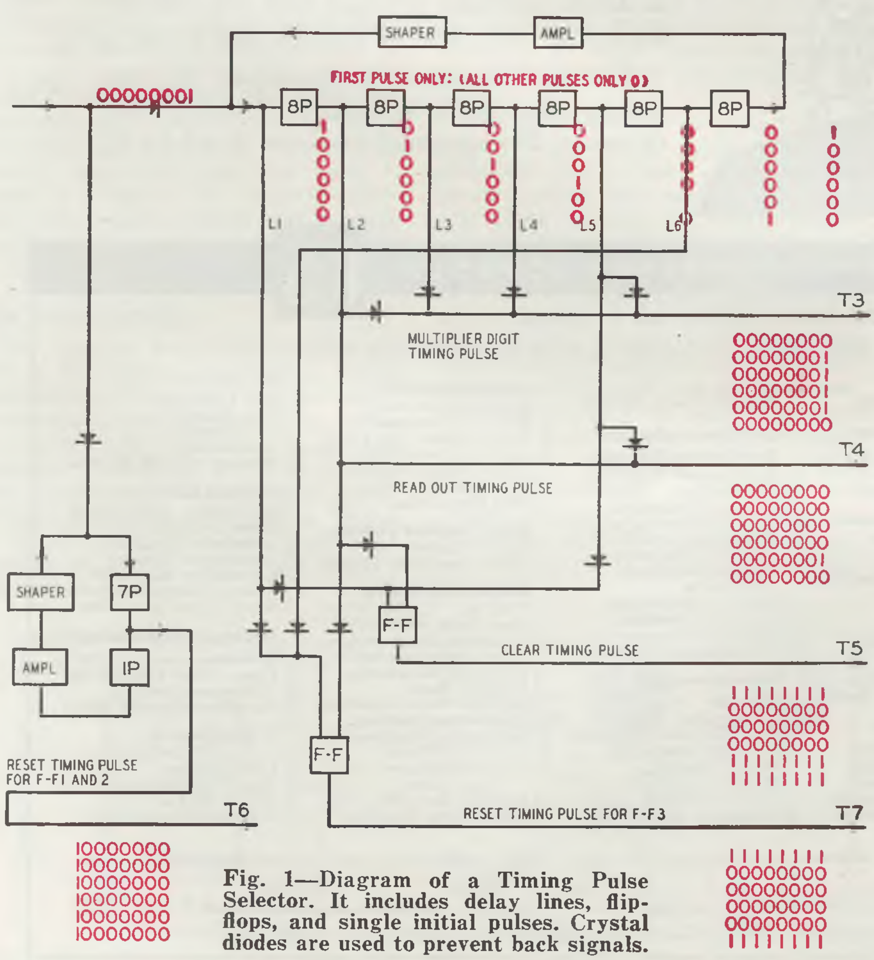


Fig. 1—Diagram of a Timing Pulse Selector. It includes delay lines, flip-flops, and single initial pulses. Crystal diodes are used to prevent back signals.

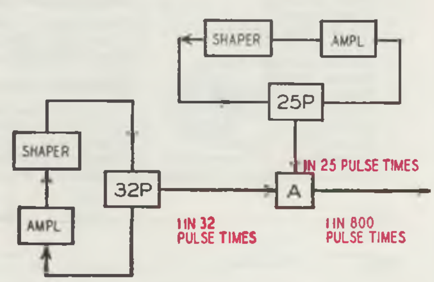


Fig. 2—Two circulating delay line loops.

The Clear Timing Pulse which we need for line T5 into the multiplier of the last article is a series of control pulses as follows:

Cycle	*Pulse Series*
1	1111 1111
2	0000 0000
3	0000 0000
4	0000 0000
5	1111 1111
6	1111 1111

To obtain this configuration of pulses —and the 1111 1111 can be one continuous signal, not necessarily a series of 8 separate positive pulses—we make use of a flip-flop. We need to set the flip-flop at the start of cycle 1 and cycle 5, and reset the flip-flop at the start of cycle 2. This we can do by appropriate connections through crystal diodes from (1) L1 and L5, and (2) L2, respectively. The reason we are using two set pulses and only one reset pulse is that when the machine is turned on, the first set pulse from line L1 is required.

Ordinary electrical delay lines are good for delays of about 50 pulse-times. After that, the pulses become indistinct. To get accurate control pulses at much longer intervals, the type of circuit shown in Fig. 2 may be used. Here there are two circulating delay line loops, one repeating at intervals of every 25 pulse-times and the other repeating at intervals of every 32 pulse-times. These two numbers have no common factor, so the pulses applying for admission to the AND circuit will match each other only once in 25 × 32 times, or once in 800 pulse-times.

We can now assemble some storage registers and some computing facilities, and begin to obtain a whole electronic computer. In Fig. 3 we have drawn a schematic diagram showing:

- a Main Storage, or memory, of 8 registers of eight binary digits each, in a 64 pulse-time delay line;
- a Computing Section, which can add, subtract, or multiply;
- an A-Register, and B-Register, which can take in numbers to be operated on in the Computing Section;
- an Operation Register, which can take in the instruction telling the operation to be performed; these last three are the Computing Section Input Registers;
- and a Result Register, which will hold the result of the operation produced by the Computing Section.

