Chapter 3

A MACHINE THAT WILL THINK:
THE DESIGN OF A VERY SIMPLE MECHANICAL BRAIN

We shall now consider how we can design a very simple machine that will think. Let us call it Simon, because of its predecessor, Simple Simon.

SIMON, THE VERY SIMPLE MECHANICAL BRAIN

By designing Simon, we shall see how we can put together physical equipment for handling information in such a way as to get a very simple mechanical brain. At every point in the design of Simon, we shall make the simplest possible choice that will still give us a machine that: handles information, transfers information automatically from one part of the machine to another, and has control over the sequence of operations. Simon is so simple and so small, in fact, that it could be built to fill up less space than a grocery-store box, about 4 cubic feet. If we know a little about electrical work, we will find it rather easy to make Simon.

What do we do first to design the very simple mechanical brain, Simon?

SIMON'S FLESH AND NERVES—REPRESENTING INFORMATION

The first thing we have to decide about Simon is how information will be represented: as we put it into Simon, as it is moved around inside of Simon, and as it comes out of Simon.

We need to decide what physical equipment we shall use to make Simon's flesh and nerves. Since we are taking the simplest convenient solution to each problem, let us decide to use: *punched paper tape* for putting information in, *relays* (see Chapter 2) and wires for storing and transferring information, and *lights* for putting information out.

For the equipment inside Simon, we could choose either electronic tubes or relays. We choose relays, although they are slower, because it is easier to explain circuits using relays. We

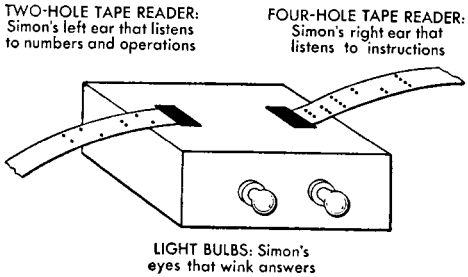

FIG. 1. Simon, the very simple mechanical brain.

can look at a relay circuit laid out on paper and tell how it works, just by seeing whether or not current will flow. Examples will be given below. When we look at a circuit using electronic tubes laid out on paper, we still need to know a good deal in order to calculate just how it will work.

How will Simon perceive a number or other information by means of punched tape, or relays, or lights? With punched paper tape having, for example, 2 spaces where holes may be, Simon can be told 4 numbers—00, 01, 10, 11. Here the binary digit 1 means a hole punched; the binary digit 0 means no hole punched. With 2 relays together in a register, Simon can remember any one of the 4 numbers 00, 01, 10, and 11. Here the binary digit 1 means the relay picked up or energized or closed; 0 means the relay not picked up or not energized or open. With 2 lights, Simon can give as an answer any one of the 4 numbers 00, 01, 10, 11. In this case the binary digit 1 means the light glowing; 0 means the light off. (See Fig. 1.)

We can say that the two lights by which Simon puts out the answer are his *eyes* and say that he tells his answer by *winking.* We can say also that the two mechanisms for reading punched paper tape are Simon's *ears.* One tape, called the *input tape,* takes in numbers or operations. The other tape takes in instructions and is called the *program tape.*

SIMON'S MENTALITY—POSSIBLE RANGE OF INFORMATION

We can say that Simon has a *mentality* of 4. We mean not age 4 but just the simple fact that Simon knows only 4 numbers and can do only 4 operations with them. But Simon can keep on doing these operations in all sorts of routines as long as Simon has instructions. We decide that Simon will know just 4 numbers, 0, 1, 2, 3, in order to keep our model mechanical brain very simple. Then, for any register, we need only 2 relays; for any answer, we need only 2 lights.

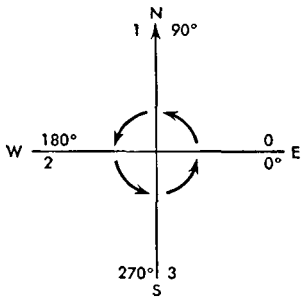

FIG. 2. Four directions.

