SCHAUM'S OUTLINE SERIES

THEORY AND PROBLEMS OF

Introduction to

COMPUTER SCIENCE

FRANCIS SCHEID

INCLUDING 300 SOLVED PROBLEMS

SCHAUM'S OUTLINE SERIES IN COMPUTERS

McGRAW-HILL BOOK COMPANY

SCHAUM'S OUTLINE OF

THEORY AND PROBLEMS

OF

Introduction to COMPUTER SCIENCE

•

BY

FRANCIS SCHEID

Professor of Mathematics
Boston University

SCHAUM'S OUTLINE SERIES

McGRAW-HILL BOOK COMPANY

New York, St. Louis, San Francisco, Düsseldorf, Johannesburg, Kuala Lumpur, London, Mexico, Montreal, New Delhi, Panama, Rio de Janeiro, Singapore, Sydney, and Toronto

07-055195-2

13 14 15 SH SH 8 7 6 5 4 3 2

Preface

Today even first graders know that computers can do fantastic things, that for certain purposes at least computers are truly superhuman, one computer being more than a match for a million human rivals. The fundamental question facing any beginner in computer science asks just how it is that electricity can be trained to such a spectacular level of performance. The answer can be compressed into two words, *hardware* and *software.* Computer hardware involves circuits, memory units, reading and writing devices. An elementary description of these things, requiring no background in electricity or engineering, is presented in Chapters 1 to 4. No effort is made to tell the whole story, only to suggest the underlying ideas. Computer software means programming, and this aspect of the subject is treated in Chapters 5 to 14. The book thus divides into two parts which are almost entirely independent of each other. This means that it may be used in several different ways.

1. *A short study of programming* may be based on Chapters 7 and 8, with further examples drawn from later chapters as desired.

2. *A longer study of programming* may include most of the material in Chapters 7 to 14. Not all of the longer programming examples need be taken in detail. A few in particular, such as the scheduling program of Chapter 11 and the checkers program of Chapter 14, may be omitted entirely without losing continuity. It would be a shame not to find time for the quite different Chapter 5.

3. *A general introduction to computer science* is available by undertaking the entire book. Here again there is the option of omitting some of the longer programming examples, with the further option of abbreviating the hardware presentation as well. This could be done, for example, by selecting just a few samples from Chapter 3 and taking Chapter 4 only up to Problem 4.22 or perhaps 4.36. The main purpose, of seeing how electricity can be trained, will still have been achieved. For those who want to deepen their study of the subject a bibliography is appended.

All the Fortran programs in this book were tested on the IBM 7090/7094 at the Ecole Polytechnique of the University of Lausanne, Switzerland, where it has been a genuine pleasure to spend my sabbatical leave. I am very grateful to Professor Blanc, director of the computing center, for this opportunity and have expressed a liquid merci with an offering of vin blanc Vaudois. Unless copying errors have been made, all programs should work. A few in later chapters, notably those in which random numbers are generated, include statements designed particularly to be used on the Lausanne machine. The changes needed to adapt such programs to other machines are minor and local experts will be happy to demonstrate their talents in making them. It is also possible that some of the programs can be improved and suggestions will be welcomed.

To better mutual understanding between man and machine,

FRANCIS SCHEID

CONTENTS

Chapter 1

Information Processing

Computer science involves the development and use of devices for processing information. Information in one form is presented to the device and information in another, presumably more useful, form is required from it. The former is called the input information; it is the raw material of the process. The latter is the output information, the finished product. (Fig. 1-1.)

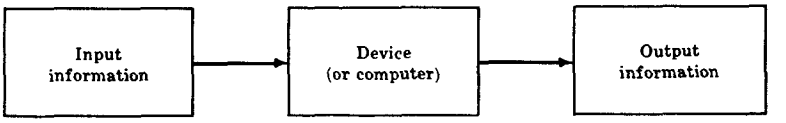

Fig. 1-1

Information is not an easy word to define with precision. It is related somehow to ideas and meaning, and can be communicated between intelligent sources as well as processed or converted into different forms. Part of the difficulty is that information may be presented in a variety of languages. Unless one understands the language the information may be meaningless or "uninformative". Several languages will be used as we proceed, some of which were designed specifically to be understood by electrical machines, or computers. Such machines do exhibit certain aspects of intelligence. At least, they are capable of receiving and processing information. The extent of machine intelligence, whether the processing they perform can be called "thinking", the potential capability of machines of the future and their relationship with the human population of this planet, such issues suggest the more exotic levels of our subject. Most of our time and energy will be devoted to the simpler aspects of computer science today, but at times it will be interesting to speculate about its future.

Example 1.1.