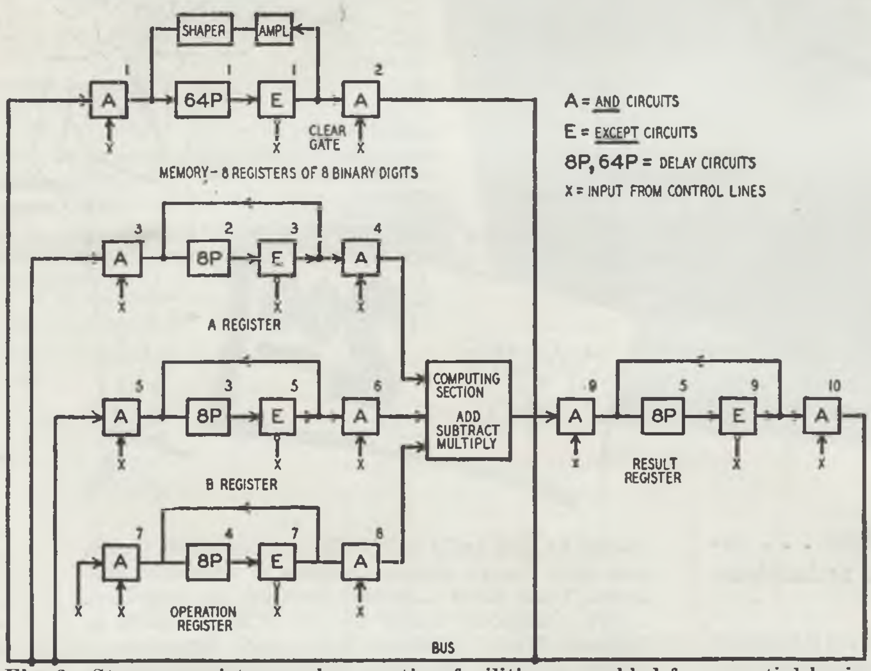


Fig. 3—Storage registers and computing facilities assembled form partial brain.

At the bottom of the diagram is the *bus*, a line along which numbers can travel from any register in the Main Storage to any Computing Section Input Register, and back again from the Computing Section Result Register to a register in Main Storage. Permission to any number to travel on the bus depends on the opening of the AND circuit. The operation of the circuit hinges on the control lines running to the ten AND circuits and the five EXCEPT circuits. These 15 control lines and the 16th line, the input of the Operation Register, all marked as ending with x, lead to controlled timing pulses and signals, and are related to the programming of an electronic computer.

A typical problem

How does this partial schematic of an electronic computer operate? In the Chart we show how this assembly would carry out a typical problem like:

> Take the number in the 2nd register, and the number in the 5th register, multiply them, and put the result in the 7th register.

The Chart lists minor cycles and major cycles. What are they? A minor cycle consists of eight pulse-times, beginning with the first digit of an 8-binary-digit number, and ending with the last one. A major cycle consists of eight minor cycles, a time sufficiently long for all the numbers in the Main Storage to circulate once completely around their loop. In general, to get any desired number out of Main Storage, we have to wait until it comes round the loop and grab it then.

To carry out our problem, the first thing is to get the number in the 2nd register out of Main Storage. This we do by waiting until the 2nd minor cycle comes along; we then open AND circuit No. 2, and let the series of pulses come out into the bus. But there is no way of storing them there, so we simultaneously open AND circuit No. 3, allowing this number to go into the A-Register delay line. This will do us no good unless we clear out of the A-Register any number already there, so we energize EXCEPT circuit No. 3. In this way we succeed in making the transfer we desire. This is step 1.

In step 2, we proceed in almost the same way, and transfer the number in the 5th register, using minor cycle 5. In step 3, we assume that the command for making the computing section multiply is available at minor cycle 8, and transfer it at that time.

Having filled the input of the Computing Section with the information it is going to use, in step 4 (the time is now minor cycle 1 within major cycle 3) we send the numbers and the operation into the computer, and assuming that not more than 6 or 7 minor cycles are necessary for the multiplication, the result comes out into the Result Register at the time major cycle 4, minor cycle 1.

Since the answer is to be stored in the 7th register, we do not have to wait for the next major cycle, but in this same 4th major cycle we can send in the number at the time of minor cycle 7. So at that time we transfer from the Result Register through the bus into the 7th register of Main Storage by opening the appropriate AND circuits, A10, A1, and operating EXCEPT circuit No. 1 to clear out any previous information in this register.

The chief topics remaining to be discussed are function tables, programming, and input and output. These we shall begin in the next article.

(Continued next month)

CHART OF MAJOR AND MINOR CYCLES

Process *Step*	Major *Cycle*	Minor *Cycle*	AND Circuits *Conducting*	EXCEPT Circuits *Inhibiting*
1	1	2	2, 3	3
2	2	5	2, 5	5
3	2	8	7	7
4	3	1	4, 6, 8	—
5	4	1	9	9
6	4	7	10, 1	1

How an Electronic Brain Works

Part XII—Pulse patterns rearranged and programmed

BY EDMUND C. BERKELEY and ROBERT A. JENSEN

IN THE last few articles, we have shown how an electronic brain can store information, add, subtract, multiply, divide, and arrange different timing pulses and select them. We have also shown how, when the right control pulses are provided, an electric brain can pick numbers out of storage, run them into the computer, produce computed results, and put the results back in storage.