Any calculating machine has a *mentality,* consisting of the whole collection of different ideas that the machine can ever actually express in one way or another. For example, a 10-place desk calculating machine can handle numbers up to 10 decimal digits without additional capacity. It cannot handle bigger numbers.

What are the 4 *operations with numbers* which Simon can carry out? Let us consider some simple operations that we can perform with just 4 numbers. Suppose that they stood for 4 directions in the order east, north, west, south (see Fig. 2). Or suppose that they stood for a turn counterclockwise through some right angles as follows:

0: Turn through 0°, or no right angles.
1: Turn through 90°, or 1 right angle.
2: Turn through 180°, or 2 right angles.
3: Turn through 270°, or 3 right angles.

Then we could have the operations of *addition* and *negation,* defined as follows:

ADDITION $c = a + b$

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

NEGATION $c = -a$

a	c
0	0
1	3
2	2
3	1

For example, the first table says, "1 plus 3 equals 0." This means that, if we turn 1 right angle and then turn in the same direction 3 more right angles, we face in exactly the same way as we did at the start. This statement is clearly true. For another example, the second table says, "2 is the negative of 2." This means that, if we turn to the left 2 right angles, we face in exactly the same way as if we turn to the right 2 right angles, and this statement also is, of course, true.

With only these two operations in Simon, we should probably find him a little too dull to tell us much. Let us, therefore, put into Simon two more operations. Let us choose two operations involving both numbers and logic: in particular, (1) finding which of two numbers is greater and (2) selecting. In this way we shall make Simon a little cleverer.

It is easy to teach Simon how to find which of two numbers is the greater when all the numbers that Simon has to know are 0, 1, 2, 3. We put all possible cases of two numbers a and b into a table:

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

Then we tell Simon that we shall mark with 1 the cases where a is greater than b and mark with 0 the cases where a is not greater than b:

GREATER THAN

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

For example, "2 is greater than 3" is false, so we put 0 in the table on the 2 line in the 3 column. We see that, for the 16 possible cases, a is greater than b in 6 cases and a is not greater than b in 10 cases.

There is a neat way of saying what we have just said, using the language of *mathematical logic* (see Chapter 9 and Supplement 2). Suppose that we consider the statement "a is greater than b" where a and b may be any of the numbers 0, 1, 2, 3. We can say that the *truth value* p of a *statement* P is 1 if the statement is true and that it is 0 if the statement is false:

$$p = 1 \text{ if } P \text{ is true, } 0 \text{ if } P \text{ is false}$$

The truth value of a statement P is conveniently denoted as $T(P)$ (see Supplement 2):

$$p = T(P)$$

Now we can say that the table for the operation *greater than* shows the truth value of the statement "a is greater than b":

$$p = T(a > b)$$

Let us turn now to the operation *selection*. By *selecting* we mean choosing one number a if some statement P is true and choosing another number b if that statement is not true. As before, let p be the truth value of that statement P, and let it be equal to 1 if P is true and to 0 if P is false. Then the operation of selection is fully expressed in the following table and logical formula (see Supplement 2):

SELECTION $\quad c = a \cdot p + b \cdot (1 - p)$

	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

For example, suppose that a is 2 and b is 3 and the statement P is the statement "2 is greater than 0." Since this statement is true, p is 1, and

$$a \cdot p + b \cdot (1 - p) = 2(1) + 3(0) = 2$$

This result is the same as selecting 2 if 2 is greater than 0 and selecting 3 if 2 is not greater than 0.

Thus we have four operations for Simon that do not overstrain his mentality; that is, they do not require him to go to any numbers other than 0, 1, 2, and 3. These four operations are: addition, negation, greater than, selection. We label these operations also with the numbers 00 to 11 as follows: addition, 00; negation, 01; greater than, 10; selection, 11.