What is the product 12×43? This question serves as the input information for the present example. Elementary school students soon learn a routine for discovering the answer to the question and thus they serve as the device which processes this input information, perhaps as follows.

$$\begin{array}{r} 12 \\ 43 \\ \hline 36 \\ 48 \\ \hline 516 \end{array}$$

The output information is the number 516. There are of course many other ways of finding the same result, and this is typical of computer science. Here, for example, one could always arrange twelve 43's in a column and obtain the same 516 by addition. There is also the method of continually doubling one factor while halving the other, ignoring remainders until a 1 is produced.

12	43
6	86
3	172 ✓
1	344 ✓

✓	12	43
✓	24	21
	48	10
✓	96	5
	192	2
✓	384	1

Where remainders occur, the doubled factors (shown checked) are then added together. The sum will be 516. By any of these methods the human computer may process the input information (12×43) and generate the output information (516) as summarized in Fig. 1-2. It is worth noting that someone unfamiliar with decimal symbols for numbers or with decimal procedures of computation would find no meaning in any of the information involved in this example.

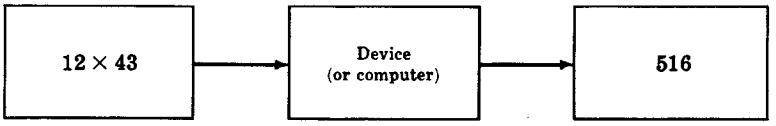

Fig. 1-2

Example 1.2.

Arrange the names JOHN, JOAN, JILL, JACK, FRED in alphabetical order. These five names, together with the instruction to alphabetize them, are a form of information, the input information of the present example. For anyone, or any device, familiar with the particular alphabet (or language) involved it is no severe challenge to process this information and achieve the output shown in Fig. 1-3.

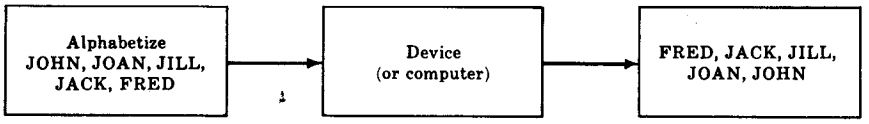

Fig. 1-3

Example 1.3.

Find the largest number in the set consisting of 43, 12, 47, 31, 44, 25, 46, 39. The eye will quickly pick out the number 47, a few visual comparisons being enough to identify it. In a set including several thousand numbers it is conceivable that the eye might be outperformed by some other device, but here it is clearly adequate. Here again a human computer familiar with decimal language and with the meaning of largest number can perform the required information processing. Fig. 1-4 summarizes the flow of information.

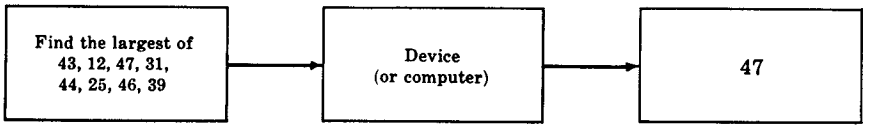

Fig. 1-4

A computer, for our purposes, will be an electrical device, or machine, which processes information. No knowledge of electricity will be needed except for a few very primitive ideas, such as that a wire may at one time be carrying a current and at another time may not. These two states of the wire will be called hot and cold, respectively. It is this basic fact, that two easily identifiable electrical states exist, which has been exploited to represent information in forms "understandable" by electrical machines. Our purpose in the first part of this book will be to examine some of the details of this exploitation, to see *how it is that human beings and electrical machines can both understand the same information processing efforts,* or to put it loosely, to watch the machines think. We will look only for the general view, the precise electrical details being left to the engineers.

The language for an electrical machine, or computer, will thus be based on hot and cold values, rather than on decimal symbols or alphabets. For example, if alternate hot and cold values are introduced to a wire by turning the current on and off, the wire's experiences may be recorded as

1 0 1 0 1 0 1

where for our own human understanding 1 means hot and 0 means cold. The wire itself will understand, so to speak, the actual current pulses and has no need for our symbols. The sequence

0 0 0 0 0 1 1

means similarly that five cold values are followed by two hots. In such sequences we have the beginning of a language that can be understood by both man and machine.