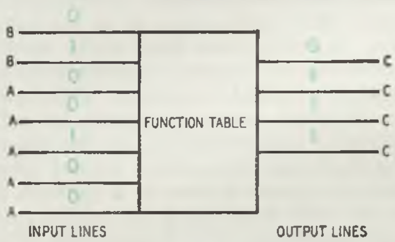


Fig. 1—The function table converts incoming pulses into other pulse patterns.

In this article we shall finish with such of the theory of electronic computer construction as we intend to cover.

We must emphasize again that not all the work in the design of a miniature electronic brain has been done yet, not by a long way. For example, one of the elements that will turn up in such a computer over and over again is the AND circuit; there are many different ways of making one; energy spent on perfecting an AND circuit useful for all parts of the machine would be well repaid; but we shall not investigate that subject here. Furthermore, the authors have as yet constructed only a miniature relay Simon, not a miniature electronic Simon; no one has yet made a miniature automatic electronic sequence-controlled digital computer (to give it its full name).

It is interesting to note one reader of RADIO-ELECTRONICS, Thomas P. Weir, W7GDM, of Powell, Wyoming, has written us that he has started construction of "a small machine using tape and a pulse frequency of 60 cycles"; and it may well be that one of the readers of RADIO-ELECTRONICS will be the first man to make a miniature automatic electronic computer.

Function tables

The term *function table* means an arrangement of equipment which will take in any one of a number of patterns of pulses and will put out any one of a number of other patterns of pulses, in such a way that any outgoing pattern is precisely determined by the incoming pattern. See the block diagram in Fig. 1, where the pattern 0100100 on seven input lines is being converted into the pattern 0111 on four output lines. Other names for function-table are *matrix*, *coding device*, *coder*, or *decoder*.

In an electronic computer, function tables may be used in many different ways, in fact wherever a mathematical function of pulses is desired. Examples are: 1—a built-in multiplication table; 2—conversion of binary digits to decimal digits, or of decimal to binary; 3—built-in tables of first approximations to reciprocals, so that hardware for dividing can be left out and the accurate reciprocals can be calculated by successive multiplications (see article X of this series); 4—built-in tables for first approximations to square roots, logarithms, etc., with the same kind of scheme for successive approximation;

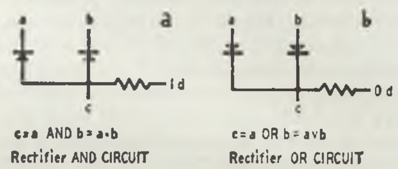


Fig. 2—Function tables use rectifiers.

5—conversions of orders (i.e., one kind of a set of pulses) given to the computer into control signals (i.e., another kind of a set of pulses) for gates (AND circuits), so that the machine can be automatically controlled.

One of the most convenient elements to use in a function table is a crystal

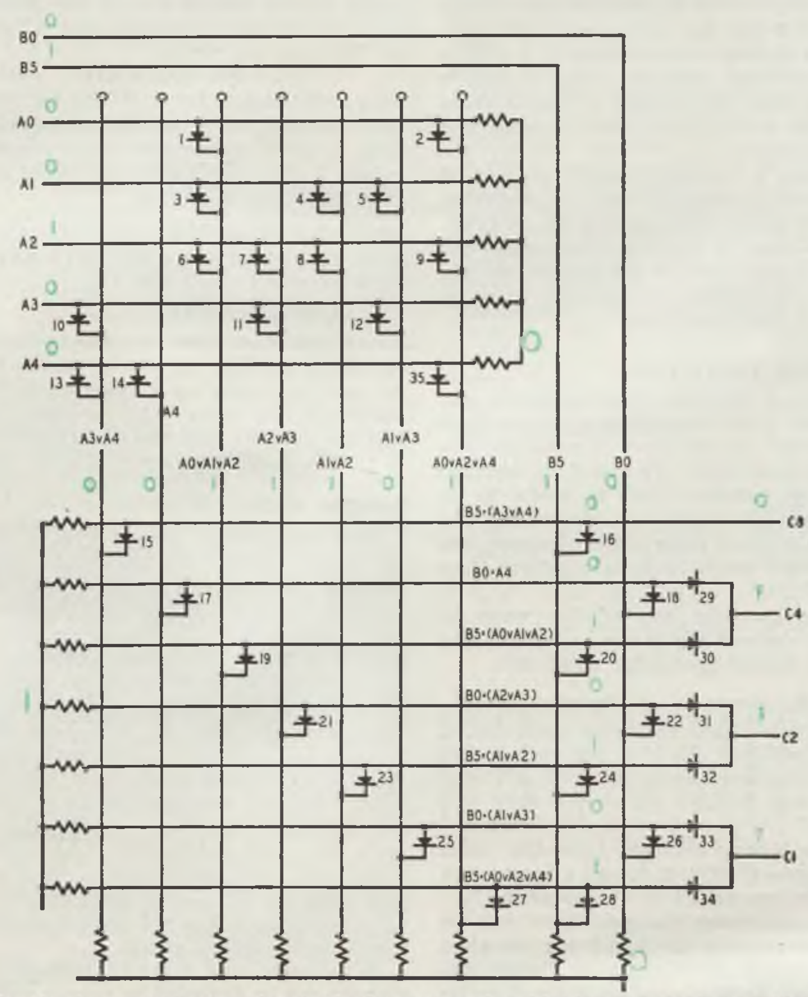


Fig. 3—A complete function table for converting biquinary notation to binary.