SIMON'S MEMORY—STORING INFORMATION

The *memory* of a mechanical brain consists of physical equipment in which information can be stored. Usually, each section

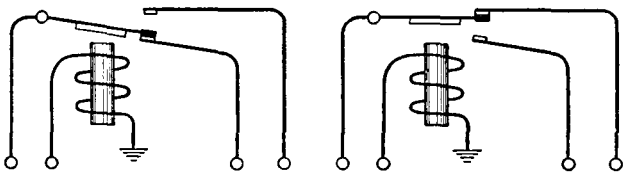

FIG. 3. Register *S1* storing 10.

of the physical equipment which can store one piece of information is called a *register*. Each register in Simon will consist of 2 relays. Each register will hold any of 00, 01, 10, 11. The information stored in a register 00, 01, 10, 11 may express a number or may express an operation.

How many registers will we need to put into Simon to store information? We shall need one register to read the input tape and to store the number or operation recorded on it. We shall call this register the *input register I*. We shall need another register to store the number or operation that Simon says is the answer and to give it to the output lights. We shall call this register the *output register O*. We shall need 5 registers for the part of Simon which does the computing, which we shall call the *computer:* we shall need 3 to store numbers put into the computer (*C1*, *C2*, *C3*), 1 to store the operation governing the computer (*C4*), and 1 to store the result (*C5*). Suppose that we

decide to have 8 registers for storing information, so as to provide some flexibility for doing problems. We shall call these registers *storage registers* and name them *S*1, *S*2, *S*3, ··· *S*8. Then Simon will have 15 registers: a memory that at one time can hold 15 pieces of information.

How will one of these registers hold information? For example, how will register *S*1 hold the number 2 (see Fig. 3)? The number 2 in machine language is 10. Register *S*1 consists of two relays, *S*1-2 and *S*1-1. 10 stored in register *S*1 means that relay *S*1-2 will be energized and that relay *S*1-1 will not be energized.

THE CONTROL OF SIMON

So far we have said nothing about the control of Simon. Is he docile? Is he stubborn? We know what his capacity is, but we do not know how to tell him to do anything. How do we connect our desires to his behavior? How do we tell him a problem? How do we get him to solve it and tell us the answer? How do we arrange control over the sequence of his operations? For example, how do we get Simon to add 1 and 2 and tell us the answer 3?

On the outside of Simon, we have said, there are two ears: little mechanisms for reading punched paper tape. Also there are two eyes that can wink: light bulbs that by shining or not shining can put out information (see Fig. 1). One of the ears—let us call it the *left ear*—takes in information about a particular problem: numbers and operations. Here the *problem tape* or *input tape* is listened to. Each line on the input tape contains space for 2 punched holes. So, the information on the input tape may be 00, 01, 10, or 11—either a number or an operation. The other ear—let us call it the *right ear*—takes in information about the sequence of operations, the program or routine to be followed. Here the *program tape* or *routine tape* or *control tape* is listened to. Each line on the program tape contains space for 4 punched holes. We tell Simon by *instructions* on the program tape what he is to do with the information that we give him on the input tape. The information on the program tape, therefore, may be 0000, 0001, 0010, ···, 1111, or

any number from 0 to 15 expressed in binary notation (see Supplement 2).

How is this accomplished? In the first place, Simon is a machine, and he behaves during time. He does different things from time to time. His behavior is organized in *cycles*. He repeats a cycle of behavior every second or so. In each cycle of Simon, he listens to or reads the input tape once and he listens to or reads the program tape twice. Every complete instruction that goes on the program tape tells Simon a register from which information is to be sent and a register in which information is to be received. The first time that he reads the program tape he gets the name of the register that is to receive certain information, the *receiving register*. The second time he reads the program tape he gets the name of the register from which information is to be sent, the *sending register*. He finishes each cycle of behavior by transferring information from the sending register to the receiving register.