An OR circuit is a very simple but very useful electrical device. As shown in Fig. 1-5 it receives input information at two contacts and presents output information at one contact. Its function may be described by saying that the output is cold only when both inputs are cold. This is also shown by the four-value sequences appearing in the diagram. The input values are received in pairs and only the 0, 0 pair produces a 0 output. It is important to notice that these two input sequences have been carefully selected so that all possible input combinations are included; both contacts may be cold (0 and 0), both hot (1 and 1), or one hot and the other cold (1 and 0, or 0 and 1). There are no other possible input combinations. There is also here a first suggestion that timing is important. Each input pair must be received simultaneously, and the OR circuit must have time to produce the correct output before the next input pair appears. The OR circuit may be viewed as our first electrical device for information processing, our first computer. As already noted it is a very simple one.

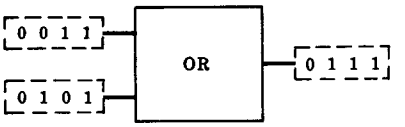

Fig. 1-5

Example 1.4.

An OR circuit processes the two seven-value sequences presented a moment ago as shown in Fig. 1-6. The output sequence is 1010111. Comparing with Fig. 1-1 one sees that the pattern is the same; information is being processed. Both we and the electrical device agree that simultaneous cold values occur in only two positions.

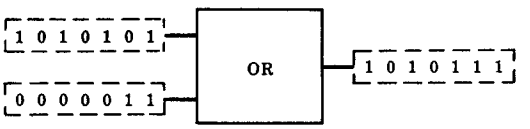

Fig. 1-6

Example 1.5.

An old-fashioned but easily understood OR circuit is shown in Fig. 1-7. A power source is trying to move current to the indicated output contact, through either of the two switches which are shown in the open position. Either switch may be closed by energizing the companion magnet (shown as a coil) and this is done by making the appropriate input contact hot. Only when both input contacts are cold will current fail to flow, since then both switches remain in the open position. More modern OR circuits use vacuum tubes, transistors or other devices.

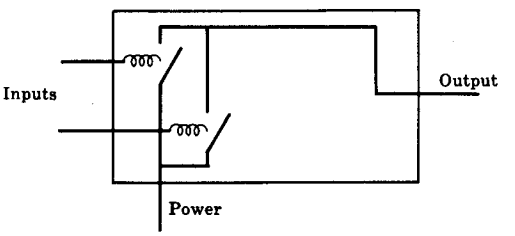

Fig. 1-7. An OR circuit.

An AND circuit is another simple but useful electrical device, or computer. Like the OR circuit it receives input information at two contacts and presents output information at one. Its function may be described by saying that the output will be hot only when both inputs are hot. This is also shown by the sequences appearing in Fig. 1-8, which once again include

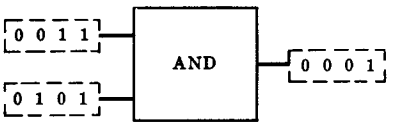

Fig. 1-8

the only possible input combinations. Only the 1, 1 combination produces a hot output. The AND circuit is our second electrical information processor.

Example 1.6.

An AND circuit processes the two sequences of Example 1.4 as shown in Fig. 1-9. The output information is the sequence 0000001. Both we and the machine arrive at this same result; simultaneous hot inputs occur only in the last position.

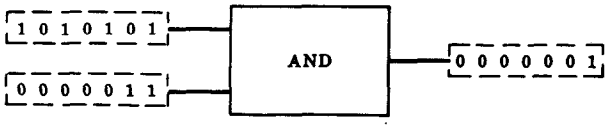

Fig. 1-9

Example 1.7.

An old-fashioned AND circuit is shown in Fig. 1-10. As with the OR circuit a power source is trying to move current to the output contact, but here it is clear that both switches must be closed if this is to occur. Closing these switches is achieved by making the input contacts hot, so only when both are hot simultaneously will current flow and produce a hot output.

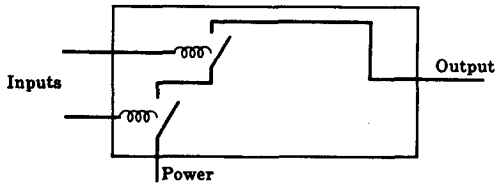

Fig. 1-10. An AND circuit.

The names OR and AND which are traditionally applied to the two circuits just introduced were originally suggested by the fact that for the first a hot output will occur if either one *or* the other (or both) of the input contacts is made hot, while for the second a hot output occurs only when both one input *and* the other are hot simultaneously. These circuits have important logical applications which will be mentioned in Chapter 3. They also prove to be fundamental in the design of more complex machines for handling arithmetical and other types of problems.

A NOT circuit has one input contact and one output. The two contacts always have opposite values. Fig. 1-11 expresses this same fact by showing that the sequence 01, which includes the only two input values possible, is processed into 10. The name NOT is obviously inspired by the fact that the output is hot precisely when the input is *not*, and vice versa. This circuit is heavily involved with OR and AND in the design of more complex devices.