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.

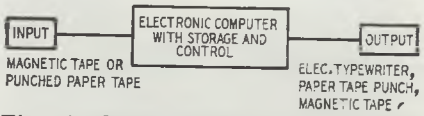


Fig. 4—Computer input and output.

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.

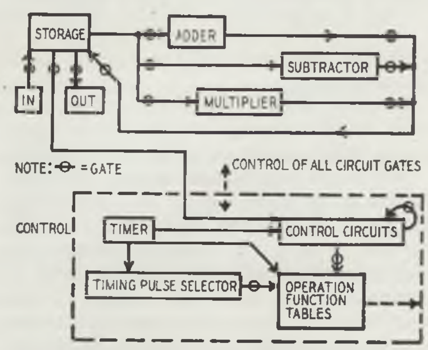


Fig. 5—Block diagram of the computer.

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

CONTROL and surrounded by a dashed line is the group of circuits which control the gates all through the machine, both outside itself and inside. The unit marked OPERATION FUNCTION TABLES converts various pieces of information, particularly instructions that program the operation of the computer, into gate controls. An example of this kind of situation in a relay computer is shown in article V and in an electronic computer is shown in article XI. Control over the connection of circuits to the bus results in the control of the flow of information through the machine.

Programming

The term *programming* is now generally used to mean giving or arranging a sequence of instructions under which an automatic sequence-controlled calculator operates. In Simon, and other "old-fashioned" automatic computers, the sequence of instructions is external to the machine, comes along on a loop of tape, and is read order by order. But in many of the new computers, and in modified examples of the old computers (including the Harvard IBM Mark I machine at Harvard University), the instructions are almost all internal, within the machine, and can be called for by the machine itself as it requires them.

Typically, the control of one of these newer types of machines is in a register called the "program register" or the "control register." On each machine cycle this register contains an "order," information which is interpreted by the machine to control the opening and closing of the gates (such as shown in Fig. 5). In general an order consists of three parts:

n, the number of the order, which will generally be from 000 to 999.

k, the kind of order, such as "transfer into . . .," and which may be denoted by 0 to 9.

r, the number of a register referred to, which may be from 000 to 999, if the machine has a thousand registers.

Many programs can be covered in 400 orders, and they may be stored in 400 of the 1,000 registers. The other 600 registers are used to store numbers involved in the calculation. The machine is designed so that regularly order number n is stored in register number n.

Necessary orders

Now what are the kinds of orders sufficient to carry on the functioning of a modern versatile electronic computer? They are surprisingly few. See the order chart.

The list shown in Chart 1 may be made longer or shorter. We have here used the device of calling for operation specifications out of certain storage registers. In this way the machine may be made very versatile. On the other hand, by lengthening the list of kinds of orders, to include directly addition, subtraction, multiplication, and other operations among the kinds of orders, some machine time can be saved, and some programming directness gained.

The most important of all the orders is the conditional transfer of program, kind of order 8 (also to some extent, kind of order 7). This order is the heart of flexible automatic control, and is one of the main reasons for asserting that these machines are intelligent—are brains—because they can adjust their activity to indications revealed as they compute.

In the next and last article of this series, we shall discuss the application of a good deal of what we have been talking about, to the Bureau of Standards Eastern Automatic Computer, SEAC.

—end—

CHART I

If k equals:	*Machine Performs Operation:*	*and For the Next Order Goes to:*
0	Transfer operation in register r to computer's *operation* register	
1	Transfers number in register r to computer's register A	Next numbered order
2	Transfers number in register r to computer's register B	"
3	Transfers number in computer's result register C to register r	"
4	Transfers number on input tape to register r	"
5	Transfers number in register r to output tape	"
6	None	"
7	None	The order numbered r
8	None	The order numbered r, provided some special register (No. 069 for example) contains 1; otherwise the next number numerically.
9	Stops	The order number stored in register r, provided special register 069 contains 1; otherwise, the next order numerically.

HIGH-EMISSION CATHODE

Electron-tube cathodes with exceptionally high emissivity are made possible with the use of boron compounds, it is announced by Dr. J. M. Lafferty of General Electric.

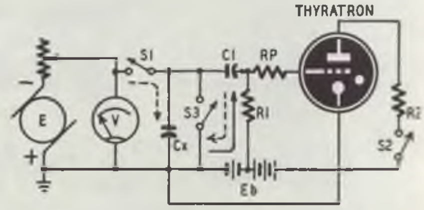
Among the best cathodes are those in which a thin layer of good emitting material is deposited on a metal cathode which can be raised to a higher temperature than would be practical with a cathode made of more volatile surface material. Thus some transmitting tubes (and some early receiving tubes) used a tungsten cathode which contained thorium.

The weakness of this type of cathode construction is that with heavy currents the surface layer may entirely evaporate before it can be replaced from the interior. By embedding the surface material in a matrix of boron, Dr. Lafferty has found that this "stripping" of the cathode can be avoided. The boron forms a compound with the thorium or similar emissive element, and has such a strong affinity for it that as soon as an atom of the emitting material evaporates, the boron immediately draws up a new one from the metal beneath.