For example, suppose that we want to get an answer out of Simon's computer into Simon's output lights. We put down the instruction

Send information from *C*5 into *O*

or, more briefly,

$$C5 \rightarrow O$$

But he does not understand this language. We must translate into machine language, in this case punched holes in the program tape. Naturally, the punched holes in the program tape must be able to specify any sending register and any receiving register. There are 15 registers, and so we give them punched hole *codes* as follows:

Register	Code	Register	Code
I	0001	*C*1	1010
*S*1	0010	*C*2	1011
*S*2	0011	*C*3	1100
*S*3	0100	*C*4	1101
*S*4	0101	*C*5	1110
*S*5	0110	*O*	1111
*S*6	0111		
*S*7	1000		
*S*8	1001		

To translate the direction of transfer of information, which we showed as an arrow, we put on the program tape the code for the receiving register first—in this case, output, O, 1111—and the code for the sending register second—in this case, $C5$, 1110. The instruction becomes 1111, 1110. The first time in any cycle that Simon listens with his right ear, he knows that what he hears is the name of the receiving register; and the second time that he listens, he knows that what he hears is the name of the sending register. One reason for this sequence is that any person or machine has to be prepared beforehand to absorb or take in any information.

Now how do we tell Simon to add 1 and 2? On the input tape, we put:

Add	00
1	01
2	10

On the program tape, we need to put:

$$I \rightarrow C4$$

$$I \rightarrow C1$$

$$I \rightarrow C2$$

$$C5 \rightarrow O$$

which becomes:

1101, 0001;
1010, 0001;
1011, 0001;
1111, 1110

THE USEFULNESS OF SIMON

Thus we can see that Simon can do such a problem as:

Add 0 and 3. Add 2 and the negative of 1. Find which result is greater. Select 3 if this result equals 2; otherwise select 2.

To work out the coding for this and like problems would be a good exercise. Simon, in fact, is a rather clever little mechanical brain, even if he has only a mentality of 4.

It may seem that a simple model of a mechanical brain like Simon is of no great practical use. On the contrary, Simon has the same use in instruction as a set of simple chemical experiments has: to stimulate thinking and understanding and to produce training and skill. A training course on mechanical brains could very well include the construction of a simple model mechanical brain as an exercise. In this book, the properties of Simon may be a good introduction to the various types of more complicated mechanical brains described in later chapters.

The rest of this chapter is devoted to such questions as:

How do transfers of information actually take place in Simon?

How does the computer in Simon work so that calculation actually occurs?

How could Simon actually be constructed?

What follows should be skipped unless you are interested in these questions and the burdensome details needed for answering them.

SIMON'S THINKING—TRANSFERRING INFORMATION

The first basic thinking operation for any mechanical brain is transferring information automatically. Let us see how this is done in Simon.

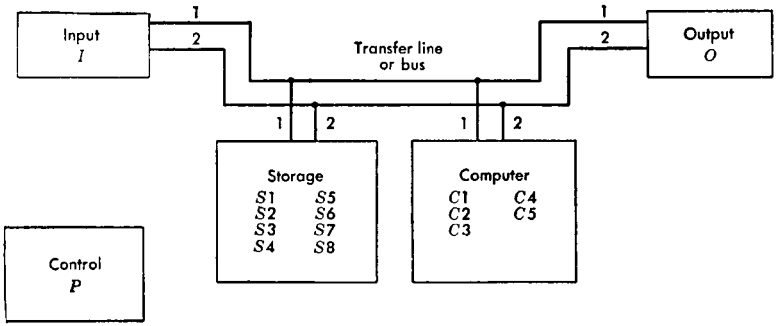

FIG. 4. Scheme of Simon.

Let us first take a look at the scheme of Simon as a mechanical brain (see Fig. 4). We have 1 input, 8 storage, 5 computer,

and 1 output registers, which are connected by means of transfer wires or a transfer line along which numbers or operations can travel as electrical impulses. This transfer line is often called the *bus,* perhaps because it is always busy carrying something. In Simon the bus will consist of 2 wires, one for carrying the right-hand digit and one for carrying the left-hand digit of any number 00, 01, 10, 11. Simon also has a number of neat little devices that will do the following:

When any number goes into a register, the coils of the relays of the register will be connected with the bus.
When any number goes out of a register, the contacts of the relays of the register will be connected with the bus.

For example, suppose that in register *C*5 the number 2 is stored. In machine language this is 10. That means the left-hand relay (*C*5-2) is energized and the right-hand relay (*C*5-1) is not energized. Suppose that we want to transfer this number 2 into the output register *O,* which has been cleared. What do we do?