Fig. 1-11

Fig. 1-12

Example 1.8.

A NOT circuit processes the alternating sequence of Example 6 as shown in Fig. 1-12. Though this is information processing at a rather primitive level, the point is that the electrical circuit does "understand" what it is doing. The language involved is the one language which such a device can handle. Moreover, once again we and the machine agree on the result; we both understand the nature of the information processing involved. Achieving higher and higher levels of understanding between man and machine is one of the larger tasks of computer science.

Example 1.9.

An old-fashioned NOT circuit is shown in Fig. 1-13. Here the switch will be assumed to be naturally closed, rather than naturally open as in Figs. 1-7 and 1-10. A hot input will energize the magnet (again represented as a coil) and pull the switch open. Since this stops the flow of current the output contact will be cold. The two contacts thus have opposite values at all times, exactly what is wanted. Crude models of OR, AND and NOT circuits have now been presented. Modern versions use transistors or other electrical components, but it is not our purpose to plunge deeply into these engineering affairs.

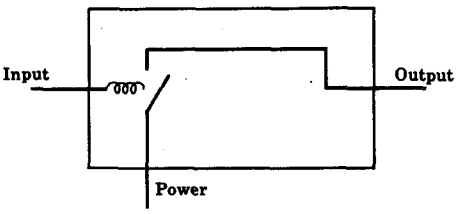

Fig. 1-13. A NOT circuit.

Solved Problems

1.1. Given the input information "Find the remainder when 144 is divided by 17", what is the correct output information?

Ordinary division soon leads to the remainder 8. Note that in addition to a knowledge of decimal symbols and the division process one must also understand the *words* used in the input information or he cannot proceed.

1.2. Given the input information "How many prime numbers are between 10 and 50", what is the correct output information?

Eleven. Here again both the number and word *languages* must be understood by the device (human being) which is to do the information processing.

1.3. The input information consists of "Arrange in increasing order the numbers represented by II, VII, XI, IV, VI". What is the correct output information?

After recognizing the Roman numerals one easily manages II, IV, VI, VII, XI.

1.4. The input information is "Changing one letter at a time convert CAT into TIN in three steps, each step producing an English word". What is the correct output information?

CAN, TAN, TIN is one suitable output. Here a memory of English words is needed to determine acceptable steps. For example, is TAT, TIT, TIN acceptable, or CIT, CIN, TIN? Problems like this one are not difficult for human beings but can be somewhat substantial hurdles for electrical machines.

1.5. The input information is "Rhyme the word BOWL from the set of words FOWL, SCOWL, HOWL, POLL". What is the correct output?

POLL. This is also easy for a human computer but could be severe for an electrical one.

1.6. What output information would be provided by an OR circuit given the following sequence pairs as inputs?

(*a*)	(*b*)	(*c*)
00001111	01010	0011
00111100	10101	0101

Recalling that a cold (0) output only comes from a pair of simultaneous cold inputs, we can answer as follows.

(*a*) 00111111 (*b*) 11111 (*c*) 0111

The computation is easy enough, but the important point is that we and the OR circuit are on common ground here. We do the same information processing.

1.7. What output information would be provided by an AND circuit given the sequence pairs of the preceding problem as inputs?

Recalling that a hot (1) output only comes from a pair of simultaneous hot inputs, we can answer as follows.

(*a*) 00001100 (*b*) 00000 (*c*) 0001

Again the computation is trivial, but again the important thing is that we have established rapport between man and machine. Though the level of rapport is meager at the moment, at least we have made a start.

1.8. What output information would be provided by a NOT circuit given the three outputs of the preceding problem as successive inputs?

Recalling the reversal of values, we can forecast the machine's answers.

(*a*) 11110011 (*b*) 11111 (*c*) 1110

1.9. Fig. 1-14 shows an electrical circuit (or machine, or computer) which combines an AND circuit with a NOT circuit. The original input sequences are called simply *A* and *B*. The output sequence of the AND circuit has been labeled *C*, and this sequence also serves as input to the NOT. The final output sequence has been called *D*. What will *C* and *D* become if this machine is offered the sequence pair 00001111 and 00111100 as *A* and *B*?

C will be 00001100 as in Problem 1.7 and *D* will be 11110011 as in Problem 1.8. This machine essentially combines the circuits of Problems 1.7 and 1.8, and is known as a NAND circuit.

1.10. What will the *C* and *D* of the preceding problem become if *A* is the sequence 01010 and *B* is 10101?

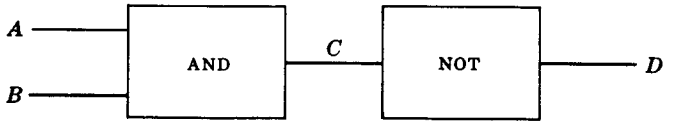

Fig. 1-14. NAND circuit.