CAPACITOR TESTER

An intermittent capacitor can be very troublesome in radio or electronic circuits. Breakdowns may occur momentarily and at random, so tracing is difficult. The Bell Laboratories* have designed an instrument which indicates momentary breakdowns in capacitors. In the figure Cx is the unit under test.

First S2 is closed to put anode voltage on the thyratron. Bias on the starter element is set just under the firing point. Test voltage is applied across Cx when S1 is closed. Electrons flow into this capacitor as shown (*dotted arrow*). There is also a flow into the shunt circuit R1-C1, as the second *dotted arrow* shows. The drop across R1 adds *negative* bias to the starter element, so the tube cannot ionize.



If the test unit breaks down even for a duration of a few milliseconds, C1 discharges through the test capacitor and R1. The *full arrow* indicates that the drop across the resistor now adds *positive* voltage to the starter. Ionization follows. The tube lights and continues to conduct until S2 is opened.

The thyratron may be one of the WE 313 series or one similar to the OA4-G type. RP and R2 are protective resistors.

*Bell Lab Record, January 1951.

How an Electronic Brain Works

Part XIII—SEAC, the 800-tube Thinking Machine

By EDMUND C. BERKELEY and ROBERT A. JENSEN

IN the last five articles, we discussed the organization of an automatic electronic digital computer (for short, an electronic brain). In the seven before those, we described the organization of a similar machine made out of relays. Now in this article—the last one of the series—we shall take a good look at one of the big automatic electronic digital computers. This one is the National Bureau of Standards Eastern Automatic Computer, SEAC. It began to operate in May, 1950.

Fig. 1 is a picture of SEAC in one of the buildings on the grounds of the National Bureau of Standards in Washington, D. C. But, like any photograph of a machine that handles information, this photo does not tell very much.

SEAC came to be built as a result of three factors. The first of these was the demonstration that giant electronic computers could be built and made to work. ENIAC, now at the Ballistic Research Laboratory in Aberdeen, Md., proved that. ENIAC started working in 1946 and has been working ever since.

The second factor was the decision of the office of the Air Comptroller, Department of the Air Force, in early 1948, that it needed a big automatic computer for the study of supply programs for the Air Force. The question is: what sort of materials and personnel should be supplied and trained at what times, so that the United States should have the best possible Air Force? This is a prodigious planning problem, and it must be planned. The Air Force set up Project SCOOP (Scientific Computation of Optimum Programs) for this purpose.

The third factor was private industry's continuing delay in constructing giant automatic electronic digital computers. Two firms received government contracts for big electronic computers in 1946-47. One finished its first machine in 1951, the other company has not yet finished its first big computer.

So the Air Force and the Bureau of Standards got together in early 1948; and by the end of 1949, the machine system, the circuits, and the construction techniques had been settled on, and construction begun. The machine SEAC was completely assembled in March 1950. After some preliminary computing, it ran its first significant practical problem in May, 1950. This short period of 20 months was a fine accomplishment. The main reason was the decision to stick to well-established techniques.

The appearance of SEAC

When you walk into the room where SEAC is, and see the front of the machine, it looks like Fig. 1. It is about 15 feet long, 5 feet deep, and 8 feet high. The nine racks are: 1, 2, 3, the control unit; 4 and lower 5, the arithmetical (or computing) unit; upper half of 5 and all of 6, the time pulse generator; rack 7, upper half, the clock pulse generator; rack 7 lower half, spare; racks 8 and 9, controls and power supplies, and also the circuits for the input-output systems using punched tape (this is the first of the planned input-output systems). But there is more to SEAC.

If you walk around the machine to the right, and go behind, you find another large bulky cabinet about 5 feet wide, 3 feet deep, and 7 feet high. This contains the machine's "memory," or rather the first installment of the memory of the machine, the "serial memory." This cabinet consists of 64 units like the one shown in Fig. 2—a long glass tube filled with mercury, mounted in an aluminum holder, and connected to recirculation amplifiers; these units are "mercury delay lines" (see Article VIII). Each of these tubes stores eight units of information called "words." The words are usually numbers, but may be sets of logical indications, or instructions to the machine.

If you take a good look at a certain part of the machine, you will see an assembly like that shown in Fig. 3. This is an inside view of a section of the main part of SEAC. It shows, first, long fiber tubes containing short lengths of electrical delay lines—"rapid memory"; and, second, clusters of germanium diodes mounted in the tube bases—used as high-speed electronic switches.

The register which stores instructions (which we called the "program register" in our last article) is shown in Fig. 4. Fiber turrets support the resistors in the gating circuits; these turrets make the resistors accessible, and allow them to cool. In aluminum shields on the rear of the chassis, there is a coiled electrical delay line which can delay for 48 microseconds. When the computer is working, this delay line stores an instruction word (consisting of 48 binary digits) in continuous circulation.