Let us take a look at a circuit that will transfer the number (see Fig. 5). First we see two relays in this circuit. They belong to the *C*5 register. The *C*5-2 relay is energized since it holds 1; current is flowing through its coil, the iron core becomes a magnet, and the contact above it is pulled down. The *C*5-1 relay is not energized since it holds 0; its contact is not pulled down. The next thing we see is two *rectifiers.* The sign for these is a triangle. These are some modern electrical equipment that allow electrical current to flow in only one direction. In the diagram, the direction is shown by the pointing of the triangle along the wire. Rectifiers are needed to prevent undesired circuits. Next, we see the bus, consisting of two wires. One carries the impulses for left-hand or 2 relays, and the other carries impulses for the right-hand or 1 relays. Next, we see two relays, called the *entrance relays* for the *O* register. Current from Source 1 may flow to these relays, energize them, and close their contacts. When the first line of the program tape is read, specifying the receiving register, the code 1111 causes Source 1 to be energized. This fact is shown schematically by the arrow running from the program tape code 1111 to Source 1. Finally,

we see the coils of the two relays for the Output or O register. We thus see that we have a circuit from the contacts of the $C5$ register through the bus to the coils of the O register.

We are now ready to transfer information when the second line of the program tape is read. This line holds 1110 and designates $C5$ as the sending register and causes Source 2 to be

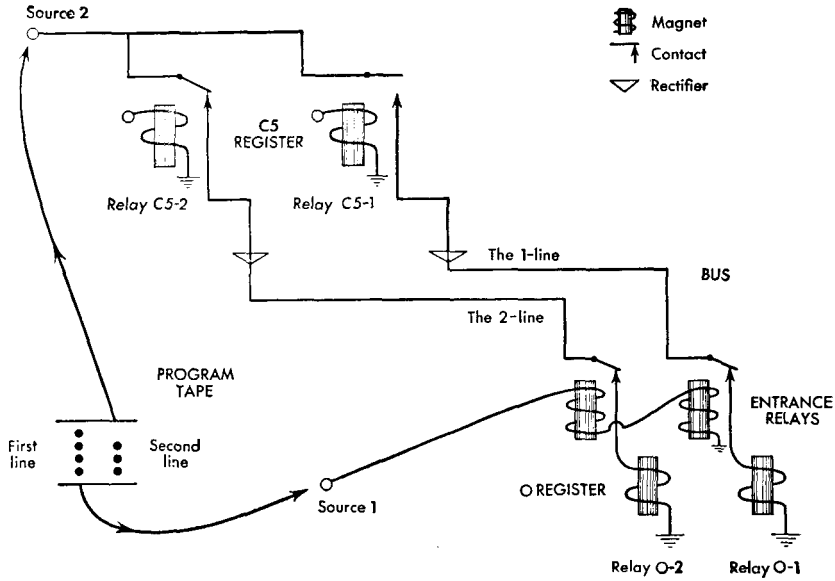

FIG. 5. Transfer circuit.

energized. This fact is shown schematically by the arrow running from the second line of the program tape to Source 2. When the second line is read, current flows:

1. From Source 2.
2. Through the contacts of the $C5$ register if closed.
3. Through the rectifiers.
4. Through the bus.
5. Through the entrance relay contacts of the O register.
6. Through the coils of the O register relays, energizing such of them as match with the $C5$ closed contacts; and finally
7. Into the ground.

Thus relay O-2 is energized; it receives current because contact $C5$-2 is closed. And relay O-1 is not energized; it receives no current since contact $C5$-1 is open. So we have actually transferred information from the $C5$ register to the O register.

The same process in principle applies to all transfers:

> The pattern of electrical impulses, formed by the positioning of one register, is produced in the positioning of another register.

SIMON'S COMPUTING AND REASONING

Now so far the computing registers in Simon are a mystery. We have said that $C1$, $C2$, and $C3$ take in numbers 00, 01, 10, 11, that $C4$ takes in an operation 00, 01, 10, 11, and that $C5$ holds the result. What process does Simon use so that he has the correct result in register $C5$?