C will be 00000 as in Problem 1.7 and *D* will be 11111 as in Problem 1.8.

1.11. What will be the output *D* of the NAND circuit in Fig. 1-14 if *A* is the sequence 0011 and *B* is 0101?

1110, as Problems 1.7 and 1.8 show. This pair of input sequences is particularly important since, as mentioned in connection with the OR and AND circuits of Figs. 1-5 and 1-8, it presents the only four input combinations possible. The operation of a NAND circuit may therefore be completely described by giving its output for this pair. (See Fig. 1-15.)

Fig. 1-15

1.12. The machine of Fig. 1-16 below is known as a NOR circuit. What will be the output sequence of this machine if the inputs *A* and *B* are 0011 and 0101? Notice once again that these four-value sequences offer the machine all four possible input combinations.

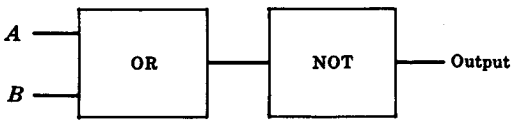

Fig. 1-16. NOR circuit.

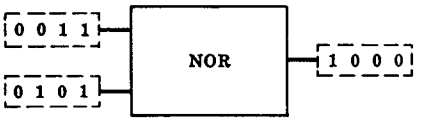

Fig. 1-17

The output of the OR circuit is 0111 as in Problem 1.6. The final output is therefore 1000 and this is exhibited in Fig. 1-17.

1.13. The machine of Fig. 1-18 combines two NOT and one OR circuits. What is its output sequence if A is 0011 and B is 0101?

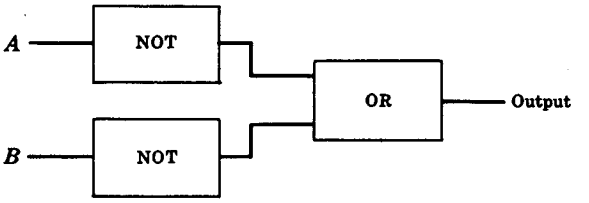

Fig. 1-18

The outputs of the two NOT circuits are 1100 and 1010. The final output is therefore 1110. But this is exhibited in the same output given by the NAND circuit of Fig. 1-14, as Problem 1.11 shows! And because the A and B being used include the only possible input combinations to either circuit, we have shown that the two machines will always produce the same output information if given the same inputs. Such machines will be called equivalent machines.

1.14. Show that the circuit of Fig. 1-19 will produce the same output information as a NOR circuit if given the same inputs A and B.

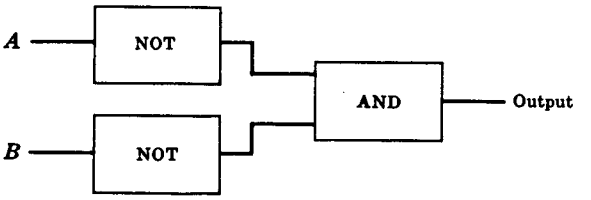

Fig. 1-19

It will be sufficient to compare their outputs when A is 0011 and B is 0101, since this includes all possible input combinations. In this case the two NOT circuits output 1100 and 1010, after which the AND manages 1000. This agrees with the output of a NOR circuit as summarized in Fig. 1-17.

1.15. Describe the output of the circuit in Fig. 1-20.

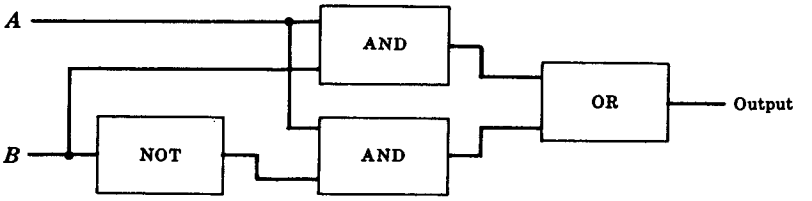

Fig. 1-20

Here a word of explanation may be helpful. In diagrams of this sort, wherever wires are joined with a heavy spot, contact is to be assumed, so that both wires carry the same 0 or 1 value. Where wires appear to touch but there is no heavy spot, no contact is assumed. This agreement is helpful in the representation of circuits. As usual we now choose 0011 for A and 0101 for B. Then the NOT circuit outputs 1010. The two ANDS then manage 0001 and 0010. Since these two sequences are the inputs to the OR circuit, the final output will be 0011. But this is identical with the sequence A itself! In other words, no information processing of any real significance has been achieved by this machine. Its output information will always be one of the original input sequences. Wasted labor is not uncommon but it makes sense to try to recognize and eliminate it where possible, and we shall try to avoid such machines as the one in Fig. 1-20.