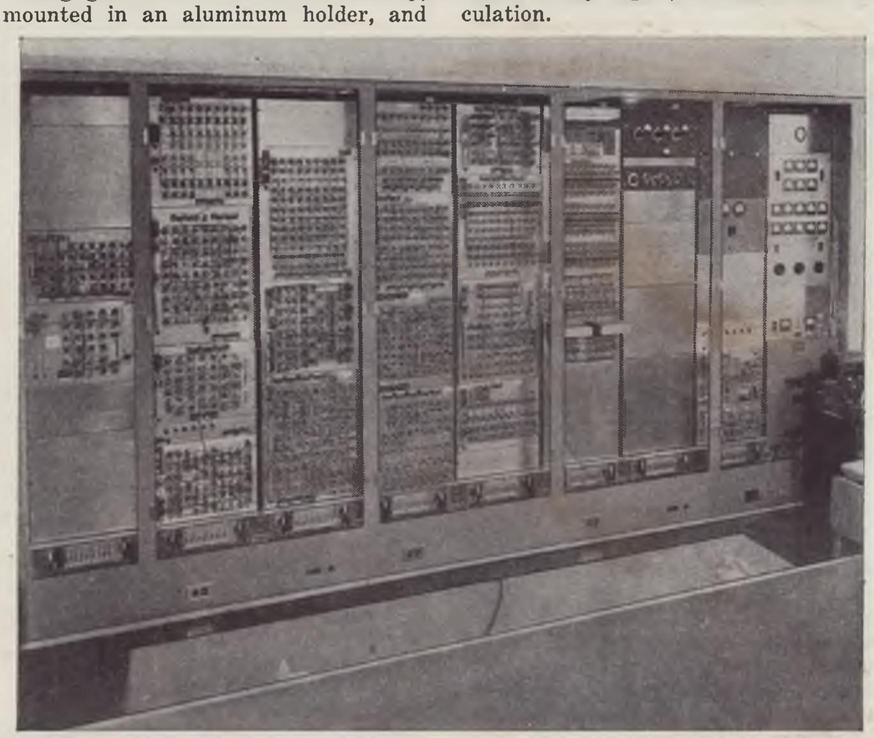


Fig. 1—Front view of SEAC shows nine racks holding all but the memory unit.

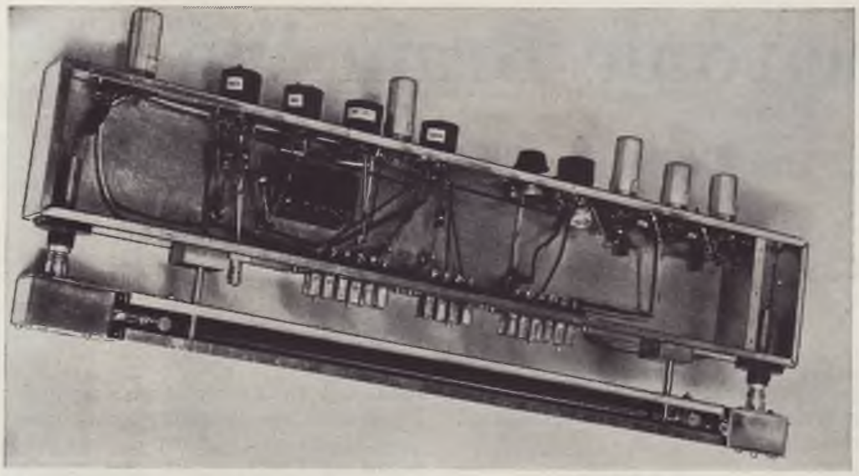


Fig. 2—One of 64 mercury delay lines in the serial memory section of SEAC.

How SEAC is organized

The over-all block diagram of SEAC is shown in Fig. 5, as drawn by the Bureau of Standards. The switch symbols indicate electronic operations performed automatically under the direction of the control unit of the machine. The time for switching is about 1 microsecond.

The usual five parts of the block diagram of an automatic computer are all here in the six blocks drawn. Input and output are shown at the top of the diagram in Fig. 5. In the machine, input consists of: (1) manual keyboard; (2) 5-hole paper-tape reader; (3) magnetic-wire reader, reading one channel of magnetized spots on wire. Output consists of: (1) a teletype printer; (2) paper-tape punch; (3) magnetic wire recording.

The next lower block in the block diagram is the "parallel memory unit." This unit was not in the machine when it started working in 1950, but at present writing is under test, about to be completed. It consists of 45 "electrostatic storage tubes" (see Article VIII of this series), each able to store 512 spots. This added faster memory will speed up the machine considerably.

The next two blocks are the arithmetical (or computing) unit, and the control unit of the machine. The last block is the second part of the memory, "the serial memory unit," consisting of the cabinet of the mercury delay lines.

The flexibility of the machine is indicated in the diagram by the solid-line arrow going into the control unit. This means that information from its various parts can affect the control of the machine.

The basic pulse rate of the machine is a million cycles per second. It contains 800 vacuum tubes, 500 pulse transformers, 11,000 germanium diodes, and 100,000 soldered connections. When the parallel memory is added, another 300 vacuum tubes and 4,500 germanium diodes will be added to the figures above.

All the information-manipulating as such in this machine is done by circuits using germanium diodes—rectifier circuits such as those described in Article XII. So far as information-manipulating goes, vacuum tubes are used only to change positive pulses to negative ones or vice versa.

How information is handled

The regular piece of information in SEAC consists of 45 binary digits, ones or zeroes, and is called a word. A word may be a number of 44 binary digits, together with one more binary digits designating the sign (plus or minus). Or the word may be an instruction, or group of instructions. A 44-binary-digit number is equivalent to an ordinary number of about 13 decimal digits.