Let us take the simplest computing operation first and see what sort of a circuit using relays will give us the result. The simplest computing operation is *negation.* In negation, a number 00, 01, 10, 11 goes into the $C1$ register, and the operation 01 meaning negation goes into the $C4$ register, and the correct result must be in the $C5$ register. So, first, we note the fact that the $C4$-2 relay must not be energized, since it contains 0, and that the $C4$-1 relay must be energized, since it contains 1.

Now the table for negation, with $c = -a$, is:

a	c
0	0
1	3
2	2
3	1

Negation in machine language will be:

a	c
00	00
01	11
10	10
11	01

Now if a is in the $C1$ register and if c is in the $C5$ register, then negation will be:

$C1$	$C5$
00	00
01	11
10	10
11	01

But each of these registers $C1$, $C5$ will be made up of two relays, the left-hand or 2 relay and the right-hand or 1 relay. So, in terms of these relays, negation will be:

$C1$-2	$C1$-1	$C5$-2	$C5$-1
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

Now, on examining the table, we see that the $C5$-1 relay is energized if and only if the $C1$-1 relay is energized. So, in order to energize the $C5$-1 relay, all we have to do is transfer the information from $C1$-1 to $C5$-1. This we can do by the circuit shown in Fig. 6. (In this and later diagrams, we have taken one more step in streamlining the drawing of relay contacts: the contacts are drawn, but the coils that energize them are represented only by their names.)

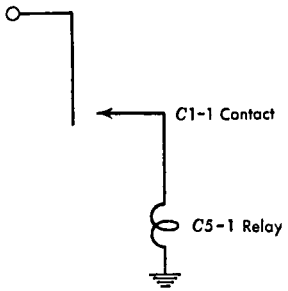

FIG. 6. Negation—right-hand digit.

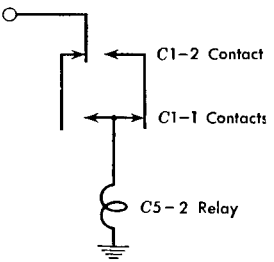

FIG. 7. Negation—left-hand digit.

Taking another look at the table, we see also that the $C5$-2 relay must be energized if and only if:

*C*1-2 HOLDS:	AND	*C*1-1 HOLDS:
0		1
1		0

A circuit that will do this is the one shown in Fig. 7. In Fig. 8 is a circuit that will do all the desired things together: give the right information to the *C*5 relay coils if and only if the *C*4 relays hold 01.

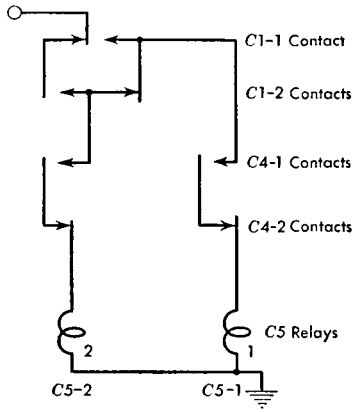

FIG. 8. Negation circuit.

Let us check this circuit. First, if there is any operation other than 01 stored in the *C*4 relays, then no current will be able to get through the *C*4 contacts shown and into the *C*5 relay coils, and the result is blank. Second, if we have the operation 01 stored in the *C*4 relays, then the *C*4-2 contacts will not be energized—a condition which passes current—and the *C*4-1 contacts will be energized—another condition which passes current—and:

IF THE NUMBER IN *C*1 IS:	THEN *C*1-1:	AND *C*1-2:	AND THE *C*5 RELAYS ENERGIZED ARE:
0	does not close	does not close	neither
1	closes	does not close	*C*5-2, *C*5-1
2	does not close	closes	*C*5-2 only
3	closes	closes	*C*5-1 only

Thus we have shown that this circuit is correct.

We see that this circuit uses more than one set of contacts for several relays (*C*1-2, *C*4-1, *C*4-2); relays are regularly made with 4, 6, or 12 sets of contacts arranged side by side, all con-

trolled by the same pickup coil. These are called 4-, 6-, or 12-*pole* relays.