1.16. Describe the output of the circuit in Fig. 1-21.

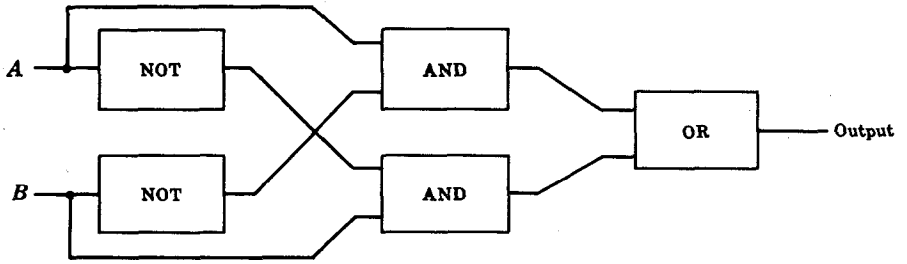

Fig. 1-21

Although this machine has a superficial resemblance to the useless circuit of the preceding problem it will prove to have important applications. Let A be 0011 and B be 0101 as usual, to cover all possible input combinations. Then the two NOT circuits produce 1100 and 1010. The ANDS then manage 0010 and 0100, after which the OR gives the final output of 0110. Notice that this output is hot (1) whenever the two inputs disagree and cold (0) when the inputs do agree. It therefore indicates with a hot output value the times when either one or the other, but *not both*, of the two input values is hot. It is sometimes called a comparator circuit, since it essentially compares the two input values and tells whether they agree or not. Fig. 1-22 summarizes the operation of such a comparator.

Fig. 1-22

Supplementary Problems

1.17. Given the input information "(10 + 3)(5 + 7)", what is the correct output information?

1.18. Given the input information "Find the largest number which divides evenly into 1365, not counting 1365 itself", what is the correct output information?

1.19. A journey from position A to position B in Fig. 1-23 will be six blocks long, if only eastward or northward travel is allowed (to minimize the length of the journey). There are several different paths by which this six-block journey can be completed. How many? Consider the foregoing as input information. What is the correct output information?

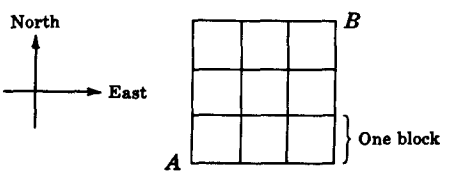

Fig. 1-23

1.20. Given the input information "Alphabetize the six possible arrangements of the letters CAT", what is the correct output information?

1.21. What output information would be provided by an AND circuit given the following pairs of sequences as inputs?

(*a*) 010101 (*b*) 01110 (*c*) 0011
110011 11011 1100

1.22. What output information would be provided by an OR circuit given the pairs of sequences of the preceding problem as inputs?

1.23. What would be the output information from a NOT circuit, given the output sequences of the preceding problem as inputs?

1.24. Fig. 1-24 shows four simple combinations of AND and NOT circuits. For one of them the response to the standard *A* and *B* inputs (0011 and 0101) is indicated, the output information being 0100. Fill in the remaining outputs in the boxes provided, showing that each machine provides a hot output value for only one of the four possible input combinations, but that between them all four cases are covered.

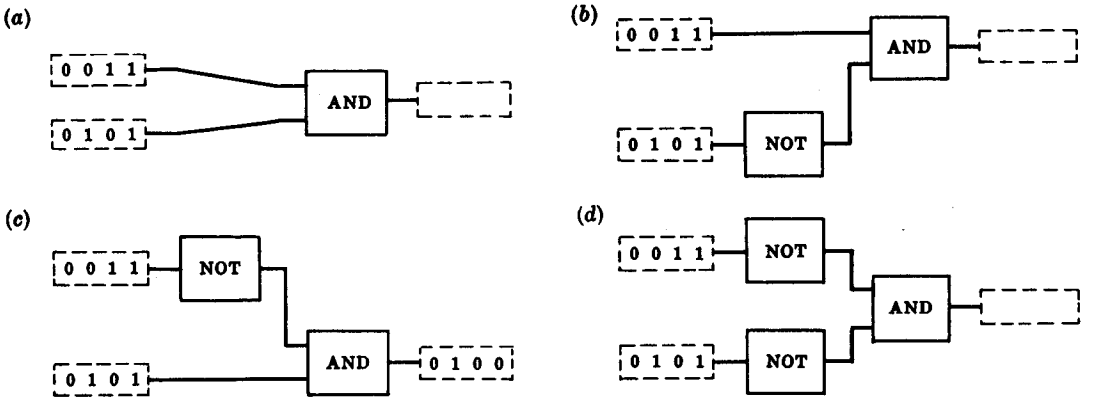

Fig. 1-24

1.25. The comparator of Fig. 1-21 uses two of the circuits of the preceding problem, channeling their output information into an OR circuit. Which two are used?