Two instruction systems may be used with SEAC. One is called a "four-address system." The machine is told the *address* (or register number, or memory location) of the first *operand* (number to be operated with); the address of the second operand; the address where the result of the operation is to go; and the address where the next order number is to be found. The other is a three-address system; and the address where the next order is to be found is normally the next consecutively numbered memory location (see the discussion of programming in Article XII of this series). Each instruction also includes four binary digits used to specify the operation that the arithmetical unit is to perform on the cycle when it is obeying such instruction (the letters of operations shown in Chart 1 are translated into machine language).

The operations, and the time in thousandths of a second they require, are shown in the chart. For example, take multiplication. Most of the time, if we multiply two 13-decimal digit numbers together, like 1.789789922981 and 2.566566783422, we want the answer to only 13 decimal digits. We shall be content to throw away the right-hand part, and we will use order R. Only when we are interested in 26 decimal digits of accuracy will we want the right-hand part. Only in that event will we use the M and N orders.

The time shown for addition is 0.9 milliseconds and for multiplication is 3.0 milliseconds. Hence SEAC will on the average perform about 1,100 additons per second or 330 multiplications per second.

SEAC handles numbers only in pure binary form. Hence if any decimal number is to be used, it must be converted into pure binary. But SEAC is clever. It will take in the decimal number, convert it to pure binary, calculate with it, and when the results come along, will turn them back into decimal number form.

This conversion and all other mathematical work is accomplished by programming the machine. Once a program or routine for a type of computation

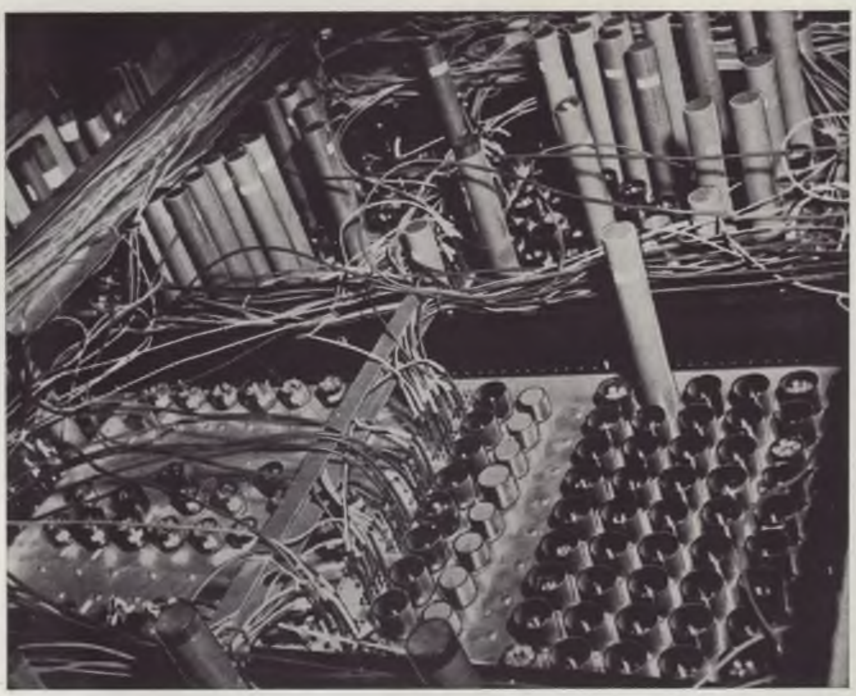


Fig. 3—Fiber tubes are the rapid memory; diode clusters are electronic switches.

has been worked out and translated into machine language, it can be stored on paper tape or magnetic wire, and given back to the machine whenever needed.

Even this labor can be reduced. For example, the Harvard Computation Laboratory has worked out a "coding machine" which will enable one to punch on a keyboard ordinary mathematical symbols, and let the machine prepare the detailed instructions for the automatic computer in machine language.

Problems worked on

What are the problems that SEAC has worked on? Up to the end of 1950, SEAC has put in 525 hours on problems for the Office of the Air Comptroller, solving large systems of linear algebraic equations in connection with planning of programs for the Air Force. It spent 72 hours in the study of the starting transient of a class C oscillator. It spent 68 hours determining sample sizes corresponding to the minimum variance in a census, using sampling methods. It spent 48 hours calculating the solution of a 27th-order system of ordinary differential equations relating to the neutron capture theory of the formation of the chemical elements in the universe. This problem was posed by the Applied Physics Laboratory of Johns Hopkins University.

Most of the problems are of course quite beyond the intelligent understanding of everyone but those few who have made a special study of the field in which they occur. We, the authors of this article, confess that we have to recite the above problems like parrots, repeating them from literature put out by the Bureau of Standards, with only a dim notion of what they mean!

But one problem we can understand is the following: SEAC calculated that the number 9,999,999,977 is a prime number, that is, has no factor except itself and 1. It did so by actually trying 80,000 trial divisors (the right trial divisors which would prove it) in 80,000 long divisions and finding that there was a reminder every time. It took SEAC 30 minutes to do that. A man with a desk calculator, working eight hours a day, would take about two months to do the same problem!