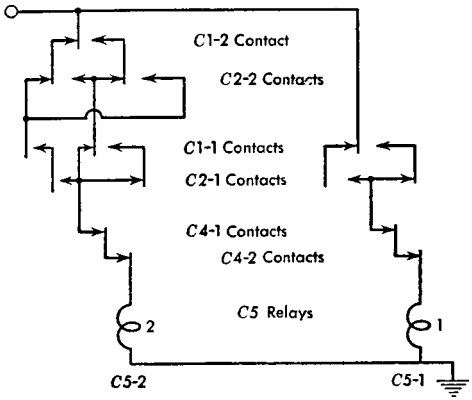

FIG. 9. Addition circuit.

Circuits for *addition*, *greater than*, and *selection* can also be determined rather easily (see Figs. 9, 10, 11). (*Note:* By means of the *algebra of logic*, referred to in Chapter 9 and Supple-

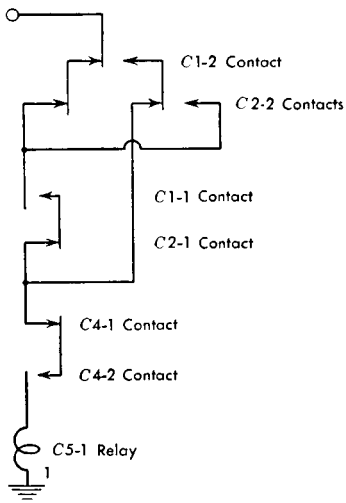

FIG. 10. Greater-than circuit.

ment 2, the conditions for many relay circuits, as well as the circuit itself, may be expressed algebraically, and the two expressions may be checked by a mathematical process.) For example,

let us check that the addition circuit in Fig. 9 will enable us to add 1 and 2 and obtain 3. We take a colored pencil and draw closed the contacts for $C1$-1 (since $C1$ holds 01) and for $C2$-2 (since $C2$ holds 10). Then, when we trace through the circuit,

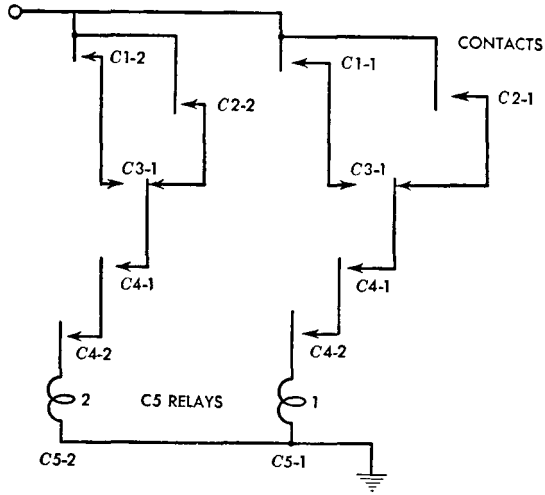

FIG. 11. Selection circuit.

remembering that addition is stored as 00 in the $C4$ relays, we find that both the $C5$ relays are energized. Hence $C5$ holds 11, which is 3. Thus Simon can add 1 and 2 and make 3!

PUTTING SIMON TOGETHER

In order to put Simon together and make him work, not very much is needed. On the outside of Simon we shall need two small mechanisms for reading punched paper tape. Inside Simon, there will be about 50 relays and perhaps 100 feet of wire for connecting them. In addition to the 15 registers (I, $S1$ to $S8$, $C1$ to $C5$, and O), we shall need a register of 4 relays, which we shall call the *program register*. This register will store the successive instructions read off the program tape. We can call the 4 relays of this register $P8$, $P4$, $P2$, $P1$. For example, if the $P8$ and $P2$ relays are energized, the register holds 1010, and this is the program instruction that calls for the 8th plus 2nd, or 10th, register, which is $C1$.