1.26. The circuit of Fig. 1-25 also uses two of the circuits of Problem 1.24. What will be its final output information if the usual 0011 and 0101 sequences are used as inputs?

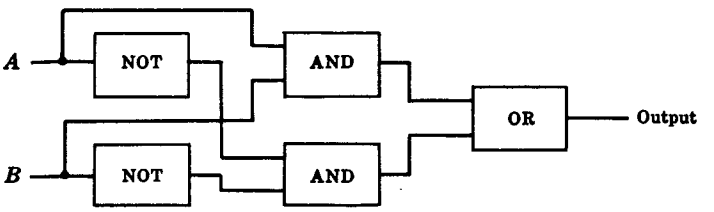

Fig. 1-25

1.27. The circuit of Fig. 1-26 does no useful information processing. Explain.

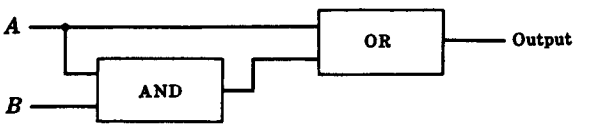

Fig. 1-26

1.28. Show that the circuit of Fig. 1-27 always gives the same output information as the comparator of Fig. 1-21, by finding the performance of each machine with the usual input sequences, 0011 for A and 0101 for B.

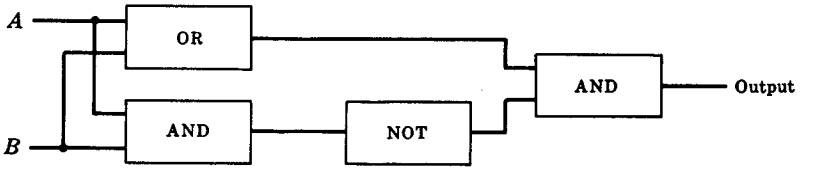

Fig. 1-27

1.29. Show that the circuit of Fig. 1-28 also gives the same output information as the comparator.

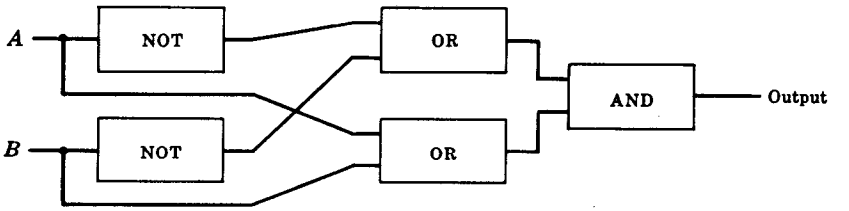

Fig. 1-28

1.30. Why are the two circuits of (*a*) Fig. 1-29 and (*b*) Fig. 1-30 not useful information processors?

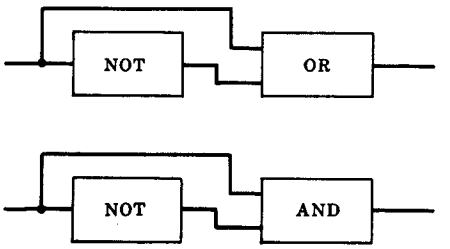

Fig. 1-29

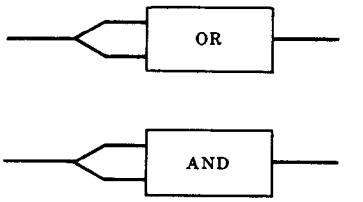

Fig. 1-30

Chapter 2

Boolean Algebra

Boolean algebra has been found to be very useful in systematically combining the simple AND, OR, NOT circuits of the preceding chapter into more complex circuits designed for information processing. This chapter presents a brief introduction to this important modern mathematical structure, in the form most suited to our needs. It is not an algebra of numbers but, for our purposes at least, of sequences of zeros and ones. There are three operations. Two sequences A and B may be *added* by treating them exactly as an OR circuit would do. The result will be called $A+B$. Thus for our two basic sequences the computation runs as follows.

$$\begin{array}{rr} A & 0\ 0\ 1\ 1 \\ B & 0\ 1\ 0\ 1 \\ \hline A+B & 0\ 1\ 1\ 1 \end{array}$$

The sum is the sequence 0111. As the example suggests, each column (each pair of input values) is added separately, making

$$0+0=0, \quad 0+1=1, \quad 1+0=1, \quad 1+1=1$$

These are the basic addition facts of Boolean algebra. The OR circuit implements this Boolean addition operation electrically, and we shall use the symbols OR and + more or less interchangeably.