Operating experience

The operating experience on SEAC has been reported for October, November, and December, 1950. In those three months, the Bureau of Standards planned to operate the machine 24 hours a day, 7 days a week. Of each week's 168 hours, "preventive maintenance" was scheduled for 16 hours. Of the remaining time, half was devoted to the solution of problems, and the balance to computing machinery development and testing.

Now, in those three months, with regard to the part of the time allocated to problem solution, the average of "good" time was 76%. In fact, in the last week, the average was 96%. By "good" time is meant time when either problem solutions or coding checks were produced correctly, or when the machine was in good operating condition but idle—as may happen when the machine is being changed from one problem to another.

Thus we can see that if a machine doing 1,000 operations a second is operating at 75% efficiency, it is still worth a good deal more than a machine that does 2 or 3 operations a second with 100% efficiency.

Achievements such as SEAC's are the pay-off, the end result of the long, fascinating road of computer design and construction and testing and operation, so that more of the mental drudgery in the world can be lifted off the minds of human beings. And we hope that many of the readers of RADIO-ELECTRONICS will take a good look into the field of electronic handling of information, and attack some of the big problems of today, such as lower cost, more and cheaper memory, cleverer ways of programming machines, and the other great and interesting unsolved problems in this new field.

—end—

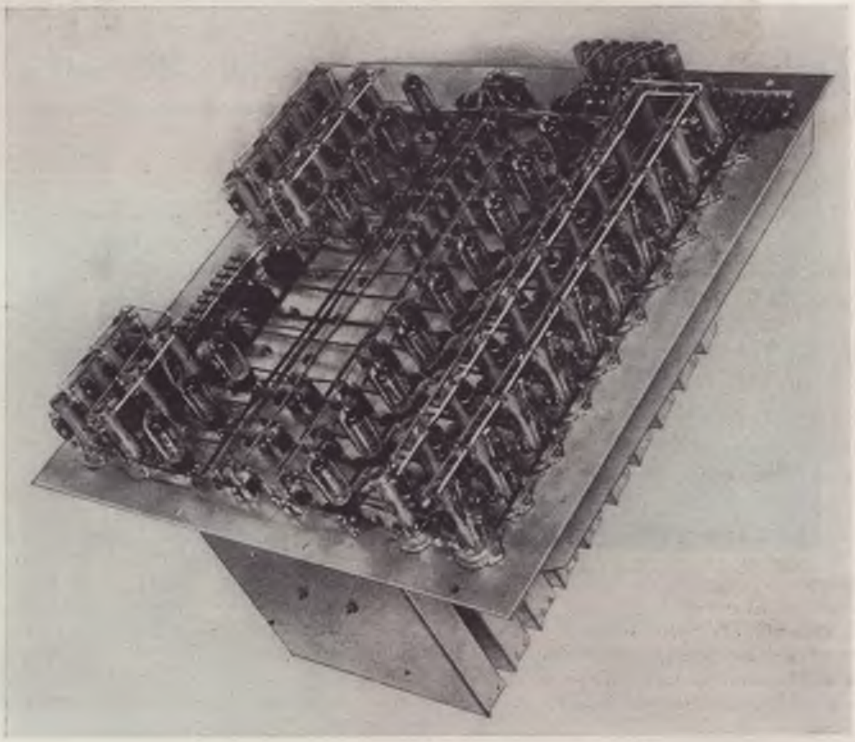


Fig. 4—Program registers were described previously. This is the SEAC unit.

CHART I

Operation	Abbreviation	Time*
1. Addition	(A)	0.9
2. Subtraction	(S)	0.9
3. Multiplication		
a. Major or left-hand part, unrounded	(M)	3.0
b. Major part, rounded	(R)	3.0
c. Minor or right-hand part	(N)	3.0
4. Division	(D)	3.0
5. Comparison (This is a conditional transfer of the control of the machine based on the value of the arithmetical result in the arithmetic unit)		
a. Value taking into account the sign plus or minus	(C)	0.7
b. Value disregarding the sign plus or minus	(K)	0.7
6. Logical Transfer (This is an arbitrary partial word transfer, for the purpose of forming composite words)	(L)	
7. Input-Out Control		
a. Read-In	(T)	50.0
b. Print-Out	(P)	50.0
c. Reverse Motion (This time is based on using input-output with a single channel magnetic wire, and handling words in blocks of 8 words)	(7)	50.0

*In milliseconds for complete operation including average access time.

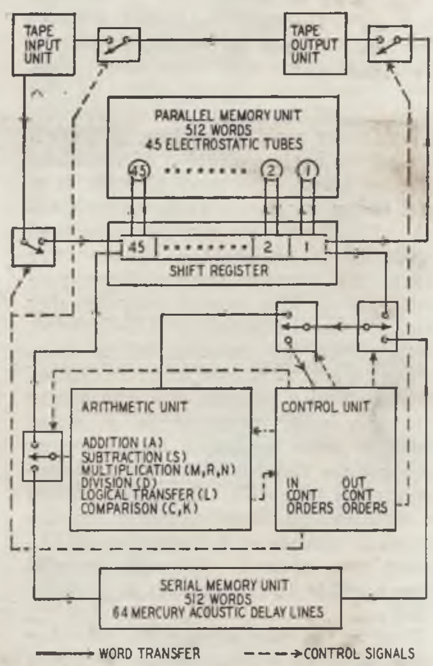


Fig. 5—How signals travel in SEAC.