For connecting receiving registers to the bus, we shall need a relay with 2 poles, one for the 2-line and one for the 1-line, for each register that can receive a number from the bus. For example, for entering the output register, we actually need only one 2-pole relay instead of the two 1-pole relays drawn for simplicity in Fig. 5. There will be 13 2-pole relays for this purpose, since only 13 registers receive numbers from the bus; registers *I* and *C*5 do not receive numbers from the bus. We call these 13 relays the *entrance relays* or *E relays,* since *E* is the initial letter of the word entrance.

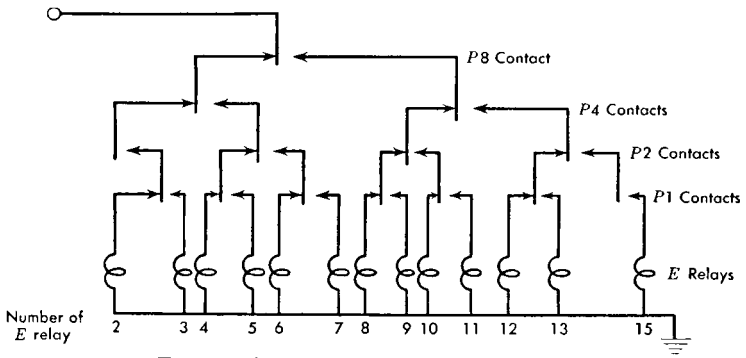

FIG. 12. Select-Receiving-Register circuit.

The circuit for selecting and energizing the *E* relays is shown in Fig. 12. We call this circuit the *Select-Receiving-Register* circuit. For example, suppose that the *P*8 and *P*2 relays are energized. Then this circuit energizes the *E*10 relay. The *E*10 relay closes the contacts between the *C*1 relay coils and the bus; and so it connects the *C*1 register to receive the next number that is sent into the bus. This kind of circuit expresses a classification and is sometimes called a *pyramid circuit* since it spreads out like a pyramid. A similar pyramid circuit is used to select the sending register.

We shall need a relay for moving the input tape a step at a time. We shall call this relay the *MI relay,* for moving input tape. We also need a relay for moving the program tape a step at a time. We shall call this relay the *MP relay* for moving program tape. Here then is approximately the total number of relays required:

Relays	Name	Number
I, S, C, O	Input, Storage, Computer, Output	30
P	Program	4
E	Entrance	13
MI	Move-Input-Tape	1
MP	Move-Program-Tape	1
	Total	49

A few more relays may be needed to provide more contacts or poles. For example, a single *P*1 relay will probably not have enough poles to meet all the need for its contacts.

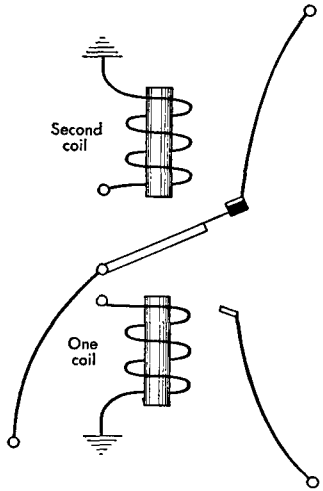

Fig. 13. Latch relay.

Each cycle of the machine will be divided into 5 equal *time intervals* or *times* 1 to 5. The timing of the machine will be about as follows:

Time	Action
1	Move program tape. Move input tape if read out of in last cycle.
2	Read program tape, determining the receiving register. Read through the computing circuit setting up the *C*5 register.
3	Move program tape. Energize the *E* relay belonging to the receiving register.
4	Read program tape again, determining the sending register.
5	Transfer information by reading through the Select-Sending-Register circuit and the Select-Receiving-Register circuit.

In order that information may remain in storage until wanted, register relays should hold their information until just before the next information is received. This can be accomplished by keeping current in their coils or in other ways. There is a type of relay called a *latch relay*, which is made with two coils and a latch. This type of relay has the property of staying or latching in either position until the opposite coil is impulsed (see Fig. 13). This type of relay would be especially good for the registers of Simon.

If any reader sets to work to construct Simon, and if questions arise, the author will be glad to try to answer them.