In the second Boolean operation two sequences are *multiplied,* by treating them exactly as an AND circuit would do. The result will be called AB. For our two basic sequences the computation runs as follows.

$$\begin{array}{rr} A & 0\ 0\ 1\ 1 \\ B & 0\ 1\ 0\ 1 \\ \hline AB & 0\ 0\ 0\ 1 \end{array}$$

The product is the sequence 0001. As the example suggests, each column (each pair of input values) is multiplied separately, making

$$0\times 0=0, \quad 0\times 1=0, \quad 1\times 0=0, \quad 1\times 1=1$$

Here the $\times$ denotes multiplication. These are the basic multiplication facts of Boolean algebra. The AND circuit implements this Boolean multiplication process electrically, and we shall use the symbols $\times$ and AND interchangeably.

The third and last operation is inversion. Any sequence may be *inverted* by replacing zeros by ones and vice versa, that is, by treating it exactly as a NOT circuit would do. The inverse of a sequence A will be denoted $\bar{A}$. The operation is summarized by the two basic inversion facts

$$\bar{0}=1, \quad \bar{1}=0$$

one of which is applied to each value of the sequence to be inverted. These three operations form the foundation of a Boolean algebra.

Example 2.1.

Given the sequences $A = 1010101$ $B = 0000011$

compute $A+B$, AB and $\bar{A}$. Actually, this has already been done in the preceding chapter. Thus

$$A+B = 1010111$$

the zeros coming from $0+0$. Similarly

$$AB = 0000001 \quad \text{and} \quad \bar{A} = 0101010$$

For the rest of this chapter the equality symbol will mean "is exactly the same sequence as", just as it does in this example.

The theory of Boolean algebra is fairly extensive, but only a limited selection will be needed here. To begin at the simplest level, several important theorems announce properties which any single sequence will have. For instance, let $\emptyset$ (read OH) denote a sequence containing only zeros. Similarly I will be a sequence containing only ones.

$\emptyset$ all zeros
I all ones

It seems clear that if any sequence at all is multiplied by $\emptyset$ the product will be the sequence $\emptyset$. If we choose $A = 01$ the computation

$$\begin{array}{rr} A & 0\ 1 \\ \emptyset & 0\ 0 \\ \hline A\times\emptyset & 0\ 0 \end{array}$$

quickly confirms this suspected result, at least for the particular A selected. But, and here we come to a very important point, with any other A sequence, though the computation may become longer it will just duplicate the two columns we already have over and over again. These two columns are sufficient to tell the whole story. Since $A\emptyset = \emptyset$ in this special case, it always will for any A sequence at all. This method of proving a theorem by examining a well chosen special case is entirely satisfactory here because, when properly selected, the special case will include all the possibilities which can arise.

Proof by testing all the possibilities, as the method just used is often called, is particularly effective in developing the theory of Boolean algebra. It will be exploited fully in the solved problems. Some of the results to be obtained will be listed here for ready reference.

Theorem 2.1.	$A\emptyset = \emptyset$	***Theorem 2.6.***	$AA = A$
Theorem 2.2.	$A+I = I$	***Theorem 2.7.***	$\bar{\emptyset} = I,\ \bar{I} = \emptyset$
Theorem 2.3.	$A+\emptyset = A$	***Theorem 2.8.***	$\bar{\bar{A}} = A$
Theorem 2.4.	$AI = A$	***Theorem 2.9.***	$A+\bar{A} = I$
Theorem 2.5.	$A+A = A$	***Theorem 2.10.***	$A\bar{A} = \emptyset$

Slightly more complex are various theorems which state properties of any pair of sequences. For example, with $A = 0011$ and $B = 0101$ the two sequences $\overline{A+B}$ and $\bar{A}\bar{B}$ are computed as follows.

$$\begin{array}{rr} A & 0\ 0\ 1\ 1 \\ B & 0\ 1\ 0\ 1 \\ A+B & 0\ 1\ 1\ 1 \\ \overline{A+B} & 1\ 0\ 0\ 0 \\ \bar{A} & 1\ 1\ 0\ 0 \\ \bar{B} & 1\ 0\ 1\ 0 \\ \bar{A}\bar{B} & 1\ 0\ 0\ 0 \end{array}